Emergent self-awareness in multi-sensor physical agents

Giulia Slavic

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in
Electrical Engineering, Electronics and Automation

Department of Electrical, Electronic, Telecommunications Engineering and Naval Architecture (DITEN)
University of Genoa

Department of Systems Engineering and Automation (DISA)
University Carlos III of Madrid

Advisors:
Carlo Regazzoni
David Martín Gómez
Lucio Marcenaro

Defense Month: May 2024

Emergent self-awareness in multi-sensor physical agents

Giulia Slavic

Joint Doctorate in Interactive and Cognitive Environments
JD-ICE

XXXVI cycle

Acknowledgements

This PhD Thesis has been developed in the framework of, and according to, the rules of the Joint Doctorate in Interactive and Cognitive Environments JD-ICE with the cooperation of the following Universities:

Università degli Studi di Genova (UNIGE)

DITEN - Dept. of Electrical, Electronic, Telecommunications Engineering and Naval Architecture

ISIP40 - Information and Signal Processing for Cognitive Telecommunications

Primary Supervisor: Prof. Carlo Regazzoni
Co-Supervisor: Prof. Lucio Marcenaro

Universidad Carlos III de Madrid (UC3M)

IEEA - Doctoral Program in Electrical Engineering, Electronics and Automation

DISA - Dep. of Systems Engineering and Automation

ISL - Intelligent System Laboratory
Supervisor: Prof. David Martín Gomez

Acknowledgements

First, I am grateful to my supervisors. I extend my sincere thanks to Prof. Carlo Regazzoni for his guidance and consistent feedback during these years of research. Having a supervisor that provides such assiduous assistance is something that cannot be taken for granted. I am grateful to Prof. David Martín Gómez for his patient guidance and support, both during the theoretical development of my work and during the extraction of the experimental datasets used for evaluation of the developed methods. I express my gratitude to Prof. Lucio Marcenaro for his constant availability and assistance.

I also extend my gratitude to Prof. Pamela Zontone for the invaluable help that she has provided during the last months of my Ph.D. journey.

Special thanks go to all my collegues at the University of Genoa and at the University Carlos III of Madrid for providing me with a friendly research environment. I thank the co-authors of my papers, in particular Pablo Marín Plaza, Damián Andrés Campo Caicedo, Mohamad Baydoun, Hafsa Iqbal, Ali Krayani, Abrham Shiferaw Alemaw, Muhammad Adnan, and Tommaso Apicella. I am also thankful to Sheida Nozari, Zhuoyao He, and David Yagüe Cuevas for their friendship.

Last but not least, I deeply thank my parents, Anna Maria Pelle and Saverio Slavic for their love, support, and for transmitting me the value of culture ever since I was a child.

Published and Submitted Content

0.1 Journal Papers

1. **G. Slavic**, M. Baydoun, D. Campo, L. Marcenaro, C. Regazzoni, "Multilevel Anomaly Detection Through Variational Autoencoders and Bayesian Models for Self-Aware Embodied Agents," *IEEE Transactions on Multimedia*, vol. 24, pp. 1399-1414, 2021, doi: 10.1109/TMM.2021.3065232.

   (This journal paper is wholly included in the thesis, in Chapter 5. The material from this source included in this thesis is not singled out with typographic means and references.)

2. **G. Slavic**, A. S. Alemaw, L. Marcenaro, D. Martín Gómez and C. Regazzoni, "A Kalman Variational Autoencoder Model Assisted by Odometric Clustering for Video Frame Prediction and Anomaly Detection," in *IEEE Transactions on Image Processing*, vol. 32, pp. 415-429, 2023, doi: 10.1109/TIP.2022.3229620.

   (This journal paper is wholly included in the thesis, in Chapter 7. The material from this source included in this thesis is not singled out with typographic means and references.)

0.2 Conference Papers

1. **G. Slavic**, D. Campo, M. Baydoun, P. Marin, D. Martin, L. Marcenaro, C. Regazzoni, "Anomaly Detection in Video Data Based on Probabilistic Latent Space Models," *2020 IEEE Conference on Evolving and Adaptive Intelligent Systems (EAIS)*, Bari, Italy, 2020, pp. 1-8, doi: 10.1109/EAIS48028.2020.9122766.

(This conference paper is partly included in the thesis, in Chapter 5. The material from this source included in this thesis is not singled out with typographic means and references.)

2. D. Campo, **G. Slavic**, M. Baydoun, L. Marcenaro, C. Regazzoni, "Continual Learning of predictive models in video sequences via Variational Autoencoders," *2020 IEEE International Conference on Image Processing (ICIP)*, Abu Dhabi, United Arab Emirates, 2020, pp. 753-757, doi: 10.1109/ICIP40778.2020.9190980.

   (This conference paper is partly included in the thesis, in Chapter 5. The material from this source included in this thesis is not singled out with typographic means and references.)

3. **G. Slavic**, P. Marin, D. Martin, L. Marcenaro and C. Regazzoni, "Interpretable Anomaly Detection Using A Generalized Markov Jump Particle Filter," *2021 IEEE International Conference on Autonomous Systems (ICAS)*, Montreal, QC, Canada, 2021, pp. 1-5, doi: 10.1109/ICAS49788.2021.9551111.

   (This conference paper is wholly included in the thesis, in Chapter 6. The material from this source included in this thesis is not singled out with typographic means and references.)

4. **G. Slavic**, A. S. Alemaw, L. Marcenaro and C. Regazzoni, "Learning Of Linear Video Prediction Models In A Multi-Modal Framework For Anomaly Detection," *2021 IEEE International Conference on Image Processing (ICIP)*, Anchorage, AK, USA, 2021, pp. 1569-1573, doi: 10.1109/ICIP42928.2021.9506049.

   (This conference paper is wholly included in the thesis, in Chapter 7. The material from this source included in this thesis is not singled out with typographic means and references.)

5. **G. Slavic**, P. M. Plaza, L. Marcenaro, D. M. Gómez and C. Regazzoni, "Simultaneous Localization and Anomaly Detection from First-Person Video Data through a Coupled Dynamic Bayesian Network Model," *2022 18th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, Madrid, Spain, 2022, pp. 1-8, doi: 10.1109/AVSS56176.2022.9959613.

(This conference paper is wholly included in the thesis, in Chapter 8. The material from this source included in this thesis is not singled out with typographic means and references.)

6. **G. Slavic**, P. M. Plaza, L. Marcenaro, D. M. Gómez and C. Regazzoni, "Localización y detección de anomalías utilizando imágenes en un marco bayesiano," *XLIV Jornadas de Automática*, Zaragoza, Spain, 2023, pp. 885-890,
doi: 10.17979/spudc.9788497498609.885.

   (This conference paper is wholly included in the thesis, in Chapter 8. The material from this source included in this thesis is not singled out with typographic means and references.)

0.3 Book Chapters

1. C. Regazzoni, A. Krayani, **G. Slavic**, L. Marcenaro, "Probabilistic Anomaly Detection Methods Using Learned Models from Time-Series Data for Multimedia Self-Aware Systems" in "Advanced Methods and Deep Learning in Computer Vision", E. R. Davies, O. Camps, M. Turk, 1st October 2021, pp. 449-479, doi: 10.1016/B978-0-12-822109-9.00022-9.

   (This book chapter is partly included in the thesis, in Chapters 2, 3 and 5. The material from this source included in this thesis is not singled out with typographic means and references.)

Other Research Merits

0.4 Journal Papers

1. M. Adnan, **G. Slavic**, D. Martín Gómez, L. Marcenaro, C. Regazzoni, “Systematic and Comprehensive Review of Clustering and Multi-Target Tracking Techniques for LiDAR Point Clouds in Autonomous Driving Applications," *Sensors*, vol. 23, pp. 6119, 2023, doi: 10.3390/S23136119.

0.5 Conference Papers

1. J. L. Patino, J. Boyle, J. M. Ferryman, J. Auer, J. Pegoraro, R. P. Pflugfelder, M. Cokbas, J. Konrad, P. Ishwar, **G. Slavic**, L. Marcenaro, Y. Jiang, Y. Jin, H. Ko, G. Zhao, G. Ben-Yosef, J. Qiu, “PETS2021: Through-foliage detection and tracking challenge and evaluation," *IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, Washington, DC, USA, 2021, pp. 1-10, doi: 10.1109/AVSS52988.2021.9663837.

2. T. Apicella, **G. Slavic**, E. Ragusa, P. Gastaldo, L. Marcenaro, “Container Localisation and Mass Estimation with an RGB-D Camera," *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Virtual and Singapore, 2022, pp. 9152-9155, doi: 10.1109/ICASSP43922.2022.9747134.

3. A. S. Alemaw, **G. Slavic**, H. Iqbal, L. Marcenaro, D. Martín Gómez, C. Regazzoni, “A Data-Driven Approach for the Localization of Interacting Agents via a Multi-Modal Dynamic Bayesian Network Framework," *IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, Madrid, Spain, 2022, pp. 1-8, doi: 10.1109/AVSS56176.2022.9959648.

4. **G. Slavic**, M. Bracco, L. Marcenaro, D. Martín Gómez, C. Regazzoni, P. Zontone, “Joint data-driven analysis of visual-odometric anomaly signals in generative AI-based

agents," IEEE ICASSP 2024 2nd Workshop on Signal Processing for Autonomous Systems.

Abstract

The cognitive approach to the development of autonomous vehicles takes inspiration from human reasoning, and, conversely to the computationalist approach, rejects formulating fixed mathematical models to describe each possible behavior of vehicles and objects around them. The computationalist approach indeed has a weakness: developers cannot examine and mathematically formulate all possible real-world situations that drivers may encounter. Cognitive approaches provide a solution to this problem, as they suggest that vehicles should continually learn through experience as humans do, which would allow them to progressively grasp rare and unexpected behaviors. This thesis mainly addresses the pivotal question of detecting when some unexpected behavior is occurring - referred to as anomaly detection - within a cognitive self-awareness framework. The adopted framework is characterized by several desirable characteristics, such as being Bayesian, hierarchical, multi-sensorial, data-driven, and interpretable.

A set of modules for anomaly detection and localization of an agent are proposed. Dynamic Bayesian Networks are used, as they are interpretable probabilistic models allowing hierarchical representation of variables potentially coming from multiple sensors; moreover, the links between variables inside Dynamic Bayesian Networks can be learned from data. All methods in the thesis adopt a filter called the Markov Jump Particle Filter that can be described through a Dynamic Bayesian Network on a minimum of three levels. When elaborating image data, Variational Autoencoders are adopted to perform dimensionality reduction while maintaining a probabilistic representation; novel methods for joining Variational Autoencoders and Bayesian filters are proposed.

This thesis focuses on the elaboration of vehicular video and odometry data. First, self-awareness anomaly detection approaches to separately handle video and odometry data are proposed; then, an approach fusing the two modalities is introduced; the capability to localize the vehicle is also added.

The proposed methods are evaluated on real-world and simulated data from terrestrial and aerial vehicles.

Resumen

El enfoque cognitivo para el desarrollo de vehículos autónomos se inspira en el razonamiento humano y, a la inversa del enfoque computacional, rechaza la formulación de modelos matemáticos fijos para describir cada posible comportamiento de los vehículos y objetos alrededor de ellos. En efecto, el enfoque computacional tiene una debilidad: los desarrolladores no pueden examinar y formular matemáticamente todas las posibles situaciones del mundo real que los conductores podrían encontrar. Los enfoques cognitivos proporcionan una solución a este problema, ya que sugieren que los vehículos deberían aprender continuamente a través de la experiencia como hacen los humanos, lo que les permitiría reconocer progresivamente comportamientos raros e inesperados. Esta tesis trata principalmente la question fundamental de detectar cuándo está ocurriendo algún comportamiento inesperado - definida como detección de anomalías - dentro de un marco de autoconciencia cognitiva. El marco adoptado tiene características deseables, como ser bayesiano, jerárquico, multisensorial, basado en datos e interpretable.

Se propone un conjunto de módulos para la detección de anomalías y localización de un agente. Se utilizan Redes Bayesianas Dinámicas, ya que son modelos probabilísticos interpretables que permiten una representación jerárquica de variables potencialmente provenientes de sensores múltiples; además, los vínculos entre variables dentro de las Redes Bayesianas Dinámicas se pueden aprender de los datos.

Todos los métodos en la tesis adoptan un filtro llamado Filtro de Partículas de Salto de Markov, que puede describirse a través de una Red Bayesiana Dinámica con un mínimo de tres niveles. Los datos de imágenes se procesan usando un Autocodificadores Variacionales para realizar una reducción de dimensionalidad manteniendo una representación probabilística; se proponen métodos novedosos para unir Autocodificadores Variacionales y filtros Bayesianos.

Esta tesis se enfoca en la elaboración de datos de vehículos de video y odometría. En primer lugar, se proponen enfoques de detección de anomalías para elaborar por separado datos de vídeo y odometría; luego se introduce un enfoque que fusiona las dos modalidades; también se añade la capacidad de localizar el vehículo.

Los métodos propuestos se evalúan con datos del mundo real y simulados, tanto de vehículos terrestres como aéreos.

Table of contents

List of figures

List of tables

Nomenclature

Acronyms / Abbreviations

2D	two-dimensional
3D	three-dimensional
ACC	Accuracy
AKF	Adapted Kalman Filter
AMJPF	Adapted Markov Jump Particle Filter
ATI	Abnormal Turn at an Intersection
AUC	Area Under the Curve
AV	Autonomous Vehicle
BN	Bayesian Network
CAE	Convolutional Autoencoder
CG-KVAE	Cluster-Guided Kalman Variational Autoencoder
CMJPF	Coupled Markov Jump Particle Filter
CNN	Convolutional Neural Network
ConvLSTM	Convolutional Long Short-Term Memory
DBN	Dynamic Bayesian Network
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DDBN	Deep Dynamic Bayesian Network

DGM Directed Graphical Model

DL Deep Learning

DVAE Dynamical Variational Autoencoder

EKF Extended Kalman Filter

ES Emergency Stop

FM Frontal Motion

FN False Negative

FP False Positive

FPR False Positive Rate

FPV First-Person Viewpoint

FV First Version

G-MJPF Generalized Markov Jump Particle Filter

GAN Generative Adversarial Network

GDBN Generalized Dynamic Bayesian Network

GE Generalized Error

GM Generative Model

GMM Gaussian Mixture Model

GNG Growing Neural Gas

GNSS Global Navigation Satellite System

GO Generalized Observation

GPS Global Positioning System

GS Generalized State

GT Ground Truth

HDBN Hierarchical Dynamic Bayesian Network

IMU Inertial Measurement Unit

KF Kalman Filter

KLDA Kullback-Leibler Divergence Abnormality

KLD Kullback Leibler-Divergence

KNN K-nearest-neighbours

KVAE Kalman Variational Autoencoder

LAM Lateral Motion

LOM Loop Motion

LSTM Long Short-Term Memory

MJPF Markov Jump Particle Filter

ML Machine Learning

MSE Mean Squared Error

NFF Null Force Filter

NN Neural Network

OF-AKF Optical Flow Adapted Kalman Filter

OF-AMJPF Optical Flow Adapted Markov Jump Particle Filter

OF Optical Flow

OOS Out-Of-Street

PA Pedestrian Avoidance

PCA Principal Component Analysis

PC Point Cloud

PF Particle Filter

PGM Probabilistic Graphical Model

PM Perimeter Monitoring

R-MJPF Rotational Markov Jump Particle Filter

RNN Recurrent Neural Network

ROC Receiver Operating Characteristic

SA Self-Awareness

SD Sudden Deceleration

SLAM Simultaneous Localization and Mapping

SLDS Switching Linear Dynamical System

SOM Self-Organizing Map

SVM Support Vector Machine

SV Second Version

TN True Negative

TPR True Positive Rate

TP True Positive

TTM Temporal Transition Matrix

UGM Undirected Graphical Model

UKF Unscented Kalman Filter

V-SLAM Visual Simultaneous Localization and Mapping

VAE Variational Autoencoder

VBL Visual-Based Localization

VFE Variational Free Energy

VIO Visual Inertial Odometry

VO Visual Odometry

Chapter 1

Introduction

1.1 Motivation

The aim of Autonomous Vehicles (AVs) is to reduce - or remove - the necessity of human intervention during the execution of the vehicles' functions. The field of AVs has been evolving since the 1980s and is anticipated to offer benefits across several domains, including transportation, agriculture, disaster response, security, and surveillance [238, 59]. Two main approaches can be identified in the AV field: computationalist and cognitive. The computationalist approach defines fixed mathematical rules that describe the behavior of both the vehicle and the agents it interacts with. In contrast, the cognitive approach recognizes the infeasibility of formulating rules for all possible real-world situations. AVs should instead continually learn from experience, similarly to what humans do. This approach is, thus, called "cognitive" because it takes inspiration from human cognition.

A great variety of cognitive architectures have been developed across the last decades [129]. One relevant concept within the cognitive approach is self-awareness. Self-awareness can be described as the "capacity of becoming the object of one's own attention" [162]. Therefore, a self-aware agent directs its attention not solely on the external environment, but also on its own state. AVs can do this by simultaneously monitoring externally oriented sensors such as cameras or lidar (named "exteroceptive sensors"), and internally oriented sensors such as Inertial Measurement Units (IMUs), or sensors that monitor steering, braking, or wheel speed (named "proprioceptive sensors").

The approaches proposed in this thesis are based on the cognitive self-awareness framework presented by Regazzoni et al. [183]. This framework is inspired by the work of neuroscientists such as Friston [173, 74, 73, 77, 13, 76], Haykin [94], and Damasio [45]. Six necessary capabilities of self-aware systems are identified in this framework: initialization, memorization, prediction, anomaly detection, new model creation, and decision-making.

Initialization is the creation of an initial model, whereas memorization refers to retaining information for future use. Through prediction, the agent can estimate how itself and the environment will evolve. It can then compare predictions and new observations through anomaly detection. Predictions that differ from observations warn that something unexpected has happened and that the available model is not reliable anymore, triggering the creation of a new model. Finally, during decision-making, the agent chooses the optimal actions to perform, based on observations and on the available models. Anomaly detection allows for identifying when a new model should be trained. Through these six capabilities, the agent is capable of learning new rules that continually emerge from the data. For this reason, the self-awareness is defined as "emergent" [131, 182]. In this thesis, the first four capabilities are analyzed, with a special focus on the fourth one, i.e., anomaly detection. It is important to notice how anomaly detection has many uses outside of the self-awareness context too, such as fraud detection [161, 100], social media security [79], medical image anomaly detection [102, 16], video and audio surveillance [168, 15, 181].

This thesis does not address the last two self-awareness capabilities (i.e., new model creation and decision-making), leaving it for future research. The problem of localization is instead considered, as it is fundamental for autonomous vehicles, especially in video surveillance applications.

The framework proposed in [183] was employed for low-dimensional positional [17] and control [115] data and, subsequently, for high-dimensional lidar data [105]. Image data were not explored, nor was the combination of a low-dimensional and a high-dimensional sensor. However, image data are of significance for autonomous cognitive systems because cameras are cheap, and the representation gained from them is similar to the one elaborated by the human brain. This representation is rich, but also complex, comprised of millions of features that are intractable without further processing. Additionally, image content is typically a highly non-linear function of the system dynamics. Therefore, modeling of image data is non-trivial.

This thesis builds on the self-awareness architecture proposed by Regazzoni et al. [2] by exploring its application to image data, by further studying positional data, and by ultimately fusing the two modalities. The primary objective of the thesis is to develop a self-awareness model for autonomous vehicles employing image and odometry data. Images are extracted from a camera that observes the environment and that is mounted on the vehicle; therefore this type of sensor is exteroceptive. Odometry data can be obtained through different sensors: for instance, from Global Positioning System (GPS), IMUs, external cameras, or lidar. Sequential global positioning data carry both exteroceptive information (the location of the

vehicle in the environment) and proprioceptive information (how the vehicle is speeding, accelerating, or rotating).

The architecture is designed to take inspiration from human reasoning and, therefore, to have several bio-inspired characteristics, including being probabilistic, hierarchical, data-driven, and multi-sensorial. Furthermore, the architecture is also interpretable, which is a desirable characteristic for AVs. Several concepts inspired from the neuroscientist Karl Friston are also employed in the proposed approach, such as the concept of Generalized States (GS) [77, 13], linear attractors [77] and minimization of free energy [76, 173]. GSs cointain the states per se, their velocities, accelerations and higher-level derivative information. In this thesis, several methods to predict how the GSs evolve across time are proposed. GSs can also be clustered, and clusters of GSs identify a specific semantic information: for examples, if the state corresponds to the vehicle's position, the GS could contain position and velocity. A GS cluster includes data with similar position and velocity. The evolution of the GS can then be also described at a higher hierarchical level as the transition between these clusters.

Dynamic Bayesian Networks (DBNs) are used, as they are probabilistic, interpretable methods allowing hierarchical modeling of variables. DBNs are a type of Probabilistic Graphical Model (PGM), graph-based representations that encode causal relationships between random variables. DBNs also delineate the dynamics of random variables over time. A key benefit of DBNs lies in their hierarchical structure. They depict causal relationships between high-level variables, which capture abstract semantic information about the world, and low-level distributions, which describe raw sensory data from the environment. We adopt a filter called Markov Jump Particle Filter (MJPF) [17] that can be represented as a DBN on a minimum of three levels and that combines a set of Kalman Filters (KFs) with a Particle Filter (PF). However, a DBN cannot be directly employed on high-dimensional data such as images. Therefore, a method for dimensionality reduction must first be adopted. We choose to use a Variational Autoencoder (VAE) [122] because it can perform dimensionality reduction of the images and at the same time maintain a probabilistic representation. Using the VAE bottleneck, we obtain a Generalized State including information about the image content and the motion between images. Furthermore, the Growing Neural Gas (GNG) algorithm [78] is adopted to cluster the Generalized States.

To achieve the above-mentioned prediction capability, the method must be able to predict how the GSs change over time. In this thesis, several novel possibilities are analyzed to combine the described models and to perform prediction. First, we explore a method based only on image data, where the prediction is performed through neural networks (NNs). Secondly, we examine the case of positional data prediction. Thirdly, we combine the odometry and image data employing a Kalman Variational Autoencoder (KVAE) [71].

In the proposed architecture, we guide the learning of the video architecture leveraging the odometry GS. We call this model Cluster Guided KVAE (CG-KVAE). Indeed, image content is complex and is a highly non-linear function of the system dynamics, whereas low-dimensional odometry information is simpler. Therefore, odometry GSs and models can be employed to aid the training phase of the image models.

Prediction is performed in this case through linear models. We also execute vehicle localization from the camera images, i.e., Visual-Based Localization (VBL). Two different ways of fusing image and odometry data are proposed in this approach. In all cases, an MJPF is adopted when performing prediction and anomaly detection, adapted to each of the considered cases.

The use of VAEs, KVAEs, and MJPFs allows us to obtain a method with the characteristics mentioned on the previous page, i.e., being probabilistic, hierarchical, data-driven, explainable, and multi-sensorial. Probabilistic modeling is achieved because VAEs and MJPFs are probabilistic methods. The MJPF enables the hierarchical modeling of variables. The VAE and the vocabulary of the MJPF can be learned directly from data, yielding a data-driven method; the combination of odometry and video data makes it multi-sensorial too. In addition, DBNs are inherently interpretable models. Further explainability can also be obtained from the anomalies: each anomaly has a different meaning, and the combination of detected anomalies can help to identify the cause of the abnormal event.

We evaluate the methods proposed in this thesis on real-world datasets (and one simulated dataset) from both terrestrial vehicles and Unmanned Aerial Vehicles (UAVs). The terrestrial vehicles move according to simpler motion patterns and mostly along two directions, whereas the UAVs can exhibit more complex motions along three dimensions.

To summarize, the main aims of this thesis are the following:

- The development of a self-awareness architecture for autonomous vehicles inspired from human reasoning, and that incorporates characteristics such as being probabilistic, hierarchical, data-driven, explainable, and multi-sensorial;
- The use of anomaly detection inside this architecture to identify new rules that continually emerge from the data and that signal the necessity to build a new model;
- The employement of low and high dimensional data, which should be handled as homogeneously as possible;
- The localization of the vehicle in the environment, as an additional capability of the architecture.

1.2 Thesis Outline

Firstly, **Chapter 2** describes some background topics necessary to understand the methods proposed in this thesis. The concepts and algorithms explored in this chapter can be applied in many fields and not only for AVs. Some of the concepts treated are general and well-established, whereas others are more recent and specific. The main topics covered are DBNs, Bayesian Filters, and VAEs (both static and dynamical). They are fundamental in this thesis, as all proposed approaches are modelled through DBNs and use the MJPF, which is a Bayesian Filter; all proposed methods for high-dimensional data employ an evolution of a VAE. Indeed, the framework underlying this thesis uses DBNs and Bayesian Filters, and is introduced in Chapter 3.

Chapter 3 analyses the state-of-the-art approaches related to self-awareness for autonomous systems, anomaly detection and Visual-Based Localization, which are the main fields and applications tackled by this thesis. Therefore, conversely to the Background chapter, this chapter does not consider general methods applicable to many fields, but specifically examines the AVs field. Concepts such as awareness, self-awareness, and explainability are introduced, as well as some relevant ideas from the neurosciences. A literature review of anomaly detection and VBL is provided. Moreover, Chapter 3 presents the framework adopted in this thesis and the reasons behind its adoption, identifying six capabilities and five characteristics that are desirable for self-aware systems. These capabilities and characteristics were already described in the introduction, but are examined in detail in Chapter 3. Furthermore, we discuss how they take inspiration from neuroscience concepts. Finally, other cognitive approaches are presented and compared with the one adopted in this thesis.

Chapter 4 is centered on the extraction, description and analysis of datasets. We distinguish this chapter from the following ones because of its transitional purpose and more technical and traditional nature. First, it introduces the experimental setup for a drone and lidar dataset extraction and it proposes the traditional Signal Processing methods adopted to perform localization of the drone. Then, it examines all the datasets that are used for evaluation throughout the thesis. Finally, it discusses in detail characteristics of some of the used datasets that will be important in later chapters. In the first part of this chapter, we consider drone localization in indoor environments, which poses a significant issue due to the unreliability or lack of GPS signal and the impossibility of adopting methods such as wheel odometry. However, drone positional information is crucial in surveillance applications. To address this issue, this chapter proposes a method for drone localization that uses onboard sensors (e.g., camera and IMUs) and ArUco markers placed in the environment. The positions estimated from Visual Inertial Odometry (VIO) are combined with the ones from marker detection through a KF. Then, an external lidar sensor is employed to track

the drone. This latter localization serves as a ground truth for the former one, and for possible successive elaborations of the positional data. The two estimations are aligned and synchronized. The method is evaluated using a DJI FLY 2S drone and an Ouster OS1 lidar in a closed environment.

In **Chapters 5 to 8**, we then describe the proposed methods, and we display and discuss the results. Each chapter is dedicated to one approach. The method adopted in Chapter 5 only uses camera data, whereas the one presented in Chapter 6 only employs odometry data. In Chapters 7, and 8, the combination of camera and odometry data is considered. In all these chapters except Chapter 8, terrestrial vehicular data are adopted; on the other hand, Chapter 8 considers both terrestrial and aerial data.

A more detailed explanation of the content of Chapter 5 to Chapter 9 is outlined in the following paragraphs.

The approach proposed in **Chapter 5** combines DBNs and NNs to detect anomalies in video data at different abstraction levels. We use a VAE to reduce the dimensionality of video frames. Two versions of the approach are proposed and compared. The first one trains the VAE only on the video frames and uses the difference between consequent latent states of the VAE to build the Generalized States. The second method extracts the Optical Flows (OFs) between subsequent images, and trains the VAE on both the video frames and the OFs, thus generating a low-dimensional latent space that captures both visual and dynamical information. In both cases, a novel MJPF extension is employed to predict the successive frames and detect anomalies in video data. The two methods are compared. Our evaluation procedure is executed using video data from a semi-autonomous vehicle performing different tasks in a closed environment. Additionally, tests on benchmark anomaly detection datasets are conducted.

Chapter 6 tackles the problem of interpretability. It presents an approach for learning prediction models and detecting anomalies by decomposing the evolution of an agent's state into its different motion characteristics, such as the way it is accelerating or oscillating. Anomaly signals are computed on the different motion parameters, allowing us to better understand what the cause of the abnormality might be. Notably, when performing anomaly detection on an autonomous vehicle's sensory data, it is fundamental to infer the cause of the possible anomalies. A filter is introduced based on a four-level DBN to increase the interpretability of the results with respect to previous methods in literature. The proposed anomaly detection method is tested on data from a real vehicle. We also consider the case in which multiple models are learned, analyzing how to extract the salient discriminatory features from each, and use the proposed anomaly detection method to perform driver behavior classification.

As we have observed, the combination of different sensory information to predict upcoming situations is an innate capability of intelligent beings. Therefore, **Chapter 7** proposes a method for video-frame prediction and anomaly detection that leverages odometric data. We adopt a DBN framework, which is combined with the use of Deep Learning (DL) methods to learn an appropriate latent space. First, a MJPF is built over the odometric data. The odometry information is clustered, and a set of clusters is obtained. As a second step, the video model is learned. It is composed of a KVAE modified to exploit the odometry clusters so that, during learning, it focuses its attention on features related to the dynamic tasks that the vehicle is performing. We denote the obtained overall model as Cluster-Guided Kalman Variational Autoencoder (CG-KVAE). Evaluation is conducted using data from a car moving in a closed environment and leveraging a part of the University of Alcalá DriveSet dataset, where several drivers move in a normal and drowsy way along a secondary road.

Chapter 8 proposes a method to simultaneously perform anomaly detection and VBL, within the adopted self-awareness framework for autonomous vehicles, by combining Deep Learning with traditional Signal Processing approaches. In the proposed method, a DBN model is learned considering the positional and video data. A modified CG-KVAE is introduced. In the online phase, a Coupled Markov Jump Particle Filter (CMJPF) is proposed to simultaneously execute anomaly detection and VBL. The extracted anomalies are used to improve the localization process, restarting some particles of the Particle Filter when anomalies exceed a threshold. The method is evaluated on trajectories extracted with the Carla simulator and on four real-world datasets, including two of the datasets extracted through the set up and the method proposed in Chapter 4.

Finally, **Chapter 9** draws some conclusions about the approaches proposed in the thesis and discusses future research directions.

1.3 Summary of Contributions

The main contributions of this thesis are summarized as follows:

- The introduction of an anomaly detection approach for image data within a DBN framework, that allows for anomaly detection at different hierarchical levels, and, therefore, provides some first insight into the anomaly's source. Two versions of this approach are presented and compared (Chapter 5).
- A novel anomaly detection approach for low-dimensional data, that breaks down agent motion into different directions and parameters to enhance interpretability (Chapter 6).

- The extension of the low-dimensional data approach to driver behavior classification (Chapter 6).
- The presented approaches ensure that low levels of the DBN are sensor-specific, while high levels are treated as uniformly as possible across sensory modalities. In this way, DBNs of diverse sensors can be joined more easily. Different levels of homogeneity are reached in the various approaches and are discussed in the thesis.
- The fusion of video and odometry data is consequently performed. A novel fusion method is proposed, in which the simpler odometry model is used to guide the learning of the more complex camera model (Chapter 7).
- An alternative fusion model for video and odometry data is introduced, where the learning of the video model is guided by both video and odometry data (Chapter 8).
- Both fusion approaches are extended to the application of VBL, thus proposing a novel VBL approach mixing traditional tracking methods and Deep Learning (Chapter 8).
- Different approaches are proposed for combining traditional Bayesian filters, including the Kalman Filter, Particle Filter, Unscented Kalman Filter (UKF), and MJPF, with deep learning approaches such as Variational Autoencoders.
- A Signal Processing method is introduced to extract the drone positioning for an indoor evaluation dataset. This dataset serves as a valuable resource for evaluating anomaly detection and VBL approaches. Drone camera data and IMU are adopted to self-locate the drone. The ground truth position is determined by tracking the drone on the lidar data and by aligning the result with the localization from the drone's onboard sensors (Chapter 4).

Chapter 2

Background and Preliminaries

This chapter introduces some background methods and algorithms necessary to understand the rest of the thesis. These include both more general and well-established concepts such as DBNs, and more specific and recent algorithms, such as Dynamical Variational Autoencoders (DVAEs). What is introduced in this chapter is relevant to understanding the methods mentioned in the State of the Art chapter and the methods proposed in this thesis. It was chosen to separate the "Background and Preliminaries" chapter from the "State of the Art" chapter because the former introduces concepts and models applicable in many fields and for many applications, whereas the latter concentrates specifically on the field of Autonomous Systems and on the anomaly detection and Localization applications. For example, in the current chapter, DBNs are introduced, and the basic concepts related to them are explained. In contrast, Chapter 3 explores how DBNs can be used in a self-awareness model for Autonomous Systems.

The structure of the current chapter is outlined as follows.

First, we define the difference between discriminative and generative models (Section 2.1) and mention some examples of the latter case. We then focus on Bayesian Networks (Section 2.2) and, specifically, on DBNs (Section 2.3). DBNs are generative models and their knowledge is fundamental to understand the framework on which this thesis is based.

Section 2.4 introduces how filters for probabilistic inference can be represented using DBNs. Some of these filters are discussed: the Kalman Filter (Section 2.4.1), the Extended Kalman Filter (Section 2.4.2), the Unscented Kalman Filter (Section 2.4.3), and the Particle Filter (Section 2.4.4). We observe how a Particle Filter and a set of Kalman Filters can be combined to create an MJPF (Section 2.4.6). This filter will be adopted in the methods proposed in this thesis and will be modified in order to work in combination with other models, such as VAEs.

Finally, Section 2.5 tackles the problem of dimensionality reduction. Indeed, probabilistic filters such as the MJPF cannot be directly applied on high dimensional data such as camera data. For this reason, the VAE (another generative model) and DVAEs are analyzed in Sections 2.6 and 2.7, respectively. A particular DVAE, the Kalman Variational Autoencoder, is also described in more detail in Section 2.7.1.

2.1 Generative Models

Two types of models can be distinguished in the Machine Learning context: *discriminative models* and *generative models* (GMs) [83]. Discriminative models learn the conditional probability of a class label given some observed input; this is the case of classifiers. On the other hand, generative models can learn the underlying hidden distribution of the data they are trained on and can generate samples from the same distribution.

Bayesian Networks (BNs) are a type of generative model that allow the factorization of the joint distribution of data through conditional probabilities depending on hidden variables. Dynamic Bayesian Networks (DBNs) perform this factorization across time [52].

When working on high dimensional data such as images, two of the most commonly used generative models are Variational Autoencoders (VAEs) [122] and Generative Adversarial Networks (GANs) [85]. VAEs learn the probability distribution of data *explicitly*, commonly assuming a Gaussian or Gaussian Mixture Model distribution. VAEs encode individual data samples by assuming a Gaussian distribution and representing each sample with both a mean and a variance. Through sampling from the learned distribution, they can generate similar data points to the ones they were trained on. Conversely, GANs implicitly learn the distribution of the data without defining any parameters. VAEs and GANs can additionally be used inside a DBN to learn the relation between high-dimensional data (e.g., images) and low-dimensional latent states. This relation corresponds to the so-called *observation model*.

2.2 Bayesian Networks

Probabilistic Graphical Models (PGMs) are graph-based representations encoding conditional dependences between random variables [127, 209].

Recent works [167, 227, 20, 9] have demonstrated the usefulness of PGMs for encoding semantic relationships between random variables. Additionally, relevant research contributions [21, 118, 198] suggest that PGMs are suitable representations for explaining and factorizing conditional dependences between latent states extracted from data through Deep Leaning techniques.

PGMs can be divided into two main classes: *Undirected Graphical Models (UGMs)* and *Directed Graphical Models (DGMs)*. UGMs, also known as Markov random fields, consider systems in which there is no directionality in the relationship between the variables [51]. Conversely, DGMs, also known as Bayesian Networks (BNs), allow us to model causal interactions between random variables. A BN is typically represented by two sets of parameters: the nodes in a BN, which represent random variables, and the directed links, which represent conditional probability distributions between the connected variables.

A simple example of BN is displayed in Fig. 2.1. Note how the arrows connect node z to node x and y. The arrows represent the direction of the causality relations. For example, being z the probability of having rain, and x and y being respectively the probabilities of the grass being wet and of the rain barrel being full, the causal relationships between z and x and between z and y are clear. z is called the *parent* and the conditional probabilities of x and y with respect to z can be written as $p(x|z)$ and $p(y|z)$. Note that, if we observe that the grass is wet (i.e., $x = True$), it becomes more likely that the rain barrel is full. However, when we know that it has rained (i.e., $z = True$), the wet grass does not alter our expectations regarding the rain barrel being full; similarly, a full rain barrel does not affect our expectations regarding the grass being wet. This means that x and y are conditionally independent of each other, given z. These conditional independence relationships can be written as $p(x|z,y) = p(x|z)$ and $p(y|z,x) = p(y|z)$. Each variable is, in fact, conditionally independent of its non-descendants, given its parents.

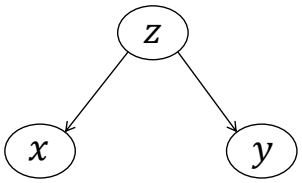


Fig. 2.1 Example of a simple Bayesian Network.

A BN represents a probability distribution over all the variables. Supposing a general case in which we have N random variables x_i, with $i = 1 \ldots N$, a generic entry in the distribution represents the probability of having a particular value for each variable, e.g., $p(x_1 = v_1, \ldots, x_N = v_N)$. The probability value of this entry can be found through the formula [192]:

$$p(x_1 = v_1, \ldots, x_N = v_N) = \prod_{i=1}^{N} p(v_i | parents(x_i)), \tag{2.1}$$

where $parents(x_i)$ denotes the specific values of the parent nodes of variable x_i.

In the case displayed in Fig. 2.1, this becomes $p(x,y,z) = p(z)p(x|z)p(y|z)$.

It should be noted how conditional independence relationships simplify the computation of the probability distribution over the graph, as they correspond to missing links. The graph is, therefore, not fully connected, as it would otherwise be, and the computations are fewer.

2.3 Dynamic Bayesian Networks

A particular type of BN, namely, a DBN [80], describes dynamic processes that incorporate changes through time. It is a Directed Acyclic Graph containing directed links between the involved variables, not allowing loop cycles.

DBNs can decompose data with complex and non-linear dynamics into segments that are explainable by simpler dynamical units. One of the main advantages of DBNs concerns their potential hierarchical nature: a DBN allows us to formulate in a simple way the causal relationships present at each time instant between high-level variables (capturing abstract semantic information) and low-level distributions (capturing rough sensory information). DBNs that have a hierarchical structure are called Hierarchical Dynamic Bayesian Networks (HDBNs). Accordingly, at each time instant, i.e., in a DBN's *slice*, causal relationships between variables are encoded through *inter-slice links*. Additionally, DBNs also model how random variables change over time; this information is encoded in *temporal links*. Fig. 2.2a and 2.2b show a DBN representation consisting of two and three hierarchical variable levels, respectively. These two DBNs will be referenced in the next sections, showing examples of how they can be implemented.

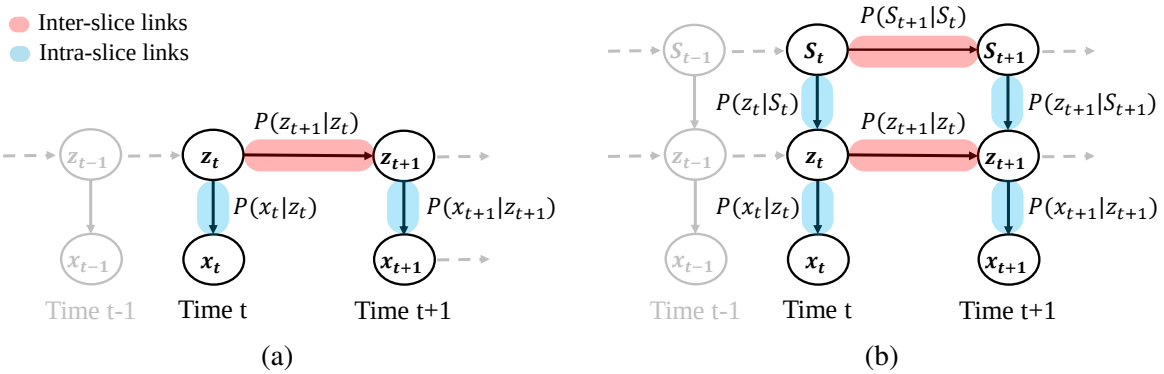


Fig. 2.2 A 2-level DBN (a) and a 3-level DBN (b). The links highlighted in red are inter-slice links, and the ones in blue are intra-slice links.

As described above for BNs, the nodes represent variables (e.g., x_t, z_t, S_t), and the links represent conditional dependences between them (e.g., $P(x_t|z_t)$, $P(z_t|S_t)$, $P(z_{t+1}|z_t)$). The

low-level variable (x_t) corresponds to sensory observation. In contrast, higher-level variables capture increasingly more conceptual information.

In this thesis, stationary DBNs are considered. The stationarity condition supposes that the same probabilistic relationships hold between each couple of subsequent time instants t and $t+1$ [186]. We can distinguish two types of links in the figure: inter-slice links and intra-slice links. Inter-slice links define relationships between variables at distinct time instants, whereas intra-slice links define relationships between variables at the same time instant.

In Fig. 2.2, we also display in lighter color the slice at time $t-1$. However, recall from Section 2.2 that, in these networks, variables are conditionally independent of their non-descendants given their parents. Consequently, taking the DBN in Fig. 2.2a as an example, the state z_{t+1} depends on the state at the previous time instant, i.e., z_t. It is conditionally independent of z_{t-1}, x_{t-1} (and of x_t), given z_t, i.e., the past and future are conditionally independent, given the present. This is called the *Markov property* (or *Markov condition*).

Learning a DBN consists of parameter learning and structure learning. The former is the process of learning the distributions for a certain structure, whereas the latter entails the identification of the optimal topology for a DBN, determining which variables should be included and how they are interconnected [142].

2.4 Probabilistic Inference

Now that the fundamental theory about DBNs has been covered, we can observe how DBNs can be used to represent the filtering process of the state of an agent. The state is a vector that could, for example, contain odometry information of the agent such as its position and speed, or a compressed representation of what the agent is seeing in the environment. In this thesis, Generalized States (GS) are considered, which will be introduced in Chapter 3. It is important to remark that the hidden state is not directly observable but is obtained from observations. When dealing with odometry data from a vehicle, the observations could contain position, velocity, acceleration, and higher-level derivatives. Tracking the state of an object means predicting how the state evolves and correcting this prediction through the newly obtained observations. This process can therefore be performed through two steps: *prediction* and *update*. In the prediction phase, the state z_t of an object at time t is estimated based on the transition probability $P(z_t|z_{t-1})$ and given all the observations until step $t-1$ ($X_{1\ldots t-1}$) [192]:

$$P(z_t|X_{1\ldots t-1}) = \int P(z_t|z_{t-1})P(z_{t-1}|X_{1\ldots t-1})dx_{t-1} \tag{2.2}$$

As it can be observed, a marginal is calculated over all the possible values of the state at time $t-1$.

When the new observation x_t is obtained, the update phase is performed, i.e., the prediction is corrected based on the observation [192]:

$$\begin{aligned} P(z_t|X_{1\ldots t}) &= P(z_t|X_{1\ldots t-1},x_t) && = && (2.3)\\ &= \alpha P(x_t|z_t,X_{1\ldots t-1})P(z_t|X_{1\ldots t-1}) && = && (2.4)\\ &= \alpha P(x_t|z_t)P(z_t|X_{1\ldots t-1}) && && (2.5) \end{aligned}$$

In 2.3, the evidence $X_{1\ldots t}$ is divided into two parts. In 2.4 Bayes law, $P(x|y) = \frac{P(y|x)P(x)}{P(y)}$, is applied. In 2.5 the Markov property is used. α is a a normalizing constant introduced so that the probabilities sum to 1.

At a time instant t, $P(z_t|X_{1\ldots t-1})$ is called the *prior*, $P(x_t|z_t)$ is the *likelihood* and $P(z_t|X_{1\ldots t})$ the *posterior*. The objective of Bayesian filtering is finding the posterior. At the next time instant, $t+1$, the posterior in t serves as the new prior.

Different types of filters can be employed for performing state estimation. The most commonly used ones are the Kalman Filter (KF), the Extended Kalman Filter (EKF), the Unscented Kalman Filter (UKF), and the Particle Filter. The KF is optimal in linear systems with Gaussian noise, whereas the other three filters are used when these conditions are not satisfied. In the methods proposed in this thesis, KF, UKF, and PF are employed. These filters are briefly introduced in the following sections. Particular attention is provided in describing them, as the corresponding formulas will be recalled again throughout the thesis, and in particular in Chapter 5. Conversely, the EKF is only briefly described.

It must be noticed that not only the state could be estimated: higher-level semantics could be considered too, as, for example, the movement of an object from a zone of the state space to another one. This type of estimation is consequently considered in Section 2.4.5, and a fundamental filter for this thesis, the Markov Jump Particle Filter (MJPF), is introduced.

2.4.1 Kalman Filter

The KF is a state estimation filter first introduced in 1960, and initially used for monitoring the position and velocity of an orbiting vehicle [87]. The Kalman Filter is optimal when two conditions are satisfied: *i)* the system is linear and *ii)* the noise is Gaussian.

If the model is linear and the noise Gaussian, we can define it through these two relations:

- The *observation model*, which connects at each time instant t the observed variable x_t with the state variable z_t through the use of a linear observation matrix H and a Gaussian zero-mean observation noise w_t with covariance R:

$$x_t = Hz_t + w_t \tag{2.6}$$

- The *transition model*, which relates the state vector z_{t-1} at time $t-1$ with the state vector z_t at time t:

$$z_t = Az_{t-1} + Bu_{t-1} + n_{t-1} \tag{2.7}$$

  where A is the linear transition matrix mapping the state at time $t-1$ onto the state at time t; B is a linear control input model mapping an optional control input u_{t-1} onto the state at time t; n_{t-1} is a Gaussian zero-mean prediction noise with covariance Q.

The prediction and update phases of the Kalman Filter are consequently performed as described in the following paragraphs. It must be noticed that not only the mean value of the state must be predicted and updated but also the covariance of it. A covariance estimation Σ is therefore defined too. This is because, under the considered assumptions, the variables in a Kalman Filter are Gaussian.

Prediction The mean and covariance of the state are predicted through the following equations:

$$z_{t|t-1} = Az_{t-1|t-1} + Bu_{t-1} \tag{2.8}$$

$$\Sigma_{t|t-1} = A\Sigma_{t-1|t-1}A^T + Q \tag{2.9}$$

Update The new observation z_t is obtained and the mean and covariance of the state are updated based on it. The estimation is updated by measuring the difference between the prediction and the observation. This difference is called *innovation* and is defined as:

$$v_t = x_t - H_t z_{t|t-1} \tag{2.10}$$

The innovation also has a covariance given by:

$$S_t = H\Sigma_{t|t-1}H^T + R \tag{2.11}$$

Combining the covariance of the innovation and of the prediction, a matrix called *Kalman Gain* is obtained:

$$K_t = \Sigma_{t|t-1}H^T S_t^{-1} \tag{2.12}$$

The Kalman Gain is used as a weight for the correction of the estimation. The values in the Kalman Filter are high when there is high confidence in the sensorial data (i.e., the observation) and low confidence in the prediction. This case leads to a high correction of the prediction according to the observation. In the opposite case, the correction is smaller. Therefore, the update is performed according to the following equations:

$$z_{t|t} = z_{t|t-1} + K_t(x_t - Hz_{t|t-1}) \tag{2.13}$$

$$\Sigma_{t|t} = \Sigma_{t|t-1} - K_t S_t K_t^T = [I - K_t H]\Sigma_{t|t-1} \tag{2.14}$$

A KF can be represented through a DBN like the one in Fig. 2.2a. The prediction phase corresponds to the link $P(z_{t+1}|z_t)$, whereas the observation phase corresponds to the link $P(x_t|z_t)$. The EKF, UKF and PF can also be represented with the same DBN.

It can be noticed that the Signal Processing community typically uses "x" to represent the latent state of a KF and "z" to represent the observation. However, "z" is also typically used to represent the latent state of a VAE [122]. Consequently, we choose this second notation to represent the continuous latent state of any DBN model, and adopted it throughout the entire thesis for consistency.

The Null Force Filter A particular type of KF is the *Null Force Filter* (NFF), also called *Unmotivated Kalman Filter* [104]. It is a Generalized Filter (a concept that will be better explored in Chapter 3), in which the state is composed of L variables and their derivatives. If we stop at the first-order time derivative, the state has dimension $2L$ (think, for example, of position and velocity). The NFF consists of a KF that has no control input matrix B (see Eq. 2.7) and has a transition matrix defined as:

$$A = [A_1 A_2] \tag{2.15}$$

with $A_1 = [I_L 0_{L,L}]^\intercal$ and $A_2 = [0_{L,L} 0_{L,L}]^\intercal$, where I_L is an identity matrix of dimension L; and being $0_{L,L}$ a matrix with L columns and L rows containing only zeros.

This filter, therefore, in the case of positional data, supposes that the object does not move and keeps having null speed.

As already observed, the KF is optimal when the observation and transition model are linear. When these hypothesis are not verified, the EKF, UKF and PF can provide better results.

2.4.2 Extended Kalman Filter

Let us consider the case in which the observation and prediction model are defined by the following equations:

$$x_k = h(z_t) + w_t \tag{2.16}$$

$$z_t = f(z_{t-1}) + n_{t-1} \tag{2.17}$$

where h and f are non-linear functions.

The solution provided by the EKF proposes to linearize the non-linear models around the last estimated value of the state and then to apply the normal KF steps [184]. This means that, at each time instant t, during prediction, Eq. 2.17 is linearized around $z_{t|t}$ and then the prediction step is performed to obtain $z_{t+1|t}$ and $\Sigma_{t+1|t}$; during update, Eq. 2.16 is linearized around $x_{t+1|t}$ and consequently the update step is performed to obtain $z_{t+1|t+1}$ and $\Sigma_{t+1|t+1}$. The EKF is not explored further, as the more recent UKF allows us to obtain better accuracy. We instead concentrate on this second filter.

2.4.3 Unscented Kalman Filter

The UKF was first proposed in [221] as an alternative and more accurate solution to the EKF. The idea behind the UKF is to select a set of carefully chosen sample points (called *sigma points*) of the state distribution that completely capture the mean and covariance of the estimated state, and to propagate them in the non-linear model. These propagated points capture the posterior mean and covariance to the 3rd order (of the Taylors series expansion).

If an L-dimensional state vector is considered, $2L+1$ sigma points are propagated, each associated with a weight W^i. Considering, for example, the prediction phase, if the estimated state $z_{t|t}$ at time t can be defined through a mean $\bar{z}_{t|t}$ and covariance $\Sigma_{t|t}$, the sigma points and the corresponding weights are defined by the following equations [221]:

$$\begin{aligned}
&z^0_{t|t} = \bar{z}_{t|t} \\
&z^i_{t|t} = \bar{z}_{t|t} + (\sqrt{(L+\lambda)\Sigma_{t|t}})_i && i = 1...L \\
&z^i_{t|t} = \bar{z}_{t|t} - (\sqrt{(L+\lambda)\Sigma_{t|t}})_{i-L} && i = L+1...2L \\
&W^{0,m} = \frac{\lambda}{L+\lambda} \\
&W^{0,c} = \frac{\lambda}{L+\lambda} + (1-\alpha^2+\beta) \\
&W^{i,m} = W^{i,c} = \frac{1}{2(L+\lambda)} && i = 1...2L
\end{aligned}$$

where $\lambda = \alpha^2(L+k) - L$ is a scaling parameter. Here, α determines the spread of the sigma points around $\bar{z}_{t|t}$ and is typically set to a small value (e.g., 1e-3). k is a secondary scaling parameter typically set to 0, while $\beta = 2$ for Gaussian distributions.

When these sigma points are propagated through the non-linear function f, the newly obtained points can be defined as:

$$z^i_{t+1|t} = f(z^i_{t|t}) \tag{2.18}$$

and can be used, together with the weights W^i, to approximate the mean and covariance of the predicted state:

$$\bar{z}_{t+1|t} = \sum_{i=0}^{2L} W^{i,m} z^i_{t+1|t} \tag{2.19}$$

$$\Sigma_{t+1|t} = \sum_{i=0}^{2L} \tilde{W}^{i,c} \{z^i_{t+1|t} - \bar{z}_{t+1|t}\}\{z^i_{t+1|t} - \bar{z}_{t+1|t}\}^\intercal \tag{2.20}$$

2.4.4 Particle Filter

We have seen that, in both EKF and UKF, the posterior probability density function was approximated through the two parameters of a Gaussian distribution, i.e., mean and covariance. Conversely, the PF is a non-parametric posterior estimator: the posterior is approximated by a set of particles, where each particle has a weight assigned to it [58].

If a PF with a number N of particles is defined, during the initialization of the estimation process, each particle is given a weight $\frac{1}{N}$. As new observations are obtained, the weights of each particle are recalculated (*reweighting*) based on the particle's likelihood given the observation. A high weight is assigned to particles that reflected more the actual data, while particles that performed a bad prediction are given a low weight. A *resampling* is then performed: particles having a low weight are eliminated, while particles with a high weight are replicated. After resampling, all the N particles have again a weight of $\frac{1}{N}$. This process is defined as sequential importance resampling (SIR). Resampling is executed to avoid particle degeneracy. This happens when, after some iterations, one particle weight takes a value close to one, whereas all the other particle weights are close to zero. However, frequent resampling - for example at each iteration - can lead to particle impoverishment: samples do not diversify and collapse into a single point in the state space. This can be avoided by resampling in a controlled way. A common way to determine if particle degeneracy is present consists in evaluating the effective sample size N_{eff}:

$$N_{eff} = \frac{1}{\sum_{i=1}^{N} (w^i)^2} \tag{2.21}$$

where w^i is the weight assigned to the i-th particle.

Resampling can then be performen when N_eff is below a defined threshold. This means that resampling occurs when a limited subset of particles, with their combined weights, disproportionately contributes to the overall total weight (see [58] for further details about the PF).

2.4.5 Switching Models

In the previous sections, we only considered state estimation filters with a single hidden state level. These filters can be represented with the DBN in Fig. 2.2a. However, an additional hierarchical level could be added: not only the instantaneous state of an object could be predicted but also additional symbolic information regarding the object itself. In the case of odometry data tracking, this could be, for example, information related to the areas where an object is moving and the type of motion that it is performing. Here, the state space can be clustered in areas having similar position values and velocity values, i.e., *clusters* or *superstates*. Therefore, it is possible to track across time how the object moves from one superstate to the next ones.

DBNs that allow considering more hierarchical levels are called *mixed-state* DBNs, and the corresponding learned models are known as *switching models*. These models are named like this because the prediction model at the state level changes when switching from one superstate to another one. Typically, two types of switching models are used: the *Markov Jump Linear Model* (also known as *Markov Jump Particle Filter*) [17] and the *Rao Blackwellized Particle Filter* [54]. The first one is employed for linear systems with Gaussian noise, the second one for non-linear systems with non-Gaussian noise. In this thesis, we focus on the MJPF.

2.4.6 Markov Jump Particle Filter

In this section, we briefly discuss the basics of an MJPF.

The MJPF synthesizes a three-level DBN as the one displayed in Fig. 2.2b : the evolution of the continuous state z_t related to a sensor observation x_t is tracked on the lower level, whereas discrete variables S_t are used to switch from one linear dynamical model for continuous state prediction to another one.

The relationship between the observation z_t and x_t is described by the observation equation of the KF (Eq. 2.6). The dynamical model, used to perform prediction at the continuous

level, is defined by the transition equation of the KF (Eq. 2.7) as:

$$z_{t+1} = Az_t + BU^{(S_t)} + n_t, \tag{2.22}$$

where Az_t takes the state-space information from z_t and makes its time derivatives null. $BU^{(S_t)}$ encodes the time derivative information (actions) of the agent at time t. $U^{(S_t)}$ depends on the variable S_t, which corresponds to the active discrete state at the time t. The variable n_t is a zero-mean Gaussian distribution representing the noise of the dynamical modeling.

During the online filtering phase, the MJPF uses a set of KFs at the continuous state level, which follow Eq. 2.6 (update phase) and Eq. 2.22 (prediction phase), and a PF at the discrete level.

Again, the same two phases observed for the other aforementioned filters mentioned above can be identified and performed at each time instant t:

- The *prediction* phase. During this phase, the filter estimates the state and superstate at the next instant.
- An *update* phase. The new measurement is used for updating the state and superstate prediction.

We will see more in detail how an MJPF can be built and used for filtering and anomaly detection in Chapter 3. Note that the MJPF allows switching between different dynamical models. It is, thus, a Switching Linear Dynamical System (SLDS).

2.5 Dimensionality Reduction

As mentioned in the Introduction, this work considers a multi-modal autonomous vehicle framework. This means that low-dimensional data (e.g., odometry data or steering wheel angle) are combined with high dimensional data (e.g., video frames from a camera mounted on the vehicle). In this thesis, this second type of information is mostly taken into consideration. Therefore, a dimensionality problem arises, as it is not possible to apply the MJPF directly on the single image pixels. Instead, it is necessary to transform the high-dimensional problem into a low-dimensional one by using an encoding algorithm. An approximation of the original image could then be obtained from this encoding step.

As seen in Section 2.1, generative models learn the probability distribution of the training data and generate new data with possible variations compared to the training ones. When dealing with images, two commonly used generative methods are Variarional Autoencoders (VAEs) [122] and Generative Adversarial Networks (GANs) [85]. Since VAEs, differently

from GANs, explicitly learn the data distribution, they can be more naturally inserted in a framework that uses DBNs.

2.6 Variational Autoencoder

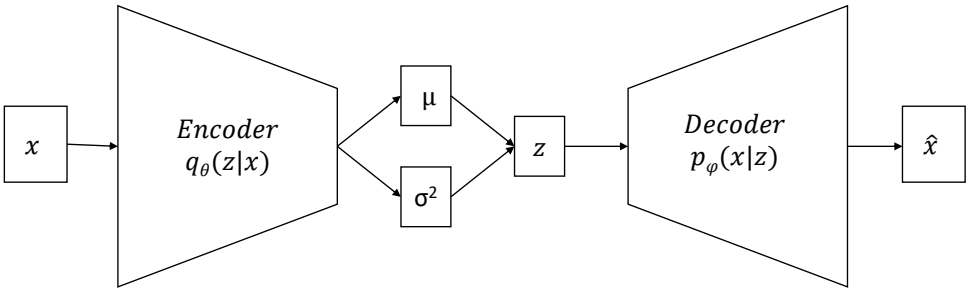


Fig. 2.3 General structure of the VAE.

In its vanilla version, a VAE is composed of an encoder $q_\theta(z|x)$ and a decoder $p_\phi(x|z)$, that can be estimated through the learning process of the VAE on training data. Through θ and ϕ, we define the parameters of the encoder and decoder, respectively. The encoder $q_\theta(z|x)$ allows representing each sample x fed into it through two bottleneck features, i.e., a mean μ and a variance σ^2. On the other hand, the decoder $p_\phi(x|z)$ synthesizes an observation x from a latent state z sampled from a Gaussian distribution $\mathcal{N}(\mu,\sigma^2)$ with mean μ and a variance σ^2. Therefore, the former allows the reduction of dimensionality of the observations (i.e., image data) and the application of anomaly detection at a low-dimensional state level; instead, the latter allows obtaining again high-dimensional data and applying anomaly detection at the observation level. Note that, in Fig. 2.2, the VAE can be seen as an observation model $P(x_t|z_t)$.

The learning phase of the VAE is conducted to optimize the parameters θ and ϕ, maximizing the sum of the lower bound on the marginal likelihood of each observation x of a dataset D, as described in [122, 123]:

$$\mathscr{L}_{\phi,\theta}(D) = \sum_{x \in D} \mathscr{L}_{\phi,\theta}(x), \tag{2.23}$$

where $\mathscr{L}_{\phi,\theta}(x)$ is defined as:

$$\mathscr{L}_{\phi,\theta}(x) = -D_{KL}(q_\theta(z|x)||p_\phi(z)) + E_{q_\theta(z|x)}[log p_\phi(x|z)], \tag{2.24}$$

where the term D_{KL} is the KL divergence. Therefore, the first term measures the difference between the encoder's distribution $q_\theta(z|x)$ and the prior $p_\phi(z)$; the prior typically being a standard normal distribution $\mathcal{N}(0,1)$. The second term is the expected log-likelihood of the observation x and forces the VAE to reconstruct the input data.

In this thesis, the VAE is used to encode the input information in a significant lower-dimensional space that exhibits probabilistic properties. As observed in Section 2.1, VAEs can be integrated inside a DBN model, when the observation model $P(x_t|z_t)$ has to be learned.

2.7 Dynamical Variational Autoencoders

VAEs were initially developed without including temporal correlation, meaning that each data vector and the corresponding latent vector at a certain time instant t were processed independently from the following time instants. Consequently, studies regarding VAEs were extended and applied to many complex Deep Dynamic Bayesian Networks (DDBNs) to model sequential data exhibiting temporal correlation. These DDBNs embedded in the VAE framework are Dynamical VAEs (DVAEs), that is, VAEs including a temporal model for sequential data. As described in [82], the DVAE model must contain two fundamental relationships:

- **Decoding link:** z_t is a parent of x_t . This link depicts the hierarchical nature of DVAE. This corresponds to what we called an *observation model*.
- **Temporal link:** at least one element in $Z_{1\ldots t-1}$ or in $X_{1\ldots t-1}$ is parent to either z_t or x_t. If the link is between z_{t-1} and z_t, it corresponds to what we called a *transition model*.

Various DVAE models have lately been proposed [71, 18, 40, 19, 133, 72, 139, 145]. Among them, the Kalman Variational Autoencoder (KVAE) by Fraccaro et al. [71] is adopted in Chapters 7 and 8 of this thesis.

There are several reasons for choosing the KVAE in our work instead of other methods. Firstly, the method in [110] was tested only on simple data and the training procedure was observed to be slow. Therefore, we opted to exclude this model. Secondly, methods such as [18, 40, 72, 139, 145] employ Recurrent Neural Networks (RNNs) or fully connected NNs to model the temporal link, whereas in Chapter 8 we desire a piece-wise linear model. Approaches adopting a combination of linear models, such as [71] and [19] can be modified to work as piece-wise linear models. The final choice of the KVAE was based on one further advantage that it presents: its disentanglement capability. The KVAE combines a VAE and a KF and separates the latent states in two levels of abstraction. The first level relates to the content of the images, whereas the second one relates to the dynamics between images,

thus proposing a disentanglement of the video information. This capability is in line with one of the characteristics of our framework that we will discuss in the next chapter, i.e., its interpretability. Therefore, as the KVAE is used in our method, Section 2.7.1 describes it.

2.7.1 Kalman Variational Autoencoder

The study in [71] examines the application of prediction models for linear video sequences and introduces the KVAE. It performs unsupervised learning to separate two different levels of abstraction. For each input frame x_t at time instant t, the following latent representations are defined: *i)* a continuous level a_t representing the content of the images, as output from the VAE and as input for a Bayesian Filter with linear Gaussian models; *ii)* another continuous level z_t representing the motion between consequent images. These lead to the DBN in Fig. 2.4.

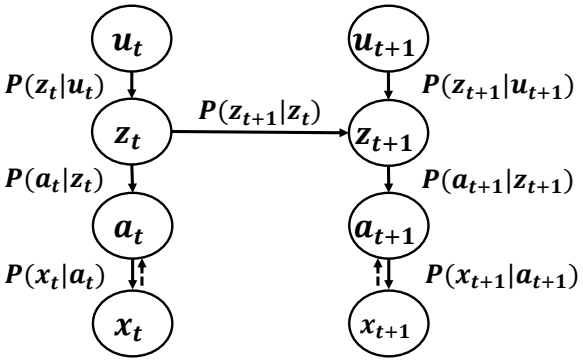


Fig. 2.4 The DBN corresponding to the KVAE proposed in [71].

The links between a_t and x_t are represented by the encoder $q_\theta(a_t|x_t)$ and by the decoder $p_\phi(x_t|a_t)$ of a VAE, respectively. The content-related state a_t is linked to the motion-related state z_t through the following pseudo-observation model:

$$a_t = \sum_{i=1}^{K} \alpha_t^{(i)} * C_i * z_t + v_t \tag{2.25}$$

whereas z_t at following time steps are connected through the dynamical model:

$$z_{t+1} = \sum_{i=1}^{K} \alpha_t^{(i)} * A_i * z_t + \sum_{i=1}^{K} \alpha_t^{(i)} B_i * u_t + \omega_t \tag{2.26}$$

The matrices $\{C_i\}_{i=1...K}$, $\{A_i\}_{i=1...K}$, $\{B_i\}_{i=1...K}$ represent a set of pseudo-observation models C, of transition models A, and a set of control matrices B, respectively, as defined in

the traditional Kalman Filter equations. Here, u_t represents an optional control input. In [71] and [117], for example, the case of a torque-controlled pendulum is considered, where the angle acceleration - proportional to the applied external torque - corresponds to the control input. Models A, B and C are combined through a vector of probabilities α_t.

Non-linearities in the latent space of image sequences are tackled by introducing an RNN with Long Short-Term Memory (LSTM) cells, that non-linearly updates network parameters over time, i.e., $d_t = LSTM(a_{t-1}, d_{t-1})$. The output of the LSTM, sent to a softmax function, generates the probabilities used to select the combinations of models A_i, B_i, C_i, i.e., $\alpha_t = softmax(d_t)$.

2.8 Conclusions

To conclude, in this chapter, we have analysed some concepts and algorithms that form the basis of this thesis.

Firstly, these concepts and algorithms are necessary to understand the State of the Art chapter, which will discuss self-awareness for Autonomous Systems. The framework adopted in this thesis, presented in Chapter 3, is based on DBNS, and on generative models in general. Moreover, in the self-awareness context, we will propose anomaly detection systems, that often adopt generative models such as VAEs. Consequently, generative models will be referenced throughout the other chapters of the thesis too. A representation based on DBNs is adopted, coherently with the framework underlying this thesis. Furthermore, VAEs are adopted, as this thesis mainly focuses on camera data, and VAEs allow performing dimensionality reduction while keeping a probabilistic representation. The KF, the UKF, MJPF, the VAE and the KVAE are employed, modified, and combined in the methods proposed from Chapter 5 to Chapter 8.

Chapter 3

State of the Art

This chapter describes the relevant state of the art in the context of this thesis. In Section 3.1, we begin by outlining the evolution of autonomous systems and their intersection with the concept of self-awareness. Then, we discuss the framework underpinning this thesis, elucidating its neurobiological inspiration and principal capabilities in Sections 3.2 and 3.3, respectively. In these two sections, we identify and discuss six basic capabilities and five characteristics that self-awareness frameworks should possess. As some of these capabilities and desiderata cover topics that have been tackled by many researchers, further details are provided in two sections: Sections 3.4 and 3.6 provide a literature review of anomaly detection and Visual-Based Localization (VBL), respectively; Section 3.5 defines what interpretability and explainability are.

After the objectives and capabilities of our analyzed framework have been introduced, the framework is described in deeper mathematical detail in Section 3.7.

Finally, in Section 3.8, we compare the adopted framework with other related ones, and elucidate the rationale behind the choice of this framework. The choice is based on the desiderata that we identified in preceding sections, and which either draw inspiration from the functioning of the human brain or are deemed advantageous for a self-awareness framework.

To summarize, this chapter provides a discussion of the framework at the base of this work, a description of some related systems, and a review of underlying topics such as anomaly detection, Visual-Based Localization, and interpretability.

Throughout this chapter, while presenting methods and discoveries from the state of the art, we also discuss extensively how they are connected to the adopted framework, and to the methods proposed in this thesis.

3.1 The Autonomous Vehicles field and self-awareness

Autonomous vehicles are vehicles designed to diminish or entirely eliminate the need for human intervention in the execution of their tasks. Fully autonomous cars are anticipated to yield numerous benefits, including a reduction in traffic incidents, alleviation of traffic congestion, and a decrease in emissions [238]. Additionally, they offer enhanced transportation options for individuals with limited mobility, for the elderly, and for people without a driving license [238]. Beyond transportation, many other applications are of relevance for autonomous vehicles, either terrestrial or aerial. Video surveillance and fault detection can benefit from the use of autonomous vehicles, which can monitor areas such as buildings or ships [32]. Among many other application examples, autonomous drones are adopted in agriculture to inspect the soil and crop growth, in Search and Rescue operations to localize survivors and deliver aid kits, in weather forecasting, and in traffic monitoring [59]. While this thesis and the underpinning framework are not limited to a specific application, the field of video surveillance is the most direct beneficiary of the methods described herein.

The field of autonomous cars has started to develop in the 1980s. Some important examples of these first self-driving research platforms were the NavLab vehicles developed by Carnegie Mellon University. NavLab 5 drove through America in a test, called the "No Hands Across America", in which the vehicle navigated autonomously 98 % of the time, while human operators controlled the throttle and brakes [6]. An important thrust to the field of Autonomous cars was provided by the three "Grand Challenges" organized between 2004 and 2007 by the U.S. Defense Advanced Research Projects Agency (DARPA). The cars competing in these challenges had to drive autonomously in real driving situations across 240 km, 212 km and 96 km, respectively. The first two challenges took place in non-urban areas, whereas the third one took place in a simulated urban area with other cars but with no pedestrians. No team completed the first challenge (2005), but five and six teams completed the second and third one, respectively [11]. These challenges and other ones (e.g., [60, 27, 31, 108, 28]) spur the development of autonomous cars [6]. Moreover, in the 21st century, other factors have accelerated AV research, such as the advancements in computer vision brought by the development of Deep Learning, and the adoption of new sensory modalities, like lidar [238]. In these last decades, drones have changed from being used for primarily military purposes to being adopted for commercial applications [205].

Amidst this growing interest in autonomous vehicles, researchers in this field have proposed two main approaches: computationalist and cognitive [156]. The former formulates fixed mathematical models to describe the behavior of the vehicles and objects around them, while the latter takes inspiration from human reasoning. The computationalist approach, however, has a weakness: developers cannot examine and mathematically formulate all

possible real-world situations that drivers may encounter. Cognitive approaches provide a solution to this problem, as they suggest that vehicles should continually learn through experience as humans do, which would allow them to progressively grasp rare and unexpected behaviors. The human brain has consequently been a source of inspiration for the development of artificial agents such as autonomous vehicles.

Some cognitive architectures are based on the concepts of awareness and self-awareness. Awareness relates to the knowledge that an agent, whether biological or artificial, has of the surrounding environment; in contrast, self-awareness is associated with the consciousness that the agent has of its own state. Self-awareness is a broad concept derived from neuroscience and several interpretations of it exist [56]. Lewis et al. state that "self-awareness is concerned with the availability, collection, and representation of knowledge about a system, by that system" [141], whereas Morin et al. defines self-awareness as the "capacity of becoming the object of one's own attention" [162]. In their book about self-aware computing systems [130], Kounev et al. define self-aware computing systems as systems with two main characteristics. Firstly, these systems "learn models capturing knowledge about themselves and their environment (such as their structure, design, state, possible actions, and runtime behavior) on an *ongoing basis*". Secondly, they "reason using the models (e.g., predict, analyze, consider, and plan) enabling them to act based on their knowledge and reasoning (e.g., explore, explain, report, suggest, self-adapt, or impact their environment) in accordance with higher-level goals, which may also be subject to change".

Following from these definitions, a self-aware agent focuses not only on the external environment but also on the evolution of its own state. For example, a person driving down a street is aware of her surroundings through her eyes and is also self-aware of how much force she exerts on the steering wheel when performing a curve, which allows her to adjust her movements and the impressed force based on what she sees. This concept can be applied to an artificial agent too, such as an autonomous car performing the same curving movement. The car must be equipped with a camera sensor to monitor its surroundings and a steering sensor to track its own state. Accordingly, two types of sensors can be distinguished: i) exteroceptive sensors to reach situational awareness, such as external cameras; ii) proprioceptive sensors to gain self-awareness, such as a steering wheel sensor in a vehicle. Through their situational and self-awareness capabilities, these agents can recognize the experience they are encountering from a set of learned experiences; this knowledge consequently forms the basis of conducting a decision-making step and perform an action, impressed on the environment through a set of actuators. The *ongoing* (or *continual*) learning of new models is also an important element of self-aware systems.

In their work [183], Regazzoni et al. proposed a cognitive framework based on the concepts of awareness and self-awareness. The work presented in this thesis is based on this framework.

It is worth mentioning that systems for AVs are typically divided into two main blocks: the perception system and the decision-making system. The former has the objective of using the sensory input to create an internal representation of the environment (and of the vehicle's state), whereas the latter is responsible for guiding the vehicle to its final goal [11]. Whereas the full framework includes both modules, this thesis only proposes methods for the first one.

3.2 The neurobiological inspiration behind the adopted self-awareness framework

Since this thesis is based on the cognitive self-awareness framework by Regazzoni et al. [183], relevant neurobiological theories and concepts that worked as inspiration are discussed in this section. First, we describe the idea of the brain as a "predictive machine" (Section 3.2.1). Then, taking into consideration this idea, we comment on some concepts from the work of neuroscientist Karl Friston that are relevant to this thesis (Section 3.2.2). Finally, we explore some further discoveries of neuroscience that work as important indicators on how to develop a bio-inspired artificial agent (Section 3.2.3).

3.2.1 The brain as a probabilistic prediction machine

From the second half of the last century, the idea of the brain as a "predictive machine" has taken hold in the neuroscience field. According to this idea, perception is not merely the reading of sensory input, but is the combination of these sensory inputs with the brain's expectations and beliefs about their causes [200]. As a consequence, the brain is a machine that continuously makes predictions about the evolution of the state of the world and of itself, and matches these predictions with the corresponding observations. What humans feel is the brain's "best guess" of the causes of the received sensory inputs [200].

Although it took hold in neuroscience only in the last decades, the idea of perception as an indirect reflection of the real state of the world dates back millennia in its first glimmers, as it can be found in Plato's Allegory of the Cave [177]. Physicist and physiologist Hermann von Helmholtz, in the late nineteenth century, defines perception as "unconscious inference" [171] and psychologist Richard Gregory, in the 1970s, as a "neural hypothesis-testing" [86]. The term "hypothesis testing" summarizes the idea described in the previous paragraph, that

is, the brain makes hypotheses (predictions) about the world and tests them against sensory inputs (observations).

Many neuroscientists have analyzed this idea, performed experiments with results that supported it, and then proposed their own take on it [200, 173, 98]. For example, Anil Seth, in his thought-provoking book "Being You", defines perception as a "controlled hallucination" [200]. As humans never actually experience sensory inputs themselves, but only interpretations of them, human experience is determined by top-down prediction and not by the bottom-up sensory input. The term "hallucination" would bring to mind the case of someone seeing or hearing something that is not real, like in schizophrenia. Seth's definition, however, refers to a "controlled" hallucination, because it is corrected through the comparison with (bottom-up) sensory input from the world [200].

Among the many neuroscience studies, in this thesis, we are particularly interested in the work of Karl Friston.

3.2.2 The free energy principle and other relevant concepts from the work of Karl Friston

The free energy minimization concept was proposed by neuroscientist Karl Friston and is related to the idea of the brain as a "predictive machine". To explain this concept, let us start by noting that the second law of thermodynamics states that the entropy of every isolated system increases over time. Friston affirms that to counter the pull of this law, living beings have a tendency to occupy states in which they are expected to be. In this case, the states might be, for example, values of heart rate, temperature, or other bodily characteristics [200]. Through this seeking of statistically expected states, living beings can minimize their sensory entropy. A living organism experiences an "unexpected" or "surprising" state when it finds itself in a condition with a low probability of occurrence [98]. Indeed, a quantity called "surprisal" specifies how statistically unexpected an event is [200].

Notice that, however, entropy and surprisal cannot be directly measured because their calculation would include analytically intractable integrals. Consequently, the concept of Variational Free Energy (VFE) is introduced, which can be approximated with a prediction error [74], i.e., the difference between what the model expects and what is observed. Prediction error is always greater than surprisal [98]:

$$Prediction\ error = surprisal + perceptual\ divergence$$

The perceptual divergence compares the current hypothesis with the true posterior beliefs. It is a Kullback Leibler divergence and, therefore, it is always greater or equal to zero.

Therefore, minimizing prediction error means minimizing surprisal. The prediction error is, therefore, an upper bound to surprisal [173]. VFE (or prediction error) can be minimized in two ways: either by finding the best explanation for sensory data, or by acting to change the observed sensory data. So, VFE can be minimized by perception or action [173]. Starting from Section 3.3, we will observe how prediction errors are relevant in the adopted framework.

Another concept used by Friston and important for this work concerns Generalized States (GSs). GSs $\tilde{z} = [z, z', z'', ...]$ are composed of states per se, by their velocities, accelerations, etc. [77].

The states that are revisited and compatible with survival are called "global random attractors" [74, 41]. The motion among global attractors can be defined through flow functions describing the evolution of the GSs [77]. The flow can be decomposed in two components: a curl-free or irrotational component that moves towards regions with high ergodic density, and an orthogonal divergence-free or solenoidal component. Ergodic density is a probability measure that can be interpreted as the probability of finding the living being in a certain state at a random point in time [75, 172].

3.2.3 Other indications from neuroscience

The work of Friston and other neuroscientists [173, 98, 200] provides suggestions for the development of an artificial agent. With "agent", in artificial intelligence, one usually describes "anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators" [192]. In our case, the agent also obtains proprioceptive observations.

From indications from neuroscience, an agent that mimics the way humans interpret the world should be: *1)* Bayesian (specifically, it should perform Bayesian inference), *2)* hierarchical, *3)* multi-sensorial, and *4)* data-driven. We examine each of these points in the following paragraphs.

As we have observed in the previous sections, the predictive brain performs inference about the state of the world and itself. We have analyzed *Bayesian* inference in Section 2.4. Notice that both human perception and sensor recordings are always accompanied by uncertainty. For example, when humans observe entities moving at a far distance, there will be uncertainty in the estimation of their identity. Humans could be uncertain in classifying a small urban animal as a dog or a cat, or in ascertaining if someone is an acquaintance or just a lookalike. Similarly, when working with vehicular camera data, we might have to consider the uncertainty in the appearance of objects due to particular viewing angles and lighting sources. When using GPS for tracking the movement of an object, as an example,

there is uncertainty due to measurement errors and interpolation errors. We must, therefore, acknowledge that we are working with random variables, and Bayesian inference provides a solution for this. However, there is also another reason behind the choice of using Bayesian inference: multiple sources of evidence suggest that the brain operates as a Bayesian system [98].

Secondly, we observed that the method should be *hierarchical*. Adopting a hierarchy allows an agent to process information from low-level observations to high-level concepts, similarly to how human beings do. Friston notices how hierarchical models might be used by the brain too, given the hierarchical arrangement of cortical sensory areas [74]. Furthermore, this hierarchical division also allows handling prediction at different time scales: complex low-level information can be accurately predicted only for short time windows, whereas high-level conceptual information can potentially be predicted in a longer time window with higher accuracy [98].

Thirdly, we remark on the importance of the method being *multi-sensorial*. We noticed in Section 3.1 how an intelligent vehicle can potentially be endowed with multiple sensors. Similarly, human beings use their senses to navigate and interact with the world. Note that these senses are not only limited to the obvious exteroceptive ones (sight, hearing, taste, touch, and smell), but also include interoceptive ones, which report, for example, heartbeats, blood pressure levels, various low-level aspects of blood chemistry, degrees of gastric tension and how breathing is progressing [200]. This relates again to the division between exteroceptive and interoceptive sensors in autonomous vehicles mentioned in Section 3.1. In this thesis, we combine two sensors of the vehicle: an exteroceptive sensor (camera) and a sensor that could either be considered interoceptive or exteroceptive (GPS). Nonetheless, the proposed framework suggests the integration of a larger number of sensors, including both exteroceptive (e.g., lidar) and interoceptive ones (e.g., IMU).

Finally, notice that *data-driven* learning is opposed to model-based learning [70]: in data-driven approaches, the process of learning predictions or controls is guided by data; in model-driven approaches, a model of the system has to be first derived through expert knowledge. Data-driven learning better resembles human learning by experience.

We also highlight the importance of having a *generative* model. We already discussed generative models in Section 2.1. Here, instead, we observe how the brain is a generative model: using an example by Seth [200], if I hold a tomato in my hand and only see the front of it, I am perceptually aware that the tomato also has a back, even if I cannot see it, because of the model of how a tomato is structured that I have learned in my head. Generative models are what allows the brain to perform the predictions that will be compared to observations. Notice, however, that we did not include this concept in the above list. This is because,

Bayesian inference (point 1) already requires a generative model [173], and so the concept was already implicitly included in the list.

In addition to the four above-mentioned characteristics, we will add an additional one, which will be discussed in the next section.

It is worth noting that the framework by Regazzoni et al. [183] - which underlies this thesis - also takes inspiration from the work of other neuroscientists except the aforementioned ones, such as Haykin [94] and Damasio [45]. However, their theories are not discussed here as they are not central to the methods proposed in this thesis.

3.3 Basic Capabilities and Characteristics of the Adopted self-awareness Framework

After having introduced neurobiological concepts and elucidated four essential characteristics inspired by them, we now turn our attention to discussing the requisite capabilities of a self-awareness framework and introduce one additional desired characteristic.

As proposed in [183], six basic capabilities should be possessed by an agent for it to be considered self-aware: *initialization*, *memorization*, *inference*, *anomaly detection*, *model creation*, and *interface with control*. We briefly analyze them and illustrate them in Fig. 3.1. The self-aware agent is first initialized with a predefined model (*initialization*), and can memorize the model (*memorization*); it consequently uses the learned model to perform inferences about its future state and the future changes in the environment (*inference*); it identifies new situations when they appear (*anomaly detection*) and learns a new model (or sub-part of the overall model) to describe them (*model creation*); finally, it uses the acquired knowledge to perform decisions and implement changes in the environment (*interface with control*).

As can be seen in Fig. 3.1, the capabilities are applied in sequence and form a closed loop, as new model creation reconnects itself with memorization. The learning of new models allows performing *incremental learning* (also called *continual learning*). Note that, in this thesis, we only consider the capabilities from initialization to anomaly detection.

Observe how the inference and anomaly detection capabilities relate to the concept of the brain as a "predictive machine": inference defines the prediction phase, whereas anomaly detection represents the comparison of the prediction with the sensory observation. If prediction and sensory observation diverge, high anomalies are detected. Anomalies indicate the necessity to create a new model, since the current one no longer accurately represents the incoming data. Anomalies, therefore, are connected with the concept of surprisal. When a

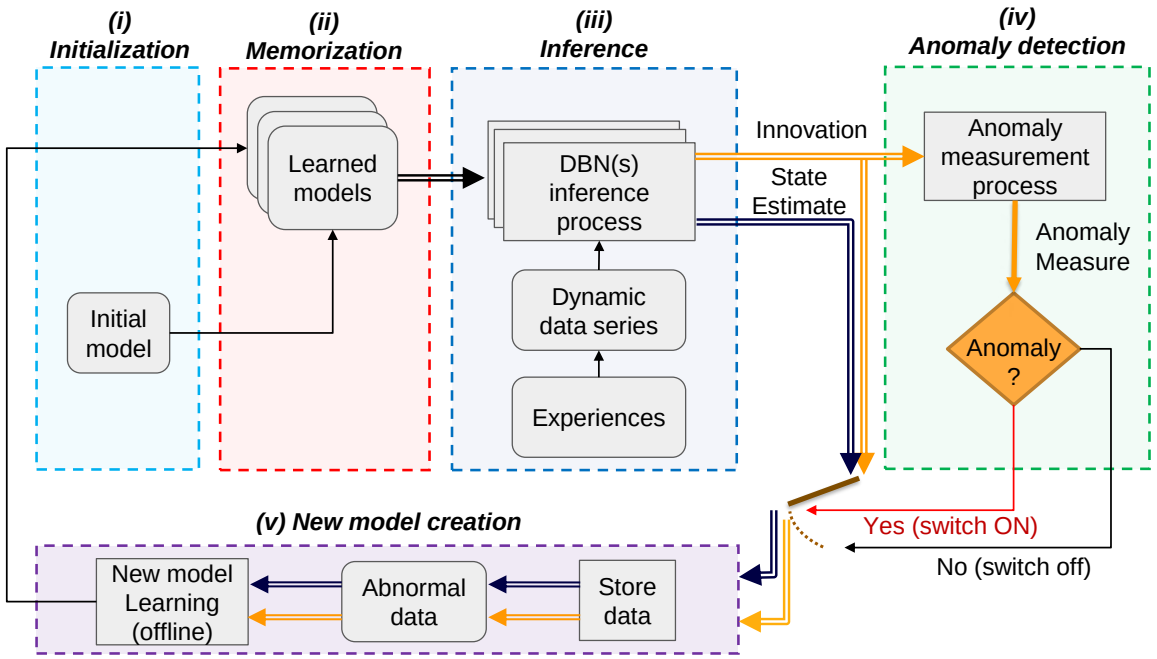


Fig. 3.1 The first five basic self-awareness capabilities and the continual learning loop that they form.

new model is created, the anomalies in the next analogous experiences are minimized, along with prediction errors, surprisal and free energy, aligning with the ideas of Friston presented in Section 3.2.2.

In this context, we introduce the final desired characteristic, which complements the four identified earlier: *interpretability* (or, at the very least, explainability). Interpretable models are inherently comprehensible, as we can elucidate their inner workings. A detailed explanation of this concept is provided in Section 3.5. First, notice, how, in the passage from anomaly detection to model creation, interpretability is important: it can be used to identify the origin of the detected anomaly and which part of the overall model should be created anew. Furthermore, also notice that interpretability is not a bio-inspired characteristic, as the human brain is not easily interpretable.

The subsequent sections delve deeper into three key areas within our adopted framework: anomaly detection (one of the six capabilities), interpretability (one of the five desiderata), and localization (an additional capability deriving from the multi-sensoriality property).

Recognizing the extensive body of research on anomaly detection and its significance as the fourth capability among the six discussed above, we dedicate Section 3.7.4 to a literature review of anomaly detection methods. Given the centrality of anomaly detection in this thesis, we devote considerable space to its exploration.

In Section 3.5, we define the concepts of interpretability and explainability, drawing connections to anomaly detection.

Finally, Section 3.6 presents a literature review of VBL methods. Our earlier discussions highlighted the importance of multi-sensoriality as a desideratum for self-awareness frameworks. From this desideratum derives the possibility of correlating different sensor values among each other, and of predicting the values of one sensor from the others. In this thesis, we study the visual and odometric sensors. When the positional information is predicted from the visual information, we are referring to VBL.

3.4 Anomaly Detection

Anomaly Detection is not only relevant for self-awareness. The identification of abnormal instances of data represents a relevant issue across different research topics. Through anomaly detection algorithms, for example, video surveillance cameras can recognize when possibly dangerous or violent events are happening, such as break-ins, assaults, armed robberies, or traffic incidents; suspicious zones in medical images like mammograms or tomography scans can be identified and can help doctors diagnose tumors or nodules; fraudulent credit card transactions can be detected as deviations from the normal usage profile of the clients. In general, anomaly detection methods constitute a necessary component in all applications that require identifying a change from a known set of rules or models. It is indeed fundamental to note that anomalies are related to a model and are not absolute. In the video surveillance case, the initial model could describe the normal interactions between customers in a store; in the medical imaging case, it could extract the salient features of a healthy organ; in the credit card case, it could keep track of user's normal credit card transaction patterns. Abnormal data correspond to a divergence from these learned models and are instead subject to different rules. Therefore, it is evident how anomaly detection, which has many applications, represents a fundamental step also for artificial self-aware systems that can continually learn from new situations, as it allows a system to distinguish the occurrence of a new situation from an already experienced one.

3.4.1 Types of Anomalies and Anomaly Detection Methods

The concept of anomalies is dependent on data type, on the cardinality of the relationship between the involved variables, data structure, and data distribution [69]. Different anomaly types - and consequently anomaly distances and anomaly detection algorithms - can be identified when working on time-independent or time-dependent applications, on low-dimensional

data such as trajectories, high-dimensional data such as images or structures such as graphs, etc.. To estimate the complexity of anomaly detection, we reference the work by Foorthuis [69], which, through a review of anomaly detection literature, recognizes three broad groups of anomaly, 9 basic types, and 63 subtypes. Considering as an example video data, Ramachandra et al. [179] distinguish five main types of anomaly: *i)* appearance anomaly, *ii)* short-term motion anomaly, *iii)* long-term motion anomaly, *iv)* group anomaly, *v)* time-of-day anomaly. As an example, a car driving along a street could either see an abnormal object such as a traffic cone at the center of the street (*i*), detect an abnormal motion when the car in front suddenly brakes (*ii*), perform a sequence of abnormal movements, for example due to the driver falling asleep (*iii*), observe two other vehicles interacting abnormally, as in a collision (*iv*). Finally, motions and speeds deemed normal during the day may not be permitted after sunset (*v*).

The work by Chandola et al. [33] serves as a foundational reference for the classification of anomaly detection methods. In this section, we report a summary of the defined anomaly classes for point anomalies, i.e., abnormalities calculated on a single datapoint. The authors distinguish six general methods for point anomaly detection; the first five are non-probabilistic methods, while the sixth one groups together the probabilistic methods. In the following paragraphs, we give a brief discussion of five out of six of these methods. Additionally, Chandola et al. [33] notice how additional methods can be distinguished for contextual anomalies such as abnormal sequences in a time series.

Classification based methods (CLA). In these methods, a classifier is learned that can either distinguish normal data from abnormal ones based on a discriminative boundary (one-class classifier) or different classes of normal data from each other (multi-class classifier). In the second case, an instance is detected as abnormal when it can not be associated with enough confidence to any of the normal classes. Any algorithm typically used for classification can be adopted for these methods, including neural networks, Bayesian Networks, Support Vector Machines (SVMs) [42], and rule-based classification methods.

Distance based methods (DB). These methods are based on the calculation of the distance of data points from their closest neighbors, leveraging the assumptions that normal data occur in dense neighborhoods, whereas anomalies are far from their closest neighbors. The choice of the distance metric is therefore essential. To account for neighborhoods of various densities, the distance value is often relative to the distances between points in the closest neighborhood.

Clustering based methods (CLU). These methods rely on assumptions dependent on the chosen clustering techniques: *i)* techniques like Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [63], RObust Clustering using links (ROCK) [88] or Shared Nearest Neighbor clustering (SNN) [62] assume that anomalies are not assigned to any of the built clusters; *ii)* when using techniques like Self-Organizing Map (SOM) [125], one can assume that abnormal data lie far away from the cluster's centroid; *iii)* finally, one could leverage the assumption that normal data belong to big clusters and abnormal ones to small clusters.

Information theoretic methods (IT). These methods use measures from Information Theory such as Kolmogorov Complexity, entropy or relative entropy to analyze the information content of the data and extract the most anomalous subset of data points (i.e., sequence, subgraph, image area) compared to the rest of the data.

Statistic/probabilistic methods (STA). Probabilistic-based methods aim to model the normal sample features' distributions (inliers) associated with the training data, whereas abnormalities can be detected as data samples known as outliers with low probability [185]. The probability that a data sample belongs to a certain distribution can be estimated using probabilistic distances such as Bhattacharyya, Mahalanobis, Hellinger, or the Kullback–Leibler divergence. Abnormality detection using probabilistic (statistical) models can be classified into two main groups: parametric such as Gaussian Mixture Model (GMM) methods and Regression methods and non-parametric such as Histogram or Kernel Density Estimation (KDE) methods [222]. A significant advantage of the probabilistic approach is that it can be easily generalized to different data types and modalities. Its limitation arises when attempting to fit the data to a specific distribution, as the distribution may not be a suitable match for the data [2].

Chandola et al. [33] propose this classification and further identify an additional set of anomaly methods, i.e., spectral anomaly detection techniques, based on the assumption that data, especially high dimensional ones, can be brought to a lower dimensional subspace where normal and abnormal data can be more easily distinguished. Examples of such methods could be using Principal Component Analysis (PCA) or VAE on image data or embedding graphs as adjacency matrices. These methods can be used as a preprocessing step applied before the other methods defined above. Similarly, when handling abnormal sequences of data, such as in time-series data, detection can be either performed through an adaptation of the previous methods to contextual anomalies or through methods that learn the structure

of the data, for example executing the prediction of the following time instants from the previous ones (*prediction-based methods (PRE)*).

Table I summarizes the classification of anomaly detection methods, indicating a set of characteristics that an anomaly detection method should have in our context and framework, that should not be confused with the five desiderata described earlier. However, the seven characteristics in the table correlate to the previous desiderata and capabilities. If data labeling is unnecessary, data-driven unsupervised learning can potentially be performed. The ability to provide anomaly scores with a confidence range and its applicability to probability models make the method suitable for use in DBN models and Bayesian inference. Providing anomalies on diverse abstraction levels is compatible with a hierarchical architecture. The capability to perform incremental learning aligns with the continual learning loop in Fig. 3.1. Finally, when elaborating video data, as is the case in this thesis, we need methods that can be applicable to high-dimensional time-series data.

Notice how all the proposed methods can be applied to time-series data and are unsupervised. Most of these methods could also be applied for some variety of incremental learning through the use of anomaly scores, but this problem is rarely tackled in the state of the art. Classification does not give an anomaly score with a confidence range, as in general in this class of method a sample is only described as belonging or not to the class; information-theoretic methods also do not have this capability. Only prediction-based methods and statistical methods (and, in part, clustering-based methods) can be applied to probability models. Finally, the methods introduced in this thesis possess all the specified characteristics and, in addition, provide anomaly scores at various levels of abstraction. Further discussion of this table from the point of view of our framework is provided in Section 3.7.1.

Table I Comparisons between the different anomaly detection methods. *: unnecessary for one-class version, necessary for multi-class version; **: valid for some methods of this category but not all.

	CLA	DB	CLU	IT	STA	PRE	Ours
Data labeling unnecessary	*	✓	✓	✓	✓	✓	✓
Gives anomaly score with confidence range	✗**	✓	✓	✗	✓	✓	✓
Extendable to HD data	✓	✓	✓	✓	✓	✓	✓
Applicable to probability models	✗	✗	✗**	✗	✓	✓	✓
Applicable to time-series data	✓	✓	✓	✓	✓	✓	✓
Allows incremental learning	✓	✓	✓	✗	✓	✓	✓
Gives anomaly on diverse abstraction levels	✗	✗	✗	✗	✗	✗	✓

3.4.2 Anomaly Detection in Low-Dimensional Data

In this thesis, we focus on unsupervised probabilistic methods based on DBN representations for time-series data. Several works have leveraged BNs and DBNs to perform anomaly detection in various fields [158, 29, 195]. This type of network, allowing a hierarchical probabilistic representation of the world, is particularly appropriate for autonomous systems that try to reflect human reasoning. BNs additionally allow the building of a hierarchy of anomalies. We will further comment on how DBNs satisfy the five desiderata of self-aware frameworks in Section 3.7.

In particular, the MJPF has been used as a basic Bayesian filter applied on DBNs to perform anomaly detection on low-dimensional data (such as odometry data in [17] and control information in [115]) from different types of sensors of multi-modal physical agents like semi-autonomous vehicles. Section 3.7 tackles the framework adopted in these papers (and in this thesis).

3.4.3 Anomaly Detection in High-Dimensional Data

As video data are high-dimensional, it cannot be elaborated pixel-wise in a time-efficient way by the majority of anomaly detection algorithms developed for low-dimensional data - including MJPF. Instead, features first need to be extracted from the video data, allowing the problem to be reformulated in a lower dimensionality [38]. The process of feature extraction can either be accomplished through *hand-crafted methods* or through *Deep Learning (DL) based methods*. In the first case, human knowledge about specific problems such as occlusion, variations in orientation, scale, or illumination is exploited to develop feature extractors. Two examples of hand-crafted feature methods, out of many, are the Histogram of Oriented Gradients (HOG) [43] and the Histogram of Optical Flow (HOF) [44]. In the case of DL-based methods, training data are given as input to an NN, which learns the most salient features for the task it needs to accomplish. Various studies on computer vision applications [7, 4, 169] demonstrate how features learned by NNs generally outperform hand-crafted features. For this reason, in recent years hand-crafted feature methods have largely been taken over by DL-based methods. Because of this, in this section, we only consider DL-based methods.

Once dimensionality reduction is performed, many anomaly detection methods described in Section 3.4.1 can be performed on the extracted features, as summarized in Table I. Additionally, Kiran et al. [124] identify three main strategies for video representation and anomaly detection, i.e., methods based on *reconstruction*, methods based on *prediction* and methods based on *generative models (GMs)*. It is worth noting how these strategies

can be combined: for example, prediction [152, 143] and reconstruction [204] methods often rely also on generative models. As seen already in Section 3.4.1, important types of anomaly detection methods are *probabilistic* methods. For high-dimensional data, some further remarks should be made, given the widespread use of probabilistic generative models like GANs and VAEs in the state of the art

Reconstruction-based methods The idea behind the baseline reconstruction-based method for video data is simple: a Convolutional Autoencoder (CAE) is trained to reconstruct either the image frames or some connected characteristics (e.g., the OF calculated between following images). Testing images (or corresponding characteristics) are then given as input to the trained CAE. Consequently, abnormal inputs tend to display a worse reconstruction than normal ones. Often, the reconstruction error is only used to train the CAE to obtain a lower-dimensional representation of the input, but it is not used during testing. In the testing phase, instead, anomaly detection is performed using the low-level features extracted through the CAE encoder. For example, in [196], a distance-based Euclidean anomaly score is calculated in the low-dimensional space related to image frames, and in [217] the extracted motion features from the OF between consecutive images is given as input to a one-class Support Vector Machine.

Prediction-based methods As noted by different research works [216, 149], reconstruction-based methods present some limitations. Since NNs have a high generalization capability when enough training data are given, abnormal frames can be reconstructed with a limited reconstruction error and, therefore, classified as normal. Possible alternatives to reconstruction-based methods are prediction-based methods, that calculate abnormalities comparing the difference between a predicted future frame and its ground truth. These methods are the high-dimensional version of the homonym prediction-based methods mentioned in Section 3.4.1.

Several works employ feed-forward architectures to make predictions of future time instants given the current/previous ones. When video data are considered, Convolutional Neural Networks (CNNs) are used inside these methods, as they allow the extraction of spatial features of the video frames. However, one main problem arises when using vanilla CNNs for video forecasting: although they can capture intra-frame dependencies in the receptive field given by the used kernel size, they cannot represent inter-frame dependencies and long-range spatial dependencies [170]. This problem has been tackled in several works and has been mitigated using dilated convolutions for capturing long-range dependencies [236]

and 3D convolutions to account also for modeling across time and for capturing inter-frame dependencies [245, 246].

Most current prediction methods perform forecasting by using RNNs - especially LSTM networks - and transformers [219]. RNNs keep an internal state that works as a compact memory of past observations that is updated when a new observation is obtained. When forecasting is performed on video data, Convolutional LSTMs (ConvLSTMs) [201] are often used, which combine CNNs and LSTMs. ConvLSTMs have become extremely popular for anomaly detection purposes [159, 153, 226].

Generative models based methods GMs can learn the underlying distribution of the training data. VAEs [122] and GANs [85] are models of this type typically used when working on images. Whereas GANs implicitly learn the distribution of the data, VAEs perform it explicitly. Vanilla VAEs assume a Gaussian distribution of the data and encode each sample through a mean and variance; they are used in many probabilistic approaches for anomaly detection.

Probabilistic approaches As mentioned previously, anomaly detection approaches may have a deterministic or probabilistic nature. Most approaches present in the current literature are deterministic, which constitutes a problem, especially in prediction-based methods. If different next frames can derive from the current one, a deterministic approach trained with a pixel-wise loss function will learn to average all possible outcomes, resulting in a blurry prediction. To obtain a better performance, recent methods have started to consider the uncertainty that is intrinsic to the world. When working with images, the uncertainty can be in the appearance of objects due to particular viewing angles and lighting sources, or in the sequence between frames, as different possible next frames can be generated from the current one (e.g., a car at an intersection can either turn left or right). VAEs constitute the basis of many probability-based methods. In [5], an uncertainty estimation is performed in the different layers of the VAE's decoder to detect novel observations. In [233], a recurrent convolutional VAE is developed, and an anomaly score is calculated from the reconstruction error probability.

This capability of VAEs and GANs aligns with the core principles of this thesis, as it focuses its attention on the development of a method that is probabilistic and that can consequently be inserted inside a DBN framework for capturing anomalies and exploiting them to perform the continual learning of models in time-series data. By reducing the dimensionality of the observed frames, VAEs and GANs can be used as non-linear observation models in a DBN such as the one described in Fig. 2.2b.

Additionally, as VAEs offer the possibility of modeling video as low-dimensional random variables, recent research has also tried to combine VAEs with DBNs [225, 110, 19, 71]. These works however did not consider anomaly detection and continual learning of models as an objective. Additionally, experiments were conducted on very simple datasets displaying only basic motions and observations from a static viewpoint.

Another work [180], instead, coupled GANs and DBNs considering anomaly detection and continual learning at the same time. In this work, a hierarchy of coupled cross-modal GANs is developed. When a high anomaly is produced due to the detection of abnormal observations or abnormal dynamics, a new GAN is built for learning the new situation. A Switching model is developed, consisting of the hierarchy of GANs at the continuous level and of an HMM at the discrete level. However, because GANs inherently learn data probability without explicit parameterization, this approach has limitations and does not contemplate the possibility of using some of the anomaly distances typically adopted for statistic anomaly detection methods. We will analyze some of these anomalies in Section 3.7.4.

In this thesis, first in Chapter 5, a method for anomaly detection in video data is presented; it employs a generative model reconstruction-based method for extracting video features. Anomaly detection is then performed through a predictive method. The algorithm combines VAEs and DBNs to obtain a probabilistic representation of the observed current image and multiple possible predictions of the next image.

3.5 Interpretability and explainability

When considering anomaly detection applied in particular to the self-aware agents' field, another important concept can be introduced, i.e., *interpretability*. As we have seen in Section 3.3, it is one of the five desiderata that we identified for self-aware systems. It can be observed that there is no strict, universally recognized definition of interpretability, which is often also associated with explainability. We refer in this thesis to the definition provided by Rudin [191], who divides between inherently interpretable models, which "provide their own explanations, which are faithful to what the model actually computes", and post-hoc explanation models, in which a second model is "created to explain the first black-box model"; the former models are the objective of interpretable Machine Learning (ML), whereas the latter ones of explainable ML. Similarly, in [148], interpretability is defined as answering the question "How does the model work?", and explainability as answering the question "What else can the model tell me?". It is interesting to note the desiderata and properties of interpretable research defined by [148], which include causality, decomposability of the

individual parameters, and algorithmic transparency. Bayesian models, such as DBNs, are interpretable algorithms.

However, since there is no strict definition of the two concepts of "explainability" and "interpretability", many practitioners in the ML field use them interchangeably.

Explainable anomaly detection is a field that has recently gained attention [146, 101]. When performing anomaly detection, it is desirable to determine where or when an abnormal event was, together with its cause, e.g., what model rules were broken. The reason can be field-specific. For example, in healthcare, anomaly detection algorithms are used to detect diseases and an explanation of the anomaly can help the doctor make a diagnosis. This is even truer in the field of Autonomous Vehicles, as unexplainable autonomous cars are less easily accepted by the public [50]. In the framework adopted in this thesis, a further reason to use interpretable models such as DBNs can be traced back to the six capabilities described in Section 3.3. As we observed, once anomalies are detected, a new model must be trained. However, sometimes not the entire model needs to be retrained, but only a part of it. Understanding the anomaly reason (or, simply, at what abstraction level it occurs) allows us to detect which part of the models must be retrained. This concept will be explored further in the methodology sections.

3.6 Multi-Sensorial Agents and Visual-Based Localization

As we observed in Sections 3.2.3 and 3.3, we consider a multi-sensorial framework. In a data-driven, multi-sensorial framework, it is possible to learn how the observations from the different sensors are related to each other and, consequently, to predict the observations of one sensor from the values of another one. When we derive absolute positional information from video data, we are in the domain of VBL. This field is of great relevance for Autonomous Systems because, to navigate in an environment, they need the capability of localizing themselves in it.

Therefore, in Chapter 8, we will tackle the challenge of localizing a moving agent in a *known* environment, given only camera data. By defining the environment as "known", we hypothesize to be provided with sequential synchronous visual and odometric data during the learning phase of the models. We can consequently correlate our framework to the fields of *VBL*, *Visual Odometry (VO)*, and *Visual Simultaneous Localization and Mapping (V-SLAM)*.

VBL consists in the task of retrieving a camera's pose given an image or a sequence of images from the camera itself [175]. The survey in [175] proposes a division of VBL methods into two main groups: *indirect/retrieval-based methods* and *direct methods*. *Indirect methods* map images of the training set in a chosen feature space, creating a database of descriptors

for already seen images; when a query image is provided during the online execution of the method, the algorithm looks for the best match in the dataset. *Direct methods* directly recover the pose of the query according to a known reference. Pose regression approaches are one sub-type of this second category. CNNs can be used to regress the pose from images; the first example of a CNN-based regression architecture for VBL is PoseNet [119]. In PoseNet, the classification layers of a GoogleNet [211] were substituted with two fully connected layers to regress the camera pose. Pose is thus regressed in an end-to-end way. In [207], several retrieval-based and regression-based architectures are tested on a real dataset of a supermarket; some of the tested methods adopting a CNN use the Posenet architecture. An approach based on classification is considered too.

Other DL architectures have also been exploited in the VBL framework in combination with CNNs, such as spatial Long Short-Term Memory network to capture contextual information [220]. Most VBL methods rely on a single image to estimate position and orientation. Whereas image sequences are not always available, they can increase accuracy when they are accessible. A set of works [138, 24, 218, 231] have consequently begun to leverage image sequentiality inside the newly proposed frameworks. In our method, we similarly assume the presence of sequential data, which enables us to track the moving agent.

It is worth noting that, except for the aforementioned ones, other subcategories of VBL methods fall under the direct and indirect groups. For example, direct approaches also include feature to points matching methods, which find the correspondence between camera image features and points in a 3D point cloud model of the environment [175]. However, these methods incorporate or derive components (e.g., the 3D point clouds) that fall outside the scope of interest for this thesis.

In contrast, the objective of Visual Odometry (*VO*) is to estimate the *relative motion* of a camera using a sequence of images [235]. Consequently, VO is susceptible to estimation drift over time. It is important to note that VO is typically employed in scenarios where the environment map is not available, differently from VBL and from our proposed objective. In recent years, end-to-end methods using CNNs have also been proposed in the VO field, such as in [223, 109, 150].

Finally, in *V-SLAM*, an agent localizes itself in an *unknown* environment and simultaneously builds a map of it [235].

This thesis, in Chapter 8, describes a method that combines some characteristics of the three general methods discussed above for the specific case of video surveillance localization in a fixed environment. The method incorporates the following features:

- Known Environment. Similar to retrieval-based and regression-based VBL methods, the environment is known. This means that both video and positional data are available during training. It is a limitation compared to V-SLAM, which builds a map online.
- Absolute Positioning. Our method, like VBL, aims to estimate the absolute positioning.
- Sequential Visual Data. As in VO and V-SLAM, sequential visual data are leveraged, achieving a more robust localization than the one obtained by single-image VBL.
- Generative Map. During training, we build a generative map of the environment, which is not continuous but sparse, as it is composed of a set of clusters. The clustering enables a hierarchical representation and higher explainability. Each cluster allows us to predict positions and their changes during a navigation task. Explainability in this case regards the capability of the model to correlate video anomalies with the dynamic behaviors of the agent.

Improving localization with anomaly detection

As in Chapter 8, we propose a method that leverages anomaly detection to improve localization, here we briefly review this topic.

[174] and [229] use a monocular inertial SLAM, combined with wheel odometry and Global Navigation Satellite System (GNSS) measurements, respectively. In the former paper, when abnormal chassis movement is detected, the wheel odometer is removed from the localization estimation; in the latter, the same is performed when GNSS data are abnormal.

In [174], a monocular inertial SLAM is proposed with three methods for detecting abnormal chassis movement; when anomaly is detected, the wheel odometer is removed from the state estimation equation and only camera and IMU (Inertial Measurement Unit) are adopted. An inertial SLAM framework is also employed in [229], this time together with GNSS measurements; the chi squared test is used to detect anomalies on the GNSS data and to potentially exclude them from the localization estimation.

Anomaly detection and localization have also been combined when using other sensors instead of a camera, e.g., acoustic data in [230], and Ultra-WideBand receivers in [57]. In all these methods, multiple sensors or measurements are provided at test time, and anomaly detection is adopted to decide which localization to trust more. Conversely, in this thesis, only one sensor measure is available at test time. Moreover, we will analyze how to leverage anomalies to partially restart the localization process.

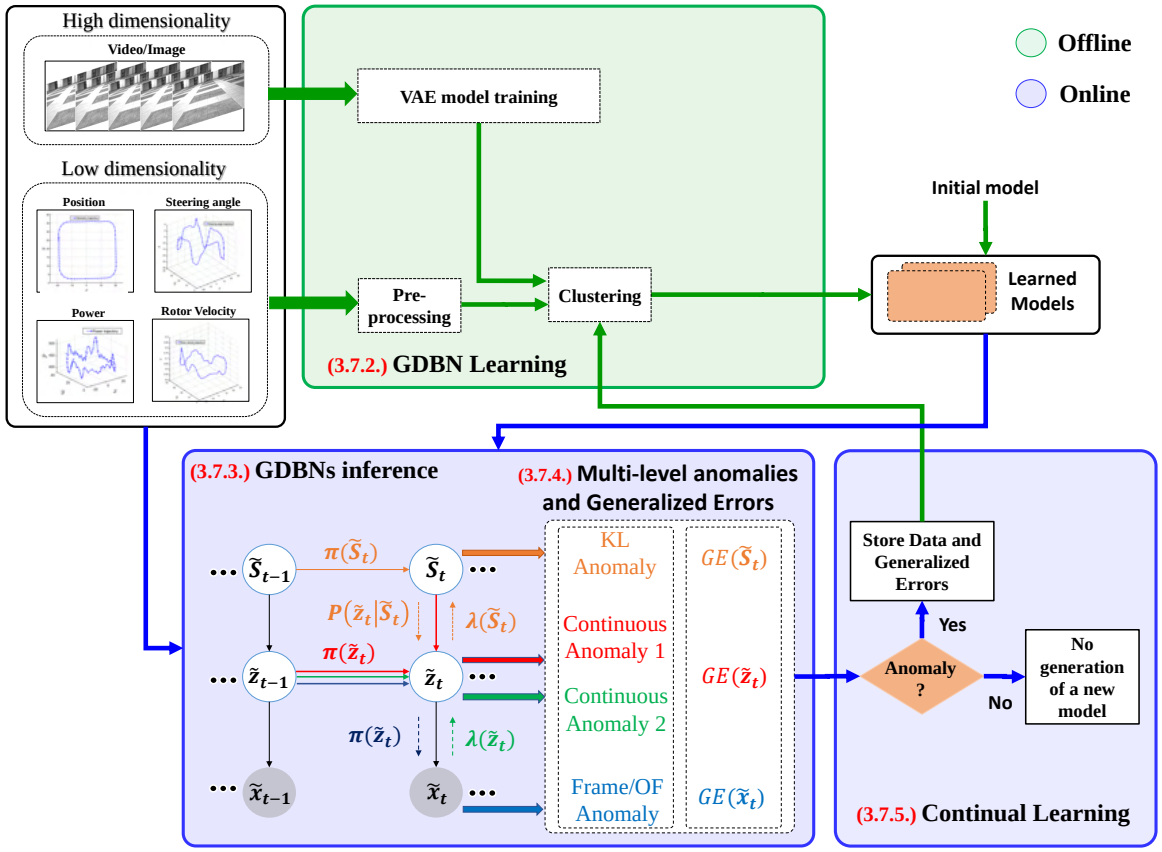


Fig. 3.2 Block diagram of the proposed self-awareness method. © 2021 Elsevier, http://www.elsevier.com.

3.7 Details of the adopted self-awareness framework

Having comprehensively introduced the essential information to understand the proposed framework — from the foundational algorithms, neurobiological inspirations, desired capabilities and characteristics, to the state-of-the-art approaches — we can proceed to a detailed mathematical description of the adopted framework.

3.7.1 General Framework description

The proposed framework is illustrated in Fig. 3.2. Low and high-dimensional data can be given as input to the framework. After executing preprocessing on low-dimensional data and feature extraction through the VAEs' encoder on high-dimensional data, a Generalized State vector is built, following from Friston's definition (see Section 3.2.2). It includes the filtered states as Generalized Errors (GEs) obtained from the initial model, which is activated during

the real-time inference in the initial iteration. In this case, the initial model represents static rules of the surrounding environment (i.e. the expected dynamic changes are quasi null and are only affected by random perturbations). Performing clustering on the Generalized Errors, a Generalized Dynamic Bayesian Network (GDBN) model is learned. It is a DBN in which the hidden states are Generalized States (GSs). When new data are obtained, the learned model is used to perform inference through the GDBN, and different levels of anomalies are extracted. If a high level of anomaly is detected, the corresponding Generalized Errors are stored, and a new model is built to be used in parallel with the previous ones in the next GDBN inference.

From Fig. 3.2 we can also deduce the characteristics in Table I:

- First of all, our method does not need any data labeling. Instead, it is unsupervised: a model is learned and the anomaly is detected with respect to it.
- Both low and high-dimensional data can be used as input. In Fig. 3.2, odometry, steering angle, power, and rotor velocity data from a vehicle are shown as examples of low-dimensional data; video data from a camera as high-dimensional data.
- Time-series data are considered.
- A probability model in the form of a GDBN is learned.
- Anomalies with respect to this model are detected at different abstraction levels, specifically at the frame level, continuous level, and discrete level.
- The detected anomalies are coupled with a confidence range, as will be evident in Section 3.7.4.
- When an anomaly is detected with respect to a learned model, a new model is created, allowing incremental learning and an expansion of previous knowledge.

In the following Sections, a more detailed description of the framework is provided. The training of the GDBN model is defined in Section 3.7.2. Sections 3.7.3 and 3.7.4 present the MJPF testing algorithm and its multi-level anomaly measurements, respectively. Then, in Section 3.7.5, the use of Generalized Errors obtained through the DBN anomalies is set as a basis for the continual learning of new models. The index of the section where each concept is explained can also be found in red in Fig. 3.2 near its graphic representation. Finally, we dedicate Section 3.7.6 to describing how this framework complies with the five identified desiderata and to showing how this was possible through the use of DBNs.

3.7.2 Generalized Dynamic Bayesian Network (GDBN) model

Initially, a self-aware agent starts perceiving the surrounding environment using an initial GDBN (i.e., a null force filter with static assumptions about the environmental states) by interpreting the received generalized observations $\tilde{x}_t = [x_t \ \dot{x}_t]^\intercal$ that comprise the variable and its generalized coordinates of motion coming from the agent's sensors. In this case, the agent detects abnormalities all the time and calculates the Generalized Errors $\tilde{z}_t$ expressed as:

$$\tilde{z}_t = [z_t \ \dot{z}_t]^\intercal \tag{3.1}$$

where z_t are the predicted hidden states obtained by the null force filter according to the following dynamic model:

$$z_t = z_{t-1} + v_t \tag{3.2}$$

while $\dot{z}_t$ are errors on derivatives calculated as innovations by the null force filter using current generalized observations as follows:

$$\dot{z}_t = \frac{x_t - H z_{t-1}}{\Delta t}, \tag{3.3}$$

where H is an identity matrix and Δt is the sampling time.

Consequently, the GEs collected in previous experience are used during the training phase as input to an unsupervised clustering algorithm (e.g. Growing Neural Gas (GNG) [78] or Self-Organizing Maps (SOMs) [126]) which encodes the GEs into discrete components producing a set of discrete variables ($\tilde{\mathbf{S}}$) or neurons which we refer to as superstates, such that:

$$\tilde{\mathbf{S}} = \{\tilde{S}_1, \tilde{S}_2, \ldots, \tilde{S}_M\} \tag{3.4}$$

where M is the total number of superstates.

Note how the equations considered above refer to the low-dimensional data case, in which the observation model is known, is linear, and is expressed by the matrix H that maps observations to hidden states. In the case of high-dimensional data, instead, we must first bring data to a low-dimensionality through the use of a feature extractor. Therefore, the first part of the GDBN that must be learned is the observation model itself. As observed in Section 3.4.3, this can be performed through the training of networks such as VAEs and GANs. In this thesis, we consider the use of VAEs for such a task. In this case, the observation model, that in the low-dimensional case was represented by Eq. 3.3, becomes:

$$\tilde{x}_t = h(\tilde{z}_t) + v_t, \tag{3.5}$$

where the function $h(.)$ corresponds to the non-linear transformation applied by the VAE's decoder $P_\phi(\tilde{x}|\tilde{z})$, and v_t is the noise.

Throughout the methodology chapters, two main methods are proposed to join VAEs and DBNs. In Chapter 5, we propose an approach that takes video frames and their OF as input; in Chapter 7 and the following ones, instead, we adopt a method solely based on video frames that uses a KVAE. In both cases, though, the concept is to have some information related to both the frame content and the motion between the frames. In this general description of the framework, we adopt the notation and the DBN of the method in Chapter 5. However, the same concepts can be applied to the method proposed in Chapter 7 and onwards.

In Chapter 5, to capture both information of frame content and of motion between subsequent frames, we train a couple of VAEs, one to reconstruct the frame at time t, x_t, and the other to reconstruct the Optical Flow (OF) between frames at time t and $t+1$, OF_t. The bottleneck is consequently divided into two parts: one, that we call z_t, representing the state/content (from x_t); the other, that we call $\dot{z}_t$, capturing the motion/velocity information (from OF_t). Clustering is performed on the combination of these two variables.

Once the dimensionality reduction process is performed, the rest of the method remains mostly the same, with some adjustments and limitations.

After clustering, the $M \times M$ transition matrix (Π) defined as:

$$\Pi = \begin{bmatrix} \pi(\tilde{S}_t = \tilde{S}_1) \\ \vdots \\ \pi(\tilde{S}_t = \tilde{S}_M) \end{bmatrix} = \begin{bmatrix} \pi_{11} & \dots & \pi_{1M} \\ \vdots & \ddots & \vdots \\ \pi_{M1} & \dots & \pi_{MM} \end{bmatrix} \tag{3.6}$$

is learned by estimating the transition probabilities $\pi_{i,j} = P(\tilde{S}_t = j|\tilde{S}_{t-1} = i), j,i \in \tilde{\mathbf{S}}$ over a period of time. Thus, it is possible to create a set of generalized superstates $\tilde{S}_t$.

This set of discrete variables realizes the top or discrete level of the GDBN. Each generalized superstate $\tilde{S}_t$ ($\tilde{S}_t \in \tilde{\mathbf{S}}$) is associated with statistical properties as mean value ($M^{(\tilde{S}_t)}$), covariance matrix ($Q^{(\tilde{S}_t)}$) and a set of hidden GSs $\tilde{z}_t$ encoded inside it. The hidden continuous GSs ($\tilde{z}_t$) represent the intermediate or continuous level of the GDBN. The relationship among the hidden states and superstates is characterized by the $P(\tilde{z}_t|\tilde{S}_t)$ link in the GDBN.

The bottom level in the GDBN stands for the real generalized observations ($\tilde{x}_t$) measured by the sensors. The path from $\tilde{S}_t$ to $\tilde{x}_t$ realizes the chain of causality among random variables at hierarchical levels. The probabilistic dependencies among the variables involved in the chain of causality are characterized by the linked edges. Considering in this case the chain rule, the joint probability of $\tilde{S}_t$, $\tilde{z}_t$ and $\tilde{x}_t$; $P(\tilde{S}_t, \tilde{z}_t, \tilde{x}_t)$, can be factorized as a product of the

conditional probabilities such as:

$$P(\tilde{S}_t, \tilde{z}_t, \tilde{x}_t) = P(\tilde{S}_t)P(\tilde{z}_t|\tilde{S}_t)P(\tilde{x}_t|\tilde{z}_t) \tag{3.7}$$

This implies the possibility of using the model to generate new data samples given the direct cause (i.e. the parent node). In addition, links between generalized hidden variables at consecutive time instants represent the corresponding conditional temporal probabilities. Π embeds the dynamic rules at the discrete level that drive the dynamic changes by switching between multiple dynamic models at the continuous level. Such probabilistic causal reasoning in the GDBN allows us to explain events, diagnose causes, and make predictions about future events. The transition probability $P(\tilde{S}_t|\tilde{S}_{t-1})$ at the discrete level is represented by the transition matrix Π. Let us suppose that, at time $t-1$, particle n is associated to cluster i, i.e., $\tilde{S}^n_{t-1} = i$. In this case, the probability $P(\tilde{S}^n_t|\tilde{S}^n_{t-1})$ can be calculated using the i-th row of the transition matrix Π:

$$P(\tilde{S}^n_t|\tilde{S}^n_{t-1} = i) = [\pi_{i1}, \pi_{i2}, ..., \pi_{iM}] \tag{3.8}$$

It is worth noting that, optionally, multiple transition matrices can be employed instead of only one. A set of Temporal Transition Matrices (TTMs) $\Pi^{(g)}$ could be computed, containing the probabilities $\pi^{(g)}_{ij}$ of moving from one cluster to the other ones, given that g time instants have been spent in the current cluster. This element of the vocabulary is optional, as the transition matrix Π could be used alone. However, performing the prediction at the discrete level with a combination of Π and of the TTMs can provide a more accurate prediction of the next cluster, because also the time spent in a cluster is taken into consideration.

In this case, let us consider that the time spent in the current cluster g corresponds to τ time instants. The probability $P(\tilde{S}^n_t|\tilde{S}^n_{t-1})$ can now be calculated as:

$$P(\tilde{S}^n_t|\tilde{S}^n_{t-1} = i, g = \tau) = [\pi_{i1}, \pi_{i2}, ..., \pi_{iM}] + \alpha[\pi^{(\tau)}_{i1}, \pi^{(\tau)}_{i2}, ..., \pi^{(\tau)}_{iM}] + \beta^{(\tau)}\delta \tag{3.9}$$

where α balances the two matrices and can be set for example to 0.5. The TTM allows us to consider how much time has been spent in the cluster; keeping the Π matrix too avoids overfitting to the TTM, which is built with less data. Additionally, δ is an element that adds a uniform probability of moving to all the clusters. This element avoids the possibility of remaining stuck in a cluster. Its importance can be incremented the more time is spent in a cluster, through the scalar $\beta^{(g)}$, which increases with g.

The dynamic causal model describing the state-space representation of a time-series process - under the assumption that each observation $\tilde{x}_t$ stems from a d-dimensional hidden

state $\tilde{z}_t$, itself generated by a discrete hidden superstate $\tilde{S}_t$ - takes the following form:

$$\tilde{z}_t = g(\tilde{z}_{t-1}, \tilde{S}_{t-1}) + w_t = A\tilde{z}_{t-1} + BU^{(\tilde{S}_{t-1})} + w_{t-1} \tag{3.10}$$

The continuous function $g(.)$ in Eq. (3.10) is assumed to be linear and determines the state temporal evolution at the continuous level guided by the discrete level predictions and it is subject to a Gaussian noise w_t. In Eq. (3.10), A and B are the dynamic model matrix and the control model matrix, respectively. A different control vector $U^{(\tilde{S}_{t-1})}$ is associated with each superstate $\tilde{S}_{t-1}$ that depends on the mean derivative of $\tilde{z}_{t-1}$ samples encoded in that superstate. So, Π encodes not only the transitions between superstates but also jumps between different linear models at the continuous level. Note that, in the case of high-dimensional data, the prediction model can change. Either linear or non-linear models can be employed. In the method in Chapter 5 non-linear models are used, whereas in the one from Chapter 7 linear models are adopted. In the case of Chapter 7, we use a model as the one in Eq. 3.10 (however, we will observe that a further DBN level is added). Instead, in Chapter 5, to keep the structure of the high-dimensional case as similar to the low-dimensional one as possible, we opt to learn for each found superstate a neural network $N^{(\tilde{S})}$ characterizing the temporal evolution of the GS for that superstate. Eq. 3.10 can therefore be substituted with the following expression:

$$\tilde{z}_t = g(\tilde{z}_{t-1}, \tilde{S}_{t-1}) + w_t = N^{(\tilde{S}_t)}(\tilde{z}_{t-1}) + w_{t-1} \tag{3.11}$$

where w_{t-1} is the error after convergence of the network and can be approximated as a Gaussian noise.

Fig. 3.3 summarizes the proposed GDBN. In red, the intra-frame connections are linked to their corresponding element: $P(\tilde{x}_{t-1}|\tilde{z}_{t-1})$ is the observation model and corresponds to matrix H in the low-dimensional case and to the VAE's decoder $P_\phi(\tilde{x}|\tilde{z})$ in the high-dimensional case; the link $P(\tilde{z}_{t-1}|\tilde{S}_{t-1})$ is correlated with the cluster mean $M^{(\tilde{S}_t)}$ and covariance $Q^{(\tilde{S}_t)}$. In green, the inter-frame connections are displayed: the prediction model $P(\tilde{z}_t|\tilde{z}_{t-1})$ is chosen through the local motion $U^{(\tilde{S}_t)}$ in the low-dimensional case and through the neural network $N^{(\tilde{S}_t)}$ in the high dimensional case; $P(\tilde{S}_t|\tilde{S}_{t-1})$ is encoded by the transition matrix Π (and, optionally, by the TTMs $\Pi^{(g)}$). All the learned elements of clusters ($M^{(\tilde{S}_t)}$, $Q^{(\tilde{S}_t)}$, Π, $\Pi^{(g)}$, $N^{(\tilde{S}_t)}$) constitute what we denominate as the clusters' vocabulary.

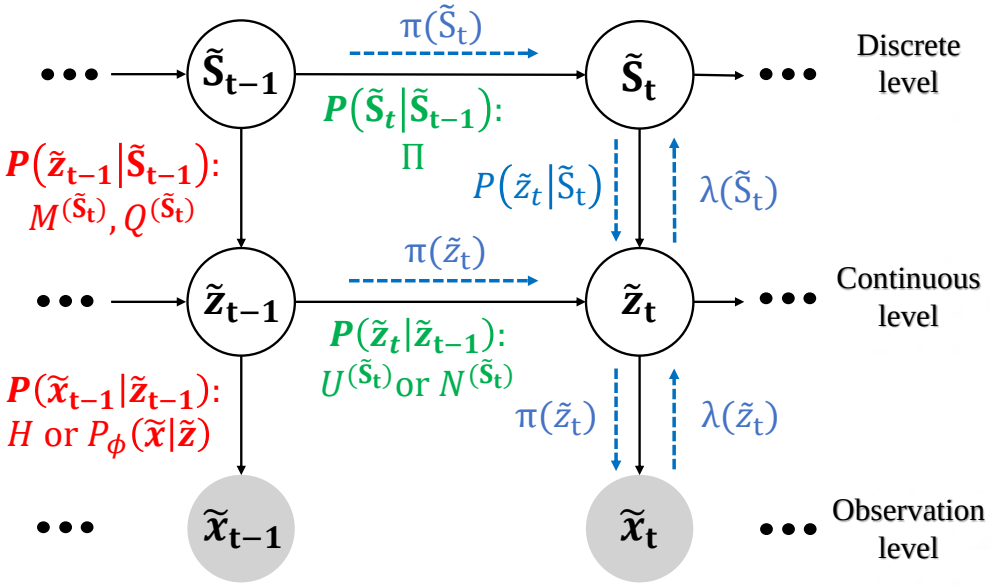


Fig. 3.3 The proposed Generalized Dynamic Bayesian Network. © 2021 Elsevier, http://www.elsevier.com.

3.7.3 Real-time Inference Algorithm

The MJPF first introduced in [17] (see Section 2.4.6) is employed during the real-time process to perform inferences at different hierarchical levels starting from the learned GDBN model. MJPF realizes a switching model that uses a combination of a PF to predict discrete superstates and a bank of KFs for continuous states' prediction and estimation. In the case of GSs extracted from high-dimensional data, in which we apply a non-linear model to perform state prediction, a bank of UKFs [221] can be used instead, as we will do in Chapter 5. Such a switching behavior between the dynamic transitions at discrete/continuous levels and observations permits updating of belief in hidden variables by passing local messages in simultaneous inference modes, namely the predictive or causal inference (top-down) and the diagnostic inference (bottom-up). This capability of DBNs is called *message passing*. Observe how it recalls the concepts seen when discussing the hypothesis of the brain as a "predictive machine" (Section 3.2.1).

Temporal predictive messages $(\pi(\tilde{S}_t), \pi(\tilde{z}_t))$ (refer to Fig. 3.3) depend on the dynamic rules stored in the model and are called intra-slice messages. Instead, top-down messages from $\tilde{S}_t$ to $\tilde{z}_t$ depend on the clustering statistics including mean values and covariance matrices. The bottom-up reasoning is based on likelihood models consisting of messages $(\lambda(\tilde{S}_t), \lambda(\tilde{z}_t))$ (refer to Fig. 3.3) passing in a feed-backward manner to adjust expectations given a sequence of observations. The PF relies on the transition matrix encoded in the dynamic model as a

proposal distribution to predict future superstates ($\tilde{S}_t^n$). Initially, it draws N samples from that distribution. These samples are equally weighted and associated with a specific superstate. Then, for each particle n, a KF is employed to predict the continuous variables $\tilde{z}_t$ which depend on the expected superstate as pointed out in Eq. (3.10).

Once the new evidence $\tilde{x}_t$ is observed, a message is backward-propagated from the bottom level towards the higher levels. Consequently, the weights of the particles can be reweighted according to:

$$W_t = W_t \lambda(\tilde{S}_t) \tag{3.12}$$

In Eq. (3.12), W_t is the vector of particles' weights $[w_t^1, w_t^2, ..., w_t^N]$. Instead, $\lambda(\tilde{S}_t)$ is the diagnostic message propagated from the bottom towards the top level of the hierarchy to update the belief in hidden variables at that level and can be calculated as:

$$\lambda(\tilde{S}_t) = \lambda(\tilde{z}_t) P(\tilde{z}_t|\tilde{S}_t) = P(\tilde{x}_t|\tilde{z}_t) P(\tilde{z}_t|\tilde{S}_t) \tag{3.13}$$

where $\lambda(\tilde{z}_t)$ is the message backpropagated from the evidence $\tilde{x}_t$ to the intermediate level $\tilde{z}_t$. $P(\tilde{z}_t|\tilde{S}_t)$ denotes a set of N Gaussian distributions with means $M^{(\tilde{S})}$ and covariances $Q^{(\tilde{S})}$. Instead, $\lambda(\tilde{z}_t) \sim \mathcal{N}(\tilde{z}_t, R)$ denotes a Gaussian distribution with mean $\tilde{z}_t$ and covariance R. The multiplication between $\lambda(\tilde{z}_t)$ and $P(\tilde{z}_t|\tilde{S}_t)$ can be obtained by calculating the Battacharyya distance (D_B) as follows:

$$D_B\Big(\lambda(\tilde{z}_t), P(\tilde{z}_t|\tilde{S}_t = i)\Big) = -ln \int \sqrt{\lambda(\tilde{z}_t) P(\tilde{z}_t|\tilde{S}_t = i)} d\tilde{z}_t \tag{3.14}$$

where $i \in [1, M]$. The vector D_λ containing all the D_B values between $\lambda(\tilde{z}_t)$ and all the superstates assigned to the N particles is here estimated as:

$$D_\lambda = \Big[D_B(\lambda(\tilde{z}_t), P(\tilde{z}_t|\tilde{S}_t = \tilde{S}_t^{n=1})), \ldots, D_B(\lambda(\tilde{z}_t), P(\tilde{z}_t|\tilde{S}_t = \tilde{S}_t^{n=N}))\Big] \tag{3.15}$$

This means that we take the D_B distance related to the cluster assigned to each of the N particles.

Therefore, the vector $\lambda(\tilde{S}_t)$ in terms of probability is:

$$\lambda(\tilde{S}_t) = \left[\frac{1/D_\lambda(1)}{1/\sum_{l=1}^N D_\lambda(l)}, \ldots, \frac{1/D_\lambda(N)}{1/\sum_{l=1}^N D_\lambda(l)}\right] \tag{3.16}$$

After the weights are updated, the particles are resampled based on them, following the SIR approach.

In classical Bayesian filtering, predictive and diagnostic reasoning are used to estimate a joint posterior at different hierarchical levels. However, an indispensable step is missed in this process to evaluate the differences between two messages arriving at a given node based on a probabilistic metric that estimates the surprisal caused by observations that can not be explained by the model. The following section shows how to provide a self-aware agent with the capability to detect multimodal abnormalities which are the basis to continually update the knowledge about the perceived environment and encode new concepts related to the emergent abnormal behavior.

3.7.4 Multi-modal Abnormality Measurements

In the proposed approach, classical Bayesian inference (i.e., MJPF) is enriched with advanced functionality that exploits a probabilistic distance between top-down and bottom-up messages incoming to a generic node at different levels inside the GDBN to define hierarchical abnormality measurements. Such probabilistic distances quantify the similarity between two probability distributions ($\pi(\ddagger)$ and $\lambda(\ddagger)$) over the domain $\ddagger$ which can be continuous or discrete distributions. Several important probabilistic distances include the following:

- Bhattacharyya distance (D_B) [23]: this distance has been proposed to reflect the degree of dissimilarity between two probability distributions. It is based on the Bhattacharyya Coefficient (BC) that approximates the amount of overlap between two distributions. In the case of discrete probability distributions, BC is defined as follows:

$$BC(\pi(\tilde{z}),\lambda(\tilde{z})) = \sum_{\tilde{z}\in\tilde{Z}} \sqrt{\pi(\tilde{z})\lambda(\tilde{z})} \tag{3.17}$$

  while, in the case of continuous distributions:

$$BC(\pi(\tilde{z}),\lambda(\tilde{z})) = \int \sqrt{\pi(\tilde{z})\lambda(\tilde{z})}dz \tag{3.18}$$

  Thus, the Bhattacharrya distance D_B is defined through the BC in the following way:

$$D_B = -\ln\Big(BC(\pi(\tilde{z}),\lambda(\tilde{z}))\Big) \tag{3.19}$$

  In either case (discrete and continuous distributions), $0 \leq BC \leq 1$ and $0 \leq D_B \leq \infty$. If the two distributions are multivariate normal distributions, $\pi(\tilde{z}) \sim \mathcal{N}(\mu_1,\Sigma_1)$ and

$\lambda(\tilde{z}) \sim \mathcal{N}(\mu_2, \Sigma_2)$, the Bhattacharyya distance can be calculated as:

$$D_B = \frac{1}{8}(\mu_1 - \mu_2)^\intercal \Sigma^{-1}(\mu_1 - \mu_2) + \frac{1}{2} ln\left(\frac{det\Sigma}{\sqrt{det\Sigma_1 det\Sigma_2}}\right) \tag{3.20}$$

with:

$$\Sigma = \frac{\Sigma_1 + \Sigma_2}{2} \tag{3.21}$$

- Mahalanobis distance (D_M) [155]: the Mahalanobis distance (D_M) can be visualized as the distance of a point from a distribution by measuring how many standard deviations the point is away from the distribution's mean. Let us suppose that x is a row vector consisting of a multivariate measurement of an observation point and μ, Σ are the mean and covariance of the distribution. Then, D_M is computed as follows:

$$D_M = \sqrt{(x-\mu)^\intercal \Sigma^{-1}(x-\mu)} \tag{3.22}$$

  where, $0 \leq D_M \leq \infty$.

- Kullback–Leibler divergence (D_{KL}) [134]: the D_{KL} is a way to measure the matching between two probability distributions. If two distributions perfectly match then $D_{KL} = 0$, otherwise it ranges between 0 and ∞. The D_{KL} between two discrete probability distributions can be formulated as:

$$D_{KL}\Big(\pi(\tilde{z})||\lambda(\tilde{z})\Big) = \sum_{\tilde{z}\in\tilde{Z}} \pi(\tilde{z}) \log \frac{\pi(\tilde{z})}{\lambda(\tilde{z})} \tag{3.23}$$

  while for continuous distributions it is defined in the following way:

$$D_{KL}\Big(\pi(\tilde{z})||\lambda(\tilde{z})\Big) = \int \pi(\tilde{z}) \log \frac{\pi(\tilde{z})}{\lambda(\tilde{z})} dz \tag{3.24}$$

In the following subsections, we use some of the aforementioned probabilistic distances as examples to associate abnormality metrics with the different levels of the GDBN.

Discrete Level

The abnormality indicator at the discrete level is defined as a distance between the predictive ($\pi(\tilde{S}_t)$) and diagnostic ($\lambda(\tilde{S}_t)$) messages entering to node $\tilde{S}_t$. Such a distance provides an awareness signal indicating how the real sensory signals received from the surrounding environment behave concerning the rules encoded in the generalized model. A high discrete-

level anomaly can indicate, for example, an abnormal sequence of clusters, or that there is some difference between the characteristics of the clusters where we expect to move and of the data that we observe. We use the symmetric Kullback-Leibler Divergence as an example to measure the similarity between the two discrete probability distributions ($\pi(\tilde{S}_t)$ and $\lambda(\tilde{S}_t)$). The Kullback-Leibler Divergence Abnormality (KLDA) as defined in [132] has the following form:

$$KLDA = D_{KL}\Big(\pi(\tilde{S}_t)||\lambda(\tilde{S}_t)) + D_{KL}(\lambda(\tilde{S}_t)||\pi(\tilde{S}_t)\Big) \tag{3.25}$$

where $||$ identifies the divergence. At each time instant t the histogram of the predicted particles is extracted and the probability of occurrence of each particle is calculated as:

$$p(\tilde{S}_t = i) = \frac{y(\tilde{S}_t = i)}{N} \quad i \in [1, M] \tag{3.26}$$

where $y(.)$ is the frequency of occurrence of a specific superstate i and N is the total number of particles propagated by PF. It is worth noting that $\lambda(\tilde{S}_t)$ is unique for all particles at time instant t. Let's define $\mathbb{S}$ as the set of winning particles whose probability of occurrence is greater than zero such that:

$$\mathbb{S} = \{i | p(\tilde{S}_t = i) > 0\} \quad i \in [1, M] \tag{3.27}$$

D_{KL} is calculated between $\lambda(S_t)$ and the rows of the transition matrix related to the winning particles in $\mathbb{S}$. Hence, (3.25) becomes:

$$KLDA = \sum_{i \in \mathbb{S}} \Big[p(i) \sum_{j=1}^{M} \pi_{ij} \log(\frac{\pi_{ij}}{\lambda_j})\Big] + \sum_{i \in \mathbb{S}} \Big[p(i) \sum_{j=1}^{M} \lambda_j \log(\frac{\lambda_j}{\pi_{ij}})\Big] \tag{3.28}$$

Continuous Level

At this level, we focus on the messages entering node $\tilde{z}_t$ and calculate the corresponding abnormality measurements defined as statistical metrics based on a probabilistic distance (e.g. D_B, D_M, D_{KL}). The message $P(\tilde{z}_t|\tilde{S}_t)$ forwarded towards the intermediate level describes the probability of having the prediction $\tilde{z}_t$ in a certain superstate. Thus calculating the difference between the predictive message $\pi(\tilde{z}_t)$ and $P(\tilde{z}_t|\tilde{S}_t)$ allows us to evaluate if the predictions performed at the discrete level match the prediction done at the continuous level. For example, such a difference can be obtained through the D_B distance in the following way:

$$Db2 = D_B(\pi(\tilde{z}_t), P(\tilde{z}_t|\tilde{S}_t^n)) = -\ln \int \sqrt{P(\tilde{z}_t, \tilde{S}_t^n|\tilde{x}_{t-1}) P(\tilde{z}_t|\tilde{S}_t^n)} d\tilde{z}_t \tag{3.29}$$

On the other hand, it is important to evaluate how much the real observations support the predictions performed at the continuous level which leads to detecting any abnormal behavior occurring in the surrounding environment. The second abnormality at the continuous level can be calculated as the difference between the observations ($\lambda(\tilde{z}_t)$) and the predicted generalized states ($\pi(\tilde{z}_t)$) which is also based on the probabilistic distances defined previously (e.g. D_B, D_M, D_{KL}). For example, using the D_B distance, the second abnormality at the continuous level can be formulated as:

$$Db1 = D_B(\pi(\tilde{z}_t), \lambda(\tilde{z}_t)) = -\ln \int \sqrt{P(\tilde{z}_t, \tilde{S}_t^n | \tilde{x}_{t-1}) P(\tilde{x}_t | z\tilde{X}_t)} d\tilde{z}_t \tag{3.30}$$

Observation Level

The anomaly at this level is particularly informative in the case of high-dimensional data. In general, two types of observation-level anomalies can be distinguished: *i)* the direct *reconstruction error* due to having learned an observation model that is not suitable for the observed data. This is the case, for example, of anomalies due to previously unseen contents and elements appearing in the scene. These anomalies lead to a high discrepancy between the observed image, x_t, and its reconstruction, $\hat{x}_t$, through the VAE network. *ii)* the distance between the observation, x_t, at t and the predictive message $\pi(\tilde{x}_t)$ forward propagated from the continuous level towards the node x_t. From the practical point of view, this can be interpreted through the calculation of a frame (and OF anomaly) such as the Mean Squared Error (MSE) between the predicted image, $\hat{x}_{t-1}$, at time $t-1$ and the observed image, x_t, at time instant t.

3.7.5 Use of Generalized Errors for Continual Learning

In the previous sections, we have seen how the described framework allows us to enrich the GDBN architecture and leverage its message-passing concept for the calculation of anomalies and GEs.

The objective of anomaly indicators is to provide the agent with the capability of understanding whether the models that it is using are suitable or not to predict the evolution of the world and of its own state in the situation it is sensing and experiencing. As shown in Fig. 3.2, in the "Continual Learning" diagram, when high anomalies are detected, the agent is alerted that it should learn a new model. It consequently stores the corresponding GEs.

On the other hand, the purpose of GEs is to identify which actions the filter has to execute to correct the predictions it performed. A new model learned from the GEs minimizes the GEs on the same type of sequence as the one on which they had been identified. By

minimizing the GEs, surprisal and free energy are minimized too. A clustering process is performed again and a new model is inserted in the set of learned models.

To summarize, anomalies indicate *that a new model should be created*, and *which part of the original model should be updated* whereas GEs instruct the agent on *how* the new model should be created.

It is to note how, in the high-dimensional case, some limitations to the Continual Learning through GEs are present. New observations (e.g., new objects appearing in the scene, new types of environments, etc.) could be detected. In this case, the training of a new VAE is necessary. In this situation, testing requires the use of multiple VAEs (and so, multiple observation models) in parallel.

Note that, whereas we introduce the concept of Continual Learning and how it is tackled to give a full idea of the proposed framework, this thesis does not address this part of the framework.

3.7.6 The Adopted Framework and the five Desiderata of self-awareness

After presenting the adopted framework, we can observe how it complies with the five desiderata of self-awareness that we have identified in Sections 3.2.3 and 3.3. We reiterate here that the desired framework should:

- perform Bayesian inference;
- be hierarchical;
- be multi-sensorial;
- be data-driven;
- be interpretable (or explainable).

The proposed framework possesses these characteristics thanks to the use of DBNs. Indeed, DBNs allow us to model the process of Bayesian inference between random variables arranged in a hierarchy, thus complying with the first two desiderata. Additionally, we could potentially join the DBNs of multiple sensors, making the model multi-sensorial, as we will perform in Chapter 7. Although low-levels of the hierarchy will be dishomogeneous, because observations are obtained from sensors of different nature and dimensionality, the higher levels of the hierarchy can work in a homogeneous way.

Moreover, in Section 3.7.2, we have seen how the links in DBNs can be learned in a data-driven way. As for the interpretability, we have already discussed in Section 3.5 how

DBNs are an interpretable model and how they can further be enhanced through anomalies to identify which part of the model should be learned anew. Much more detail regarding this last characteristic has been provided in Sections 3.7.3, 3.7.4, and 3.7.5, which also serve to remark once again the importance of anomalies in the adopted framework.

3.8 Other Related Frameworks

In this section, we compare the adopted framework with other proposed state-of-the-art frameworks and approaches.

Few self-awareness approaches have been presented throughout the years, as this area is still in its infancy [96].

One seminal contribution to the self-aware field is the work by Lewis et al. [36, 64, 140]. The authors distinguish five types of self-awareness: stimulus awareness, interaction awareness, time awareness, goal awareness, and meta self-awareness. Eight patterns for combining these types of self-awareness are introduced. This design has been applied to fields such as cloud computing and smart camera networks [36]. However, the discussion of self-awareness proposed in these works is very high-level, and the reached self-awareness in the presented practical examples is mainly limited to a form of self-monitoring and self-modeling.

Similarly, also the work in [50] proposes a high-level view of self-awareness. It focuses on verification and explainability through a self-aware approach. Verification relates to techniques to assess whether an agent is meeting its requirements. The approach proposes a "hybrid agent", which obtains continuous processed information (e.g., object recognition and engine monitoring from the sensors), performs decision-making by selecting between a set of high-level discrete choices, and outputs continuous control signals. The agent is programmed in terms of BDI principles [26]: it contains Beliefs about the world (and itself), Desires that capture its long-term goals, and Intentions that encode its plans or courses of action. The authors also propose to monitor the agent's motives behind its course of action, its progress toward the goal, the effect of its actions on the world, etc. However, the treatment of the problems is tackled in a very high-level way, and low-level methods are not proposed. Nevertheless, it is worth noting that the authors of this work remark on the importance of some aspects that we have also covered: they discuss explainability, they suggest taking into account the reliability of different modules (or sensors), they propose the use of a hierarchical structure by distinguishing continuous low-level controls and discrete high-level decisions.

In [84], a self-awareness approach is presented to detect failures in the operations of a robot following human commands. Anomaly detection is performed. Events are detected in

the robot's operations. Events are a discrete concept extracted from the elaboration of the input data. The method calculates the probability of an event e_i to follow an event e_j, given that t time instants have passed between the two. For each couple of events, a histogram is computed, with each histogram bin corresponding to a different t. During the online testing phase of the method, the learned histograms are adopted to find an anomaly score for the sequence of events. The method is, therefore, probabilistic (even if not through Bayesian inference) and data-driven. Discrete variables (the events) are calculated and used; however, continuous state variables are not elaborated for anomaly detection. Therefore, the hierarchical structure of the method is not fully leveraged. No mention of multi-sensoriality or interpretability is provided.

Another self-awareness approach is introduced with the RoboErgoSum project [34]. The presented cognitive architecture is composed of several components. First, the *Sensorial perception module* perceives the environment through visual (RGB-D) exterocetive data and proprioceptive values from the robot's joints. These inputs are preprocessed to detect objects, actions, and their effects. Preprocessing of visual data is performed by voxelising and clustering the RGB-D point clouds. The *Motor module* includes the possible actions that the robot can execute. The *Sensorimotor Learning* module elaborates the preprocessed data from the sensorial perception module. A Dynamic Bayesian network is learned, to discover the dependencies between the different variables, including the robot's actions. The *Spatial reasoning and knowledge module* and the *Knowledge base module* create and memorize symbolic data of the perceived environment. Since the architecture is primarily directed at social robots that interact with humans, two modules for this task are introduced (i.e., the *Human-aware task planning module* and the *Human-aware motion and manipulation planning module*). As Reinforcement Learning is employed for action selection, a *Reinforcement Learning model-free decision-making system* is also present. Finally, the *Motivation module* handles the robot's goals, and the *Supervision system* supervises the other modules during action planning, and to monitor the interaction with humans. Consequently, this method is probabilistic, data-driven, multi-sensorial, and potentially hierarchical. No mention of interpretability or explainability is made.

Other approaches have been proposed in areas such as embedded systems [55], cloud computing [35], or healthcare monitoring [8]. In the field of robotics, self-aware models were employed, among other applications, for social robots [208, 188], for swarms and multi-robot organisms [243, 128], and for underwater robots [3]. Software frameworks for quick application development of self-aware systems have also been proposed, such as the Research on self-awareness (RoSA) framework [89], which is based on an Obserse-Decide-Act loop

[97], and introduces features for abstracting data information, checking data reliability and content, and recording data history.

As we observed in Section 3.1, self-awareness approaches fall under the umbrella of cognitive methods. It is worth noting that, in 2018, Kotseruba et al. conducted a survey in which they identified approximately three hundred cognitive architectures, adopted in around nine hundred projects [129]. Additionally, one should take into account the cognitive architectures developed in the five years following that survey, further adding to this substantial body of work. In the rest of this section, we choose to concentrate on the comparison against one recent architecture proposal, i.e., the Joint Embedding Predictive Architecture (JEPA), as it assembles various ideas formulated by many authors in various contexts into a consistent whole [136, 47].

In 2022, notorious computer scientist Yann LeCun presented a proposal for the JEPA, an Autonomous Machine Intelligence architecture, drawing from different scientific backgrounds [136, 47]. At present, some aspects of it are still under analysis. It is composed of six main components: the Configurator, the Perception Model, the World Model, the Cost module, the Actor Model, and the Short-Term Memory. The Configurator obtains inputs from all the other modules and configures them depending on the agent's goal. The Perception Model uses sensory observations to estimate the state of the world, possibly in a hierarchical way. The World Model predicts future World states at different levels of the hierarchy and at multiple time scales. The Cost module is divided into two types (Intrinsic Cost module and Critic module) and evaluates the level of "discomfort" that the agent feels through a variable called "energy". Indeed, this framework employs Energy-Based Models [137] and seeks for the agent to take actions that allow it to remain in states minimizing the average energy. The Intrinsic Cost module is non-trainable and is modulated by the Configurator depending on the agent's goal. On the other way, the Critic module is trained to predict the future values of the intrinsic cost. The Actor Model chooses the sequence of actions that minimizes the cost. The Short-Term Memory stores the past, current, and future states of the world and intrinsic costs.

Two possible modes of conducting a perception-action episode are identified. In the most complex of the two, reflecting traditional Model-Predictive Control, the Perception Model first extracts a representation of the world, and the Actor Model then proposes a sequence of actions to undertake. The World model simulates possible sequences of world states deriving from the proposed actions, and the cost module evaluates the associated costs. The action sequence resulting in the lowest cost is selected. After every action is performed, the states and intrinsic costs are saved in the Short-Term Memory, so that they can later be employed to train the Critic.

We highlight two important aspects of this architecture. Firstly, the World Model does not learn to predict how World observations evolve, because not every perceived detail is relevant to accomplish the agent's goal. Consider, for example, an autonomous vehicle navigating a road: the World model needs to predict how pedestrians are crossing the road, but not how tree leaves are swaying in the wind. Therefore, the observations are encoded to a lower-dimensional space encapsulating information pertinent to the current objective. The World Model then focuses solely on learning how this projection evolves. This concept will be relevant again in Chapter 7. The other relevant aspect of this architecture is that it is a non-generative architecture. Instead, an Energy-Based Model is learned, which captures the dependencies between two states x and y through a scalar-valued energy function $F(x,y)$. This function takes low values when x and y are compatible, and high values otherwise [136].

Our assessment takes into account whether the considered framework possesses the desired characteristics that we have identified in the past sections.

Table II summarizes this comparison. To the five columns related to the characteristics defined above, we add columns for the model being "generative" and "probabilistic", as an architecture could use generative models or be probabilistic, without adopting Bayesian inference.

Firstly, the JEPA is hierarchical. As we have described, the Perception Model estimates the state of the world in a hierarchical way, and the World model predicts future World states at different levels of the hierarchy. Furthermore, the JEPA is data-driven, and multi-sensorial. Probabilistic reasoning could be added but is not seen as necessary by the author. In contrast, the JEPA does not use Bayesian Inference. It is also not generative. Interpretability and explainability are not discussed.

Table II Comparisons between the adopted framework and the JEPA. "P" identifies that the considered element could "potentially" be included, but its inclusion is not seen as necessary by the authors. "-" means that the authors don't elaborate on the considered element.

	JEPA	Ours
It uses Bayesian Inference	✗	✓
It is generative	✗	✓
It is probabilistic	P	✓
It is hierarchical	✓	✓
It is multi-sensorial	✓	✓
It is data-driven	✓	✓
It is interpretable/explainable	-	✓

3.9 Conclusions

In this chapter, we have discussed the state of the art relevant to the thesis. First, we have briefly reviewed the evolution of autonomous vehicle research and the incorporation of cognitive architectures and self-awareness into this domain. The neurobiological inspiration behind the framework adopted in this thesis is then discussed, in particular, the concepts of the brain as a probabilistic predictive machine, free energy minimization, attractors, and Generalized States. Four desirable characteristics of self-aware systems are derived from neuroscience: the use of Bayesian inference, a hierarchical structure, data-driven learning, and the adoption of multiple sensors. Furthermore, we examined the importance of the framework being interpretable or explainable. Six basic capabilities are also identified that are necessary for a system to be considered self-aware: initialization, inference, anomaly detection, model creation, and interface with control. In this thesis, the first four are examined. Since anomaly detection constitutes one of these six capabilities, we propose a literature review of this topic. A literature review is also provided for Visual-Based Localization because this is a fundamental capability of autonomous vehicles. While discussing state-of-the-art methods, we also compare them in a high-level way to the approaches presented in this thesis.

Throughout the above-described sections, we provide all the information necessary to understand the adopted framework. Therefore, we then describe the framework, we show how it includes the five identified desirable characteristics, and we compare it with other self-aware approaches and with the JEPA cognitive architecture proposed by Yann LeCun.

Chapter 4

The Adopted Evaluation Datasets: Extraction and Overview

Before moving to the chapters covering the main proposed methods, in this chapter, we examine the evaluation datasets employed throughout the thesis.

First, in Section 4.1, the extraction of an indoor drone dataset is considered. This dataset includes onboard video and odometry measurements from the drone and, additionally, sensory observations from an external lidar capturing the drone's movements from above. The instrumentation setup for these experiments is described in Section 4.1.1. Section 4.1.2 discusses how this dataset will be used in Chapter 8. For this purpose, we propose a methodology for localizing the drone using onboard sensors such as cameras and IMUs, as well as localizing it using lidar data (Section 4.1.3). Results are analyzed in Section 4.1.5.

Additionally to this drone dataset, several other datasets are adopted in the other methodology chapters. Therefore, in Section 4.2, we describe every evaluation dataset employed in the thesis and provide some first examination of their relevant characteristics.

In Section 4.3, the extraction of a dataset with the Carla simulator [53] is also discussed.

Finally, Section 4.4 summarizes the main datasets information and Section 4.5 draws some conclusions.

4.1 Drone dataset extraction

In this section, we detail the process of extracting an indoor drone dataset that can be used to evaluate methods for various applications, and, in particular, for anomaly detection and localization. In this thesis, it is employed in Chapter 8 to evaluate the localization capabilities of the proposed self-awareness method.

Indoor localization presents a distinctive set of challenges, notably the inapplicability of GPS-based positioning methods. Alternative sensor modalities are thus needed, such as IMUs, cameras, Ultra-wideband (UWB) technology, Light Detection and Ranging (lidar), Kinect and more [237]. In accordance with this, Chapter 8 explores a novel self-awareness method harnessing camera data for real-time localization and anomaly detection, which could be combined with other sensors such as IMU, or with other methods (potentially also camera-based). However, this method trains its models through known synchronous positions and images. Therefore, localization must first be performed on the training dataset with other methods. This section consequently explores localization with images, IMU, and lidar, adopting traditional Signal Processing methods.

First, we perform localization leveraging the onboard drone sensors. We combine positioning through external fiducial markers with the results of Visual Inertial Odometry (VIO). Then, we perform a localization using a more precise external sensor, a lidar, which can be employed as ground truth. The two localizations are aligned and synchronized.

Fiducial markers are artificial landmarks with a known size and shape, and are identifiable because they display a specific pattern [113]. Most fiducial markers are squared and in black and white, but colored or circular markers exist too [113]. Several papers compare fiducial markers such as AprilTag, ArUco, STag, and ARTag, and more, [112, 113, 49] from the point of view of accuracy, detection rate, and computational cost. As ArUco was shown to have a good trade-off between these characteristics, we chose this fiducial marker.

Fiducial markers have been used in the localization literature, either alone [164] or in combination with SLAM [147] or VIO [163]. For example, Mráz et al. [163] employ ArUco markers because they add a global positioning and can correct the drift of the VIO.

The localization obtained from ArUco markers is sparse and absolute. It is sparse because the markers could not be visible along the entire path of the vehicle, as is the case in our dataset; it is absolute because the precise absolute position of the markers is known.

On the other hand, the VIO uses camera and IMU data and provides a continuous localization that degrades with time due to drift. Additionally, this localization is relative to the first-time instant and is not in an absolute coordinate system. A positional reference for the initial instant is needed. In the proposed method, the first observed ArUco marker is employed for this purpose.

Finally, the lidar is an active ranging sensor. It uses laser light to measure distance and create 3D representations of an environment. In this section, we employ the lidar to track the drone. The obtained continuous localization is used as ground truth to test the localization from both the traditional Signal Processing methods on the onboard sensors and the self-awareness method in Chapter 8.

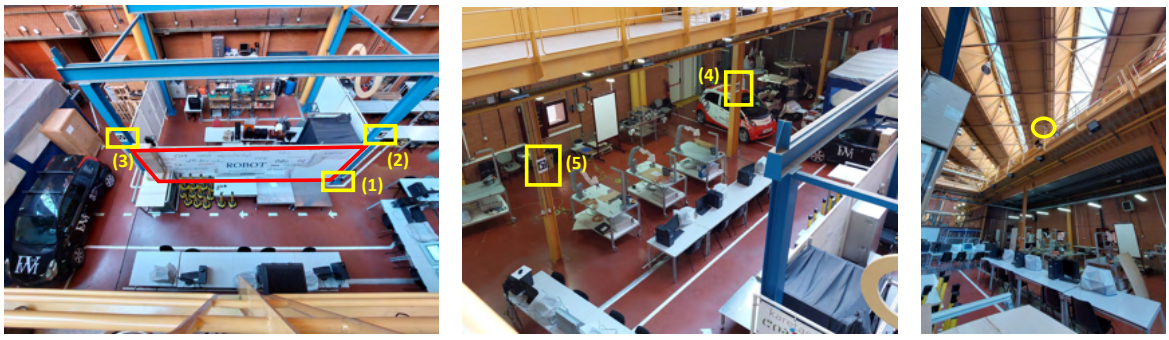
4.1.1 Instrumentation setup

The experiments were performed employing the following main instrumentation:

- A DJI FLY 2S drone.
- An Ouster OS1 lidar.

The drone was piloted by a human operator, connected through wifi using a controller. The drone collects a rich variety of information, including RGB frame recordings from a frontal camera (with 3840x2160 resolution and 30 Hz frame rate), measurements from the accelerometer, gyroscope, and magnetometer (from two IMUs on board), temperature, commands sent from the controller, and battery charge level. From these measurements, it additionally performs estimation of connected quantities, as, for example, pitch, roll, and yaw, velocity northwards, eastwards and downwards, and localization through VIO.

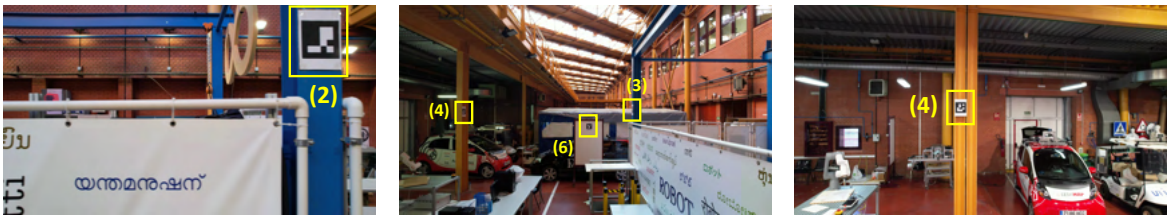
The lidar has a range from 0.3 to 120 m, a vertical field of view of 45°, a vertical angular resolution of 0.35°, a precision of around 0.7-5 cm, and a rotation rate of 20 Hz.



(a) (b) (c)

Fig. 4.1 Flying environment observed from an external perspective. (a) and (b) display the environment from the corridor above. The ArUco markers are highlighted in yellow beside their identification number. (c) shows the environment from below, with the lidar position highlighted.

The drone flies in the environment shown in the photos in Fig. 4.1, moving in the area delimited by the ArUco markers highlighted in yellow (i.e., an area of around 7m x 5m). Fig. 4.2 illustrates further photos of the environment, in this case taken from the drone onboard camera. The drone flies in the environment on the first floor, moving in the rectangular space delimited by the two blue columns with ArUco markers on one side and by the two yellow columns with ArUco markers on the other side. The lidar is positioned in an open corridor above this environment, on the second floor.



(a) (b) (c)

Fig. 4.2 Flying environment observed from the drone. The ArUco markers are highlighted in yellow beside their identification number. Note that, in (b), also the optional marker (6) is present.

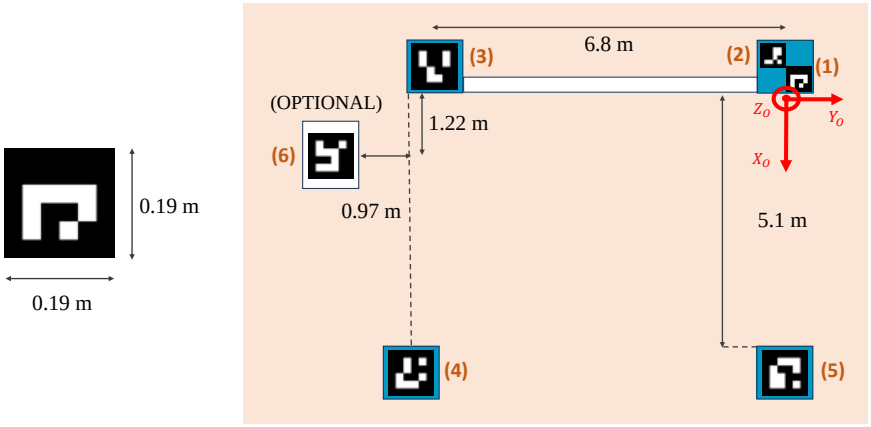


Fig. 4.3 Scheme of the ArUco placement in the flying environment. The ArUco identifications numbers are the same adopted in Figs. 4.1 and 4.2. The origin coordinate system ($X_OY_OZ_O$) displayed in red is relevant in Section 4.1.3.

In Fig. 4.3, we provide a scheme of the ArUcos placement, and of the precise distances between them. Moreover, a number is assigned to each marker, which is also used in preceding photos, to facilitate recognizing which marker in the scheme corresponds to which marker in the photos. Furthermore, in red, a coordinate system $X_OY_OZ_O$ is displayed, which will be extensively referenced in Section 4.1.3. This coordinate system is placed in what will be later referenced as the origin.

For each different type of experiment, the visible ArUco markers vary from three (i.e., the ones in Fig. 4.1a), to five (i.e., all the ones in Figs. 4.1a and 4.1b), to six (i.e., all the ones in Figs. 4.1 and 4.2). The first five markers are all attached to columns present in the environment. The sixth marker, displayed in Fig. 4.3 as "optional" and visible in Fig. 4.2b, is inserted in only one of the experimental scenarios and is placed on a white board.

The adopted marker dictionary is 4*x*4. The dictionary is the set of markers employed. A 4*x*4 dictionary is composed of ArUcos with 4 inner squares per side, i.e., 16 inner squares in total. Markers with more inner squares provide more symbols. Tests were performed to compare this dictionary with the 6*x*6 one. Due to the decrease in detection accuracy with the 6*x*6 dictionary compared to the 4*x*4 dictionary, and to the lack of necessity of a bigger dictionary, we chose to use the 4*x*4 one.

(a) (b) (c)

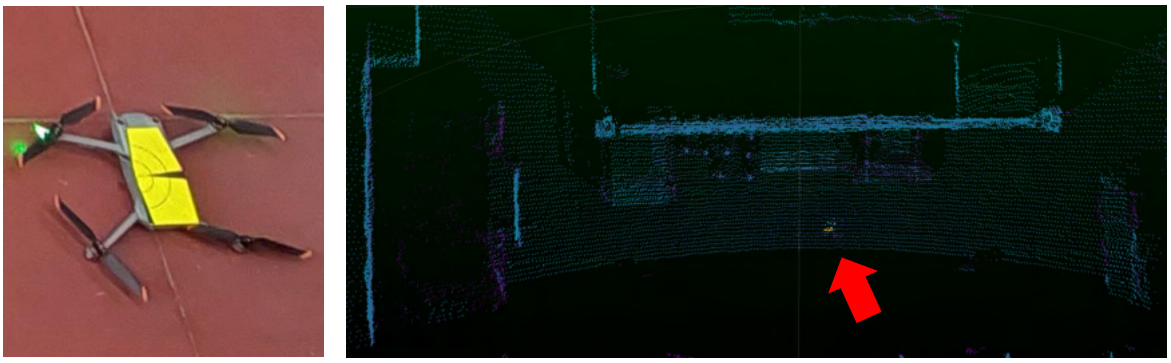
Fig. 4.4 The lidar sensor observed from different perspectives. In (b) two reference systems are compared: the lidar coordinate system (X_LZ_L) and the origin coordinate system (X_OZ_O). To enhance clarity in the figure, the z-axis of the system is omitted. These systems are relevant in Section 4.1.3.

The lidar observes the drone from a corridor above the flying environment. Fig. 4.1c highlights in yellow where the lidar is positioned, from a point of view inside the flying environment. On the other hand, Fig. 4.4 displays the lidar from above and from a closer viewpoint.

In Fig. 4.5b we display a point cloud (PC) captured from the lidar. The points are color-coded based on their intensity, which depends on the reflectivity of the objects. Darker colors indicate lower intensity, while brighter colors represent higher intensity. The yellow points indicated by the arrow correspond to the drone. This outcome is achieved by attaching a marker, made of highly reflective material, onto the drone, which facilitates its tracking. Fig. 4.5a shows the drone and the highly reflective marker (that is easily identifiable in the photo by its yellow color).

4.1.2 Use of the drone dataset in the context of this thesis

We extract a set of indoor drone datasets to use them as evaluation datasets for anomaly detection and localization methods such as the ones proposed from Chapter 5 to Chapter 8.



(a) (b)

Fig. 4.5 (a): the DJI Fly 2S drone, with the marker with high reflectivity. (b): example of a lidar point cloud. The red arrow points to the drone, which can be easily detected due to the high-reflectivity of the marker on top of it.

In this thesis, we use these extracted datasets solely for evaluating the method proposed in Chapter 8, which is centered on VBL. As seen in Chapter 3, VBL methods are trained with both image and odometry data; however, during testing, only the image data are available, and the corresponding odometry values are predicted from the learned model.

Therefore, Fig. 4.6 illustrates how the extracted datasets can be used in the context of the proposed VBL self-awareness method in Chapter 8. On the left, in green, we display some sensory inputs directly available from the datasets: the image, IMU, and lidar data. When applying the self-awareness methods proposed in the following chapters of this thesis, the data are divided in two sets: a training set without anomalies, and a testing set with anomalies.

In the center of the figure, in violet, we represent the preprocessing method described in Section 4.1.3, which enables us to obtain the position of the drone from either the onboard sensors of the drone (see above in the figure) or from the external lidar (see below in the figure).

When performing training of the VBL self-awareness method, the camera images and positions of the drone are employed. The training is unsupervised for what concerns the anomaly detection, but supervised for what concerns the localization: a loss is built between the positions extracted from the onboard drone sensors and the positions predicted by the self-awareness model.

Then, during the testing phase, we provide the drone camera testing images to the VBL self-awareness method, which extracts anomalies and predicts the drone localization. The predicted localization is compared to the one derived from the lidar, which acts as ground truth (GT).

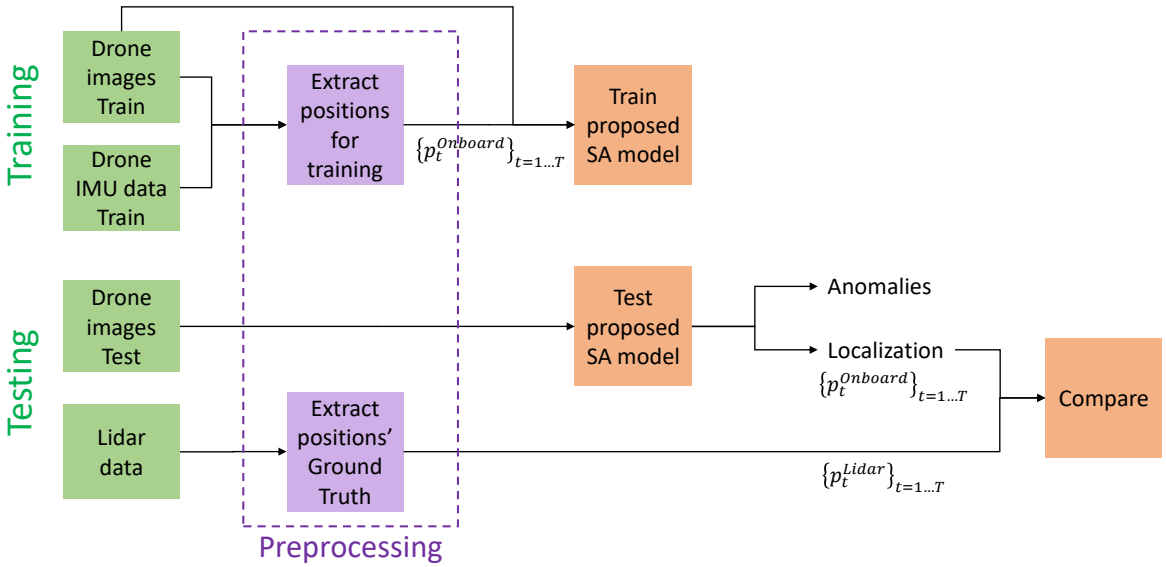


Fig. 4.6 Scheme displaying how the drone dataset is used for evaluating the approach proposed in Chapter 8. In the figure, SA stands for Self-Awareness.

Furthermore, as we observed in Section 3.3, our self-awareness models aim to perform incremental learning when exposed to new situations, such as new visual observations or previously unexplored motions. Consequently, incremental learning could be performed on the localization models too. In this case, as during training of the self-awareness model, the traditional Signal Processing localization from the onboard sensors is necessary again.

Note that, during training, the localization from the onboard sensors is adopted; conversely, during testing, we use the localization from the lidar. The rationale behind this choice is the following: *i)* the localization adopted to train the drone should not be too complex or expensive to obtain. Not all users could have costly instrumentation such as a lidar. Moreover, not all environments could potentially be apt for the use of an external lidar or other precise external localization methods. In our experiments, we chose a wide environment with an upper corridor, which easily allowed the adoption of an external lidar. However, narrow environments with several small rooms, corridors, tunnels, or containers (such as the interior of a ship) could not allow its use; *ii)* conversely, the localization for comparison during testing is solely employed as ground truth and is not needed by everyone who desires to train the model. A ground truth should be precise, and this is the case for lidar localization: as we observed in Section 4.1.1, the OUSTER OS1 lidar precision is 0.7-5 cm.

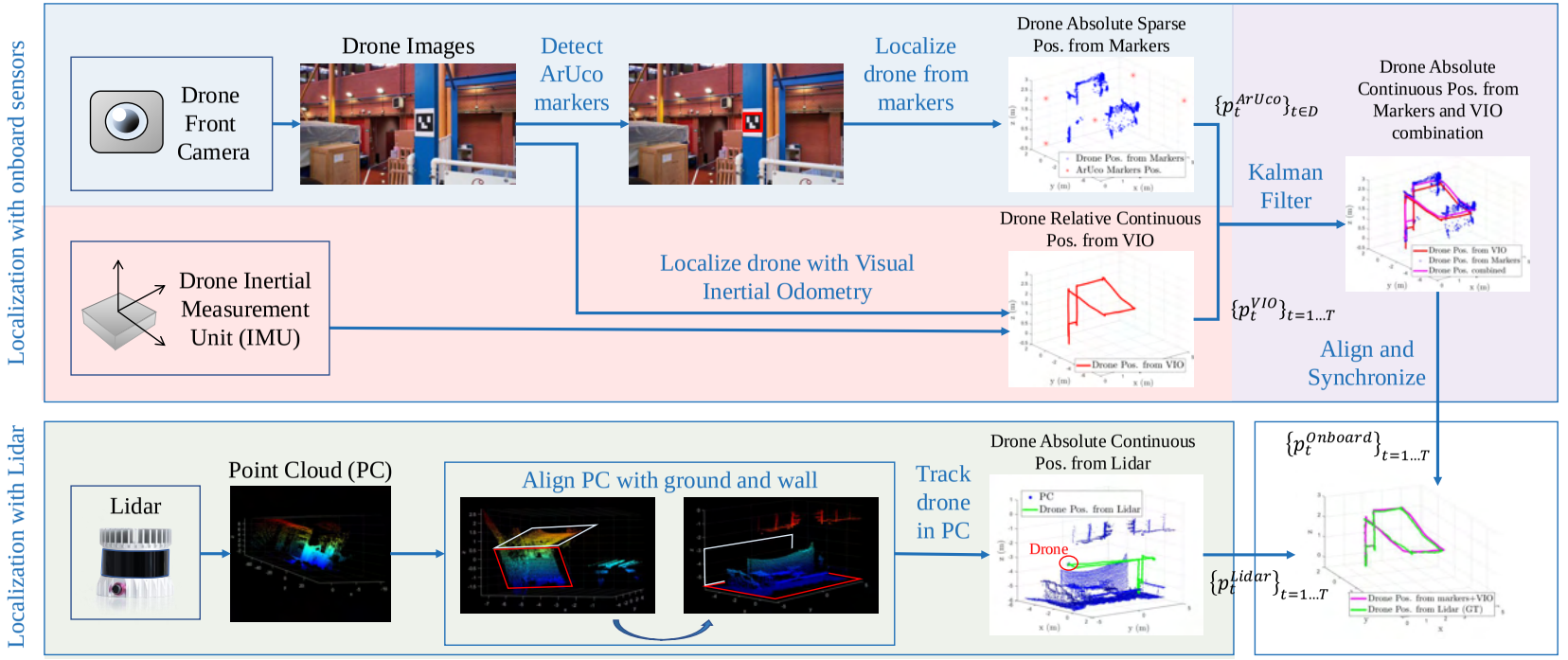


Fig. 4.7 The overall process for the extraction of the localization, from both the onboard sensors (above) and the external lidar (below), and their alignment and synchronization (below).

4.1.3 Proposed preprocessing method

Fig. 4.7 displays the proposed method. Three main steps can be identified: *i)* the localization of the drone through onboard sensors (upper part); *ii)* the localization of the drone through the external lidar (lower part on the left, in green); *iii)* the synchronization of the two localizations (lower part on the right, in white).

The first part takes as input the camera and IMU data of the drone and comprises three elaborations. On one side, the ArUco markers are detected in the images and localized; a sparse absolute localization is obtained from them (see the blue part in Fig. 4.7). It is sparse because the ArUco markers are not visible throughout the entire trajectory of the drone; it is absolute because we know the precise absolute position of the ArUcos in the environment. On the other side, the drone performs Visual Inertial Odometry using the camera and IMU data, providing a continuous relative localization (red part in Fig. 4.7). This localization is not absolute but is relative to the position in the first time instant. Finally, the two localizations are joined through a Kalman Filter to obtain an absolute and continuous positioning (violet part in Fig. 4.7).

The second main block of the method takes as input the lidar point clouds. First, it aligns the point cloud to the plane of the ground and of the wall of the environment and then tracks the drone (but the two operations could be performed in an inverted order).

Finally, the localization from the drone and from the lidar are aligned with the position of the first time instant and are synchronized in time.

The following paragraphs describe all the processing in detail, step by step.

Sparse absolute localization from the ArUco markers

The ArUco markers are detected in the drone camera images using the cv2.aruco python library. When an image x is given as input to this elaboration, a 3D translation vector t_{a2d} and a rotation vector r_{a2d} are obtained as output. These represent the pose of the ArUco marker relative to the drone camera coordinate system $X_dY_dZ_d$. From the rotation vector, the Rodriguez rotation matrix R_{a2d} is calculated. To obtain the drone localization in the environment, we need to first have the drone camera position relative to the ArUco marker coordinate system. Therefore, we compute the transpose of the Rodriguez matrix (as it is equal to its inverse) and multiply it with the translation vector:

$$p_{d2a} = R_{a2d}^{\mathsf{T}} * t_{a2d}, \tag{4.1}$$

where p_{d2a} is the position of the drone camera compared to a coordinate system centered on the ArUco marker, $X_aY_aZ_a$, as seen in Fig. 4.8. However, we need the position of the

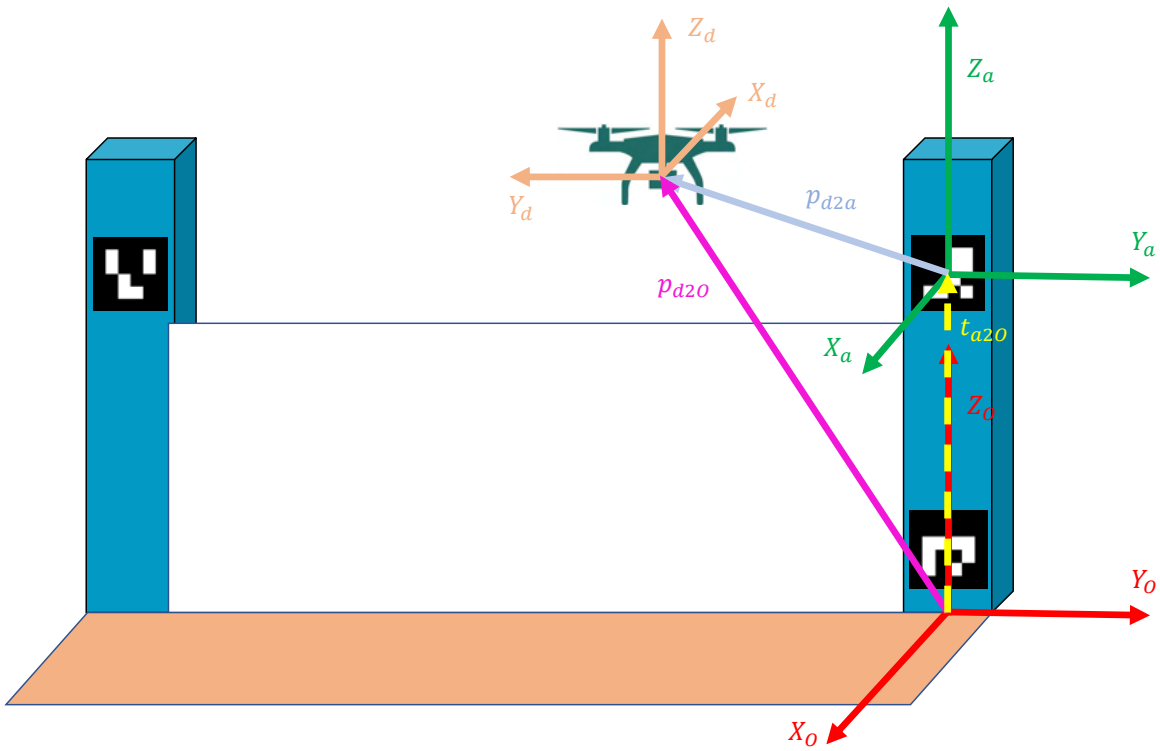


Fig. 4.8 Representation of the relevant coordinate axes and position vectors for the localization of the drone from the ArUco markers. The ArUco taken as an example for the drone localization description in this image is the one on the upper right part of the image. Note that the scale of the elements in the image does not correspond to reality.

drone camera in the origin coordinate system $X_O Y_O Z_O$ (see Fig. 4.3 to determine how it is positioned in the environment scheme). To perform this last passage, we need to measure the translation and rotation of the ArUco marker from the origin coordinate system. We identify these values with t_{a2O} and r_{a2O}. We consequently calculate the rotation matrix R_{a2O} from r_{a2O}. Finally, we obtain the desired position of the drone from $X_O Y_O Z_O$ by applying the following equation:

$$p_{d2O} = R_{a2O} * p_{d2a} + t_{a2O}, \tag{4.2}$$

A representation of the described coordinate systems and position vectors for the localization of the drone from the ArUco markers is shown in Fig. 4.8.

From this process, repeated for each camera image where an ArUco is detected, a sparse absolute localization is obtained. We define these estimated positions with $\{p_t^{ArUco}\}_{t \in D}$, where D is the set of time instants corresponding to ArUco detections. As already remarked,

this localization is sparse because, in some zones along its path, the drone does not see any ArUco markers.

Continuous relative localization from Visual Inertial Odometry

As a second step, a continuous relative localization can be obtained from the drone by joining the IMU and camera data to perform VIO. The DJI Fly 2S drone already provides a VIO output. Consequently, we take this output as the second localization. We define these estimated positions with $\{p_t^{VIO}\}_{t=1...T}$, where T is the total number of time instants of a drone trajectory.

Final continuous absolute localization from onboard sensors

Once the two separate localizations are obtained, they can be joined. Note, however, that the VIO localization is not in the $X_O Y_O Z_O$ coordinate system. Consequently, it must first be aligned with it, i.e., it must be aligned with the localization from the ArUcos. The coordinate transformation can be calculated using the first-time instant (in which an ArUco marker is always detected). This transformation can then be applied.

A Kalman Filter joins the two localizations. At each time instant, the KF uses as observation the position obtained from the ArUco (if available), and as prediction the velocity between two positions obtained from the VIO elaborated by the drone. Additionally, the observation covariance is not kept constant: its values are increased when the distance of the camera from the ArUco increases. It also increases as the disparity between two consecutive positions derived from the ArUcos becomes more pronounced.

We define the final estimated positions at the output of the KF with $\{p_t^{Onboard}\}_{t=1...T}$.

Alignment of the lidar point cloud

We now dedicate this and the following section to describing the localization of the drone through the external lidar sensor.

This localization must be aligned with the $X_O Y_O Z_O$ coordinate system too. For simplicity, in this thesis, we choose to perform the alignment in a way that is applicable to environments delimited by a flat ground and a vertical wall, as the one in which we are operating.

We define with $X_L Y_L Z_L$ the original coordinate system of the lidar, that, as we said, does not correspond to $X_O Y_O Z_O$. See Fig. 4.4b for a graphical depiction where, for simplicity of representation, only the X and Z axis are displayed (Y_O and Y_L in this scenario are roughly the same and, in the figure, are in the direction of the yellow railing, pointing far from the camera). Moreover, in this figure, we moved the origin point of the $X_O Y_O Z_O$ coordinate

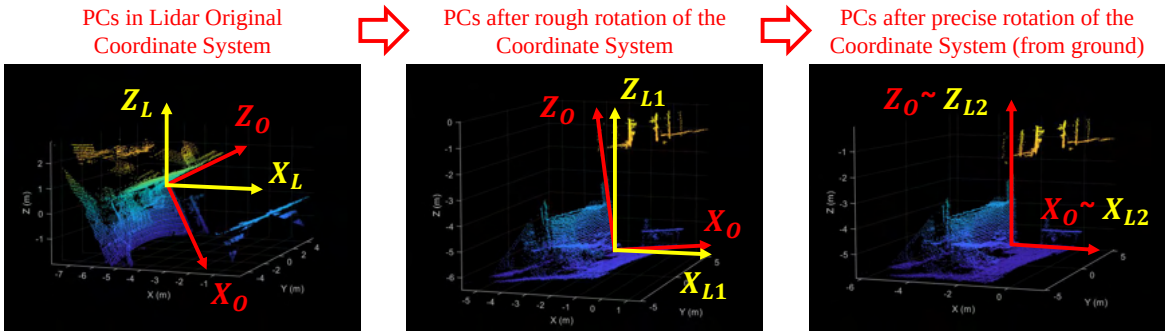
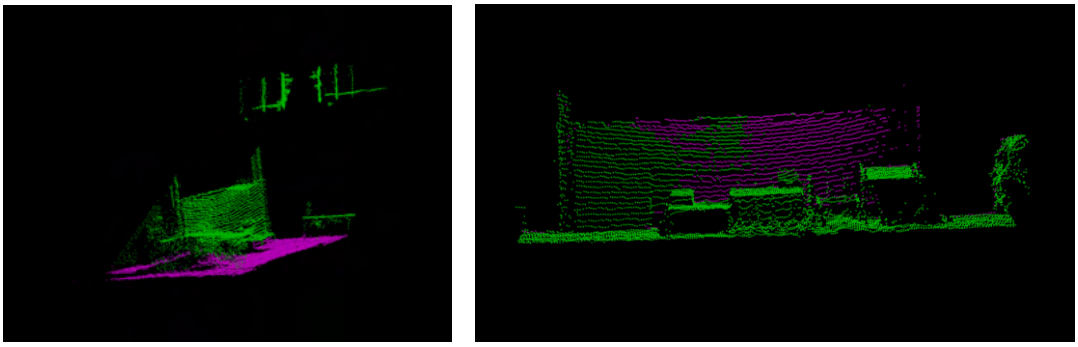


Fig. 4.9 Alignment process of the ground plane. The different coordinate systems described in the text are plotted over the lidar point cloud.



(a) (b)

Fig. 4.10 Detection of the ground (a) and of the wall (b). The points detected as belonging to either the ground or the wall are illustrated in violet; the rest of the lidar point cloud is displayed in green.

system to the lidar sensor for a better comparison with $X_L Y_L Z_L$. The true origin point of $X_O Y_O Z_O$ can be observed in Fig. 4.3.

The first plot on the left in Fig. 4.9 also shows the lidar PC against the axes in the two coordinate systems, now centered in the origin point of $X_O Y_O Z_O$. Remember that this is not the origin point of $X_L Y_L Z_L$, but this coordinate system is moved in the figures to simplify the comparison of the axes.

We don't know precisely which is the rotation vector r_{LO} that brings from $X_L Y_L Z_L$ to $X_O Y_O Z_O$, however, we can obtain a rough estimate r_{LL_1} measuring how the lidar is pointing in Fig. 4.4. This value does not have to be precise: the rotation will be refined by estimating both the ground plane and the wall plane. The rough estimate is first needed because lidar ground detection algorithms are employed, which assume that the ground is flat and approximately located on the XY plane.

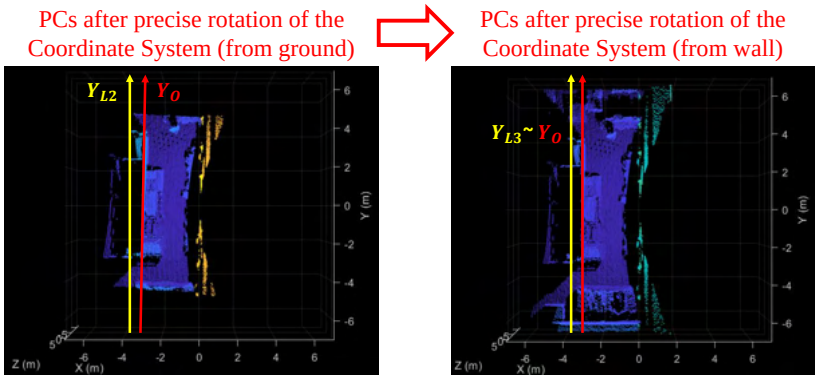


Fig. 4.11 Alignment process of the wall plane. The different coordinate systems described in the text are plotted over the lidar point cloud.

Having r_{LL_1}, we can rotate the point cloud, leading to a coordinate system $X_{L1}Y_{L1}Z_{L1}$, depicted in the center in Fig. 4.9. A ground detection algorithm is applied [176], as displayed in Fig. 4.10a, where the detected ground points are represented in violet and the rest of the PC in green. The plane of the ground and its normal are calculated. The angles between these normals and $X_{L1}Y_{L1}Z_{L1}$ are computed, leading to a rotation vector $r_{L_1L_2}$. The application of this rotation to the PC leads to the coordinate system $X_{L2}Y_{L2}Z_{L2}$, shown on the right in Fig. 4.9. The axes Z_{L2} and X_{L2} align better to Z_O and X_O than Z_{L1} and X_{L1}.

However, the alignment of the point cloud can be refined further by leveraging the plane of the wall. The wall in the environment can be treated as a ground and detected through the same ground detection algorithm as before [176]. To accomplish this, we treat the $Y_{L2}Z_{L2}$ plane as if it were the XY ground plane. In our case, the wall corresponds to the white board marked in red in Fig. 4.1a. This configuration adds a difficulty, as ground detection algorithms treat the ground as the object that has the lowest value on the Z axis. In our case, the wall is not the furthest object away from the lidar, and so it it not - once $Y_{L2}Z_{L2}$ is treated as if it were the XY ground plane - the one with lowest Z values. To solve this issue, we take into consideration that, if we split the PC in slices along the X_{L2} axis, a wall or white board such as the one in our environment will correspond to the slice with most points. Therefore, a loop is performed cutting the lidar PC in different slices along the X_{L2} axis. The ground detection algorithm is performed on each loop iteration. The wall corresponds to the detected ground with the highest number of points. Once this surface is found (see Fig. 4.10b), the same operations executed after detecting the ground are repeated: the rotation vector $r_{L_2L_3}$ from $X_{L2}Y_{L2}Z_{L2}$ to the normal of the wall is calculated and applied to $X_{L2}Y_{L2}Z_{L2}$. The new coordinate system $X_{L3}Y_{L3}Z_{L3}$ is found, which corresponds to a better approximation

of $X_O Y_O Z_O$ than $X_{L2} Y_{L2} Z_{L2}$ was. Fig. 4.11 exemplifies this alignment, further showing its rationale. The figure displays Y_O compared to Y_{L2} and Y_{L3}.

Note that this procedure is environment-specific. In the case of an environment that only has a flat ground and no wall, the process can stop at the first refinement (however, the final localization from the lidar will not be as precise). If the environment has a wall which corresponds to the furthest element in the room, there is no need to perform a loop to search for the wall.

Tracking of the drone in the lidar point cloud

We localize the drone in the environment by exploiting the intensity information in the lidar point cloud. Due to the reflective marker on the drone, the drone tends to be the object with the highest intensity in the scene. This can be observed in Fig. 4.5b. We set a threshold Th_{Int} over the intensity. Points in the point cloud with an intensity exceeding this threshold are deemed eligible for tracking, as they could belong to the drone. However, due to the different angles at which the drone is observed from the lidar, the drone might sometimes not be the object with the highest intensity in the point cloud.

Therefore, to mitigate the impact that other (small) objects have on the localization of the drone at the first time instant, we delineate the region from which the drone initiates its movement and where we initially search for it. This can be executed automatically by finding, in the first τ instants of each flight, all the points that have an intensity over Th_{Int}. Outliers in this set are removed and all other values are used to calculate a mean position and the corresponding standard deviation.

When beginning the drone tracking, we look for it within a radius of α standard deviations from the found mean position. At the following instants, points in the PC with intensity higher than Th_{Int} are located. Only the ones at a set distance Th_{Dist} from the previous drone position are kept and averaged to obtain the new drone position.

Finally, the computed localization is smoothed. We define the final estimated positions at the output of the lidar with $\{p_t^{Lidar}\}_{t=1...T}$.

Final Alignment and Synchronization of the onboard and lidar localization

We now have the localization from both the onboard sensors and from the lidar, i.e., $\{p_t^{Onboard}\}_{t=1...T}$ and $\{p_t^{Lidar}\}_{t=1...T}$. However, before comparing them, it is necessary to complete the alignment procedure and synchronize them.

The majority of the alignment process of the point cloud has already been described in a previous section. However, one last element is missing from the alignment: whereas the two

localizations are expressed in coordinate systems with approximaly parallel axes, they don't share the same origin point. This final alignment can be done in a simple way, by subtracting an offset from the vector of lidar localizations $\{p_t^{Lidar}\}_{t=1...T}$. This offset is calculated as the mean position of the drone from the onboard sensors localization in the first τ instants of flight, when the drone is static on the ground. Only these instants are adopted for the offset calculation because the following ones could potentially have a higher error in their estimation.

Finally, the last step to perform is the synchronization of the two localizations. If the two sensors share timestamps starting from the same beginning time (for example, by being synchronized through a wireless connection), the alignment is straightforward. However, in cases where synchronized timestamps are unavailable, an approximate synchronization can be carried out. First, the onboard sensors localization is downsampled (from the 30 Hz of the video frame rate to the 20 Hz of the lidar). Then, we calculate the difference in velocities obtained by the two localization systems in a loop. During each iteration of the loop, one of the two localization systems' timestamps is incrementally shifted by a varying amount of time. Consequently, at each step of the loop, different time alignments between the two localizations are considered. We choose the alignment with the lowest difference .

4.1.4 Extracted scenarios

As we have just observed in Section 4.1.2, the drone evaluation datasets considered in this chapter are used in Chapter 8 for evaluating the localization.

However, as previously observed, another objective of these datasets is evaluating anomaly detection methods, either the ones in this thesis or different ones. Consequently, all the dataset scenarios have a set of normal trajectories and a set of abnormal trajectories.

We extract the scenarios in the list below (note that, in every scenario, the drone starts flying in front of ArUco 1).

- **LOop motion (LOM)**: the drone takes off, and then performs a loop getting close to all ArUcos from 2 to 5, and finally lands at the point where it started. In its motion, the drone always faces outwards from the center of the flying environment. In the abnormal version of this scenario, two new types of objects are placed in the environment; the first type is a yellow cylinder and the second is a brown box. When the drone observes the yellow cylinder, it performs abnormal motions such as going closer and further away from it to observe it better. In contrast, the drone does not execute any abnormal movements when the brown box enters its field of view. We extracted $13,469$ train

frames, 3,049 validation frames, and 46,503 (abnormal) testing frames (at the 30 Hz video frame rate).

- **Frontal motion (FM)**: the drone takes off facing ArUco 1, then turns on the left and moves toward the white board where ArUco 6 is attached. When reaching the board, it performs the same motion but moves backward. Three abnormal versions are extracted for this scenario. In the first version ($V1$), a pedestrian walks on the side of the drone, either in the same direction of the drone or in the opposite one; no abnormal motion from the drone is performed in this scenario. In the second version ($V2$), a pedestrian walks in front of the white board, either coming from the left of the drone or from its right; in this scenario, the drone stops to let the pedestrian pass and then continues in its motion. Finally, the last version ($V3$) considers the four combinations of the previous abnormal situations, i.e., two pedestrians are present in the scene. The drone performs the same abnormal motion as before, to let the pedestrian in front of the white board pass. In this last version, only onboard sensor data are extracted. We extracted 18,253 train frames, 2,518 validation frames, and 34,920 testing frames for the first two abnormal scenarios, and 19,732 for the third one.

- **LAteral motion (LAM)**: the drone takes off facing ArUco 1, and moves laterally, from ArUco 2 to ArUco 3. Then, it flies back and lands. In the trajectories, the drone starts at five different equispaced distances from ArUco 1, ranging from 0.9 to 2.1 meters. In the abnormal scenario, squared abnormal markers of different dimensions - $3x3$, $7x7$, $10x10$, $15x15$, $21x21$ cm - and of different colors - (0,0,0), (127, 127, 127), i.e., black and gray - are placed in the environment. No abnormal motion is performed by the drone. We extracted 17,368 train frames, 1,736 validation frames, and 20,309 (abnormal) testing frames.

Fig. 4.12 shows examples of the drone camera images when anomalies appear. In Fig. 4.12a, the yellow cylinders generating motion abnormalities in the LOM scenario are displayed. Fig. 4.12b illustrates $V1$ (above) and $V2$ (below) of the abnormal FM scenario. Finally, we display in Fig. 4.12c examples of the biggest ($21x21$ cm) and smallest ($3x3$ cm) squared abnormal markers in the LAM scenario.

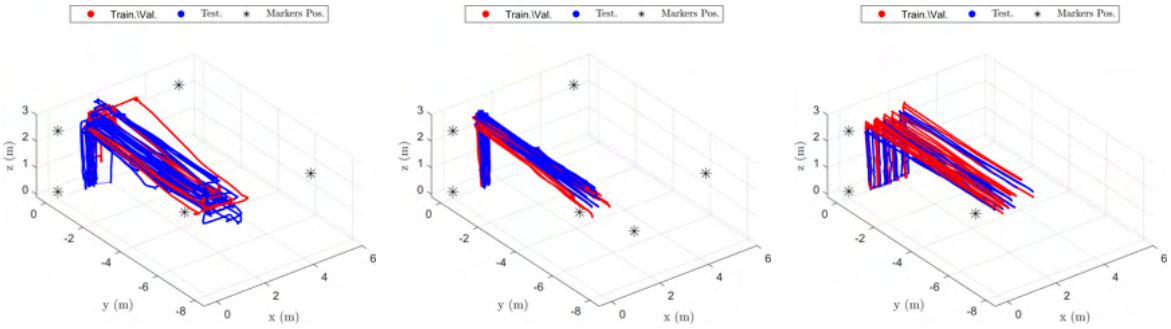
The odometry trajectories of the three scenarios are visualized in Fig. 4.13.

4.1.5 Localization results

In this Section, the results from the proposed localization method are reported. Because the lidar positioning is used as a ground truth, we visually checked and confirmed that the

(a) LOM. (b) FM. (c) LAM.

Fig. 4.12 Drone camera anomalies in the different scenarios of the drone dataset.



(a) LOM. (b) FM. (c) LAM.

Fig. 4.13 Odometry of the three scenarios of the drone dataset. The positions of the ArUco markers are displayed through black stars.

tracking of the drone was precise, by visualizing the obtained localization, instant by instant, on the lidar PC. Furthermore, we visually checked that the synchronization was accurate.

We reiterate the pros and cons of the two localizations from the onboard sensors:

- The marker detection provides an absolute but sparse estimate of the drone positioning.
- The VIO provides a continuous position, that however tends to drift over time.

These characteristics can be seen in Fig. 4.14. The blue dots correspond to the estimated position of the drone, obtained from the markers detected by the camera. They are present in three focused areas, where the markers - represented by black stars - are located. On the other hand, the continuous localization gained from the VIO is displayed in red. The lidar ground truth localization is shown in green. Comparing the lidar and VIO localization, we can discern the drift of the latter one: within the circumscribed area, the VIO estimates the

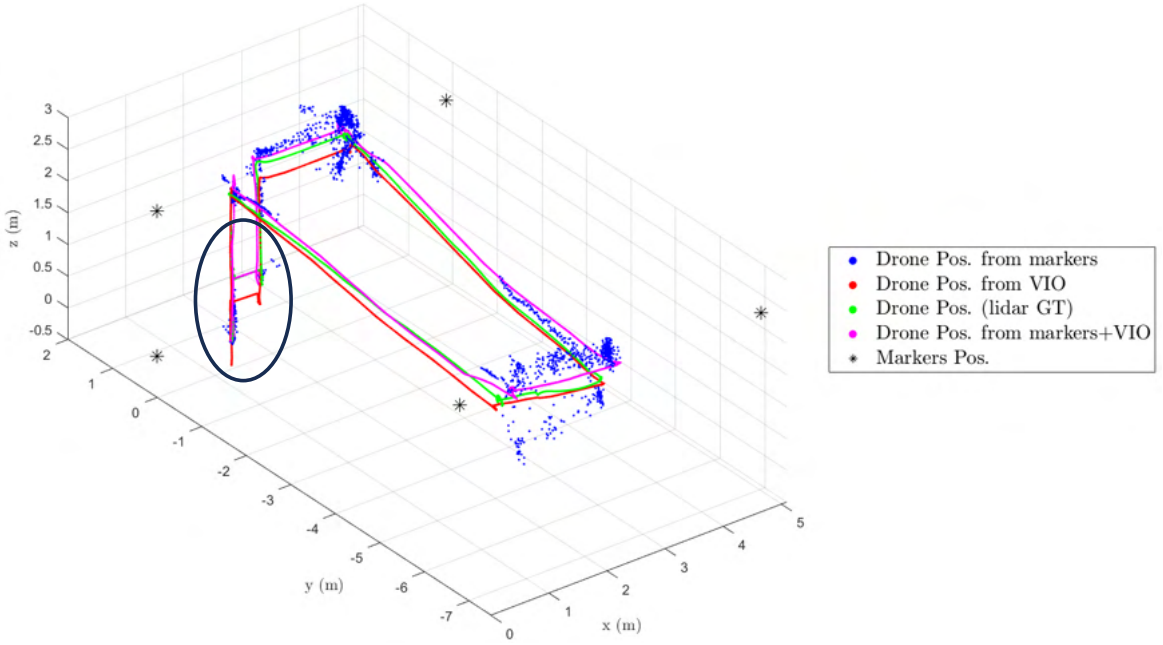


Fig. 4.14 Localization obtained from the ArUcos (blue), from the VIO (red), from the lidar (green), and joining the positioning from ArUcos and VIO (violet). The position of the five ArUco markers in this trajectory (belonging to the LOM scenario) are illustrated with black stars. A black oval highlights the final time instants of the trajectory, where the drift of the VIO is particularly evident.

position of the drone to be almost half a meter lower than what it is. Therefore, according to the VIO, in the final frames of the experiment, the drone appears to be situated below the ground plane. Conversely, the estimation obtained from the markers in that area better mirrors the true position of the drone. When the VIO and ArUco localizations are joined, a better localization is obtained than the two separate ones (see the violet plot).

Table I reports information about the three considered scenarios. From left to right, the columns illustrate the dataset scenario, the number of trajectories extracted for each scenario, the total number of frames, and the number of frames in which an ArUco marker was detected. Please note that the frame count pertains to the data following the synchronization between onboard sensors and lidar. The frame rate is now 20 Hz and not 30 Hz. For this reason, the numbers are different from the ones reported in section 4.1.4. Additionally, the drone and lidar were not started at the exact same instant, leading to one or the other capturing more time instants in the different experiments before the drone took off or after it landed. This accounts for some small additional difference in the numbers but does not pose any issues, as both drone and lidar always capture the entire trajectory from the drone's take-off to its landing.

Table I Basic information about the three drone dataset scenarios.

Data Scenario	Num. trajectories	Num. frames (at 20 Hz)	Num. frames ArUco (at 20 Hz)
LOM	18	40,714	21,797
FM	27	36,155	14,361
LAM	21	26,207	15,482

Table II Localization error using the drone onboard sensors. The localization obtained from the lidar is employed as ground truth.

Data Scenario	Mean loc. err. VIO (m)	Std. loc. err. VIO (m)	Mean loc. err. ArUco (m)	Std. loc. err. ArUco (m)	Mean loc. err. final (m)	Std. loc. err. final (m)
LOM	0.1765	0,0978	0.2352	0,1827	0.1636	0,1055
FM	0.1459	0,0791	0.1391	0,1154	0.1445	0,0882
LAM	0.1958	0,1741	0.2064	0,1608	0.1705	0,1540

Table II numerically shows the goodness of these results. The table reports the results by comparing the localization from the onboard sensors (VIO, ArUco, and VIO+ArUco) with the ground truth from the lidar. Each couple of columns displays the mean and standard deviation of the Euclidean distance between the localization predicted with the onboard sensors and the ground truth obtained with the lidar. From left to right, the table shows the results with the VIO, with the ArUco, and with the VIO combined with the ArUcos.

Notice how, in all three scenarios, the mean error is below 0.25 m. Also observed that the final continuous localization is better than the continuous localization with only the VIO.

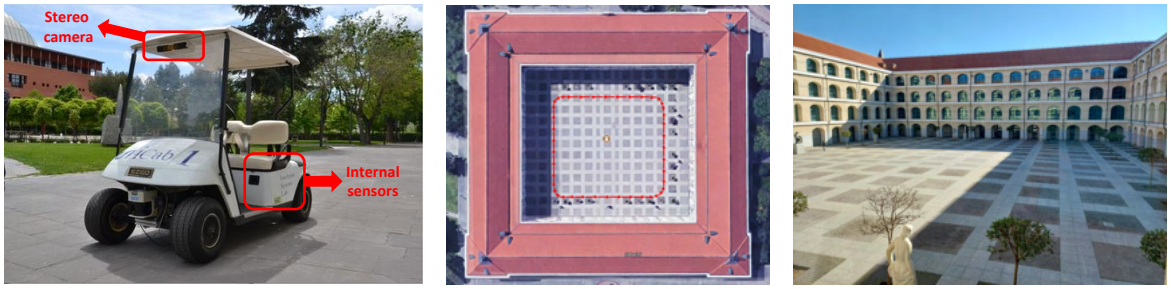
4.2 Other adopted evaluation datasets

In this section, other real-world evaluation datasets adopted in this thesis are described. Since some datasets are employed in several methodology chapters of this thesis, we chose to dedicate this section to introducing all used datasets. In this way, when encountering a dataset being mentioned in the methodology chapters, the reader can go back to this section for some basic information about it. Conversely, specific method-related analyses of these datasets will be discussed in their respective methodology chapters.

First, in Sections 4.2.1, 4.2.2 and 4.2.3, we examine three vehicular datasets provided with both camera and odometry information, i.e., the iCab dataset [157], the UAH-DriveSet

dataset [190], and the Egocart dataset [206, 207]. Then, in Section 4.2.4, we jointly introduce two fixed-camera anomaly detection benchmark datasets, i.e., the Subway dataset [1] and the Avenue dataset [151]. More space is given to those datasets that are used more or examined in deeper detail throughout the thesis.

4.2.1 ICab dataset



(a) 'iCab" vehicle (b) Environment from above. © 2021 IEEE. (c) Environment from side.

Fig. 4.15 Employed vehicle and environment of the iCab dataset.

Marin et al. adopted a real vehicle called "iCab" [157] (see Fig. 4.15a) to collect a multi-sensor dataset. A human drove the iCab performing different tasks in a closed environment displayed in Figs. 4.15b and 4.15c. The dataset was captured while the vehicle executed four different tasks:

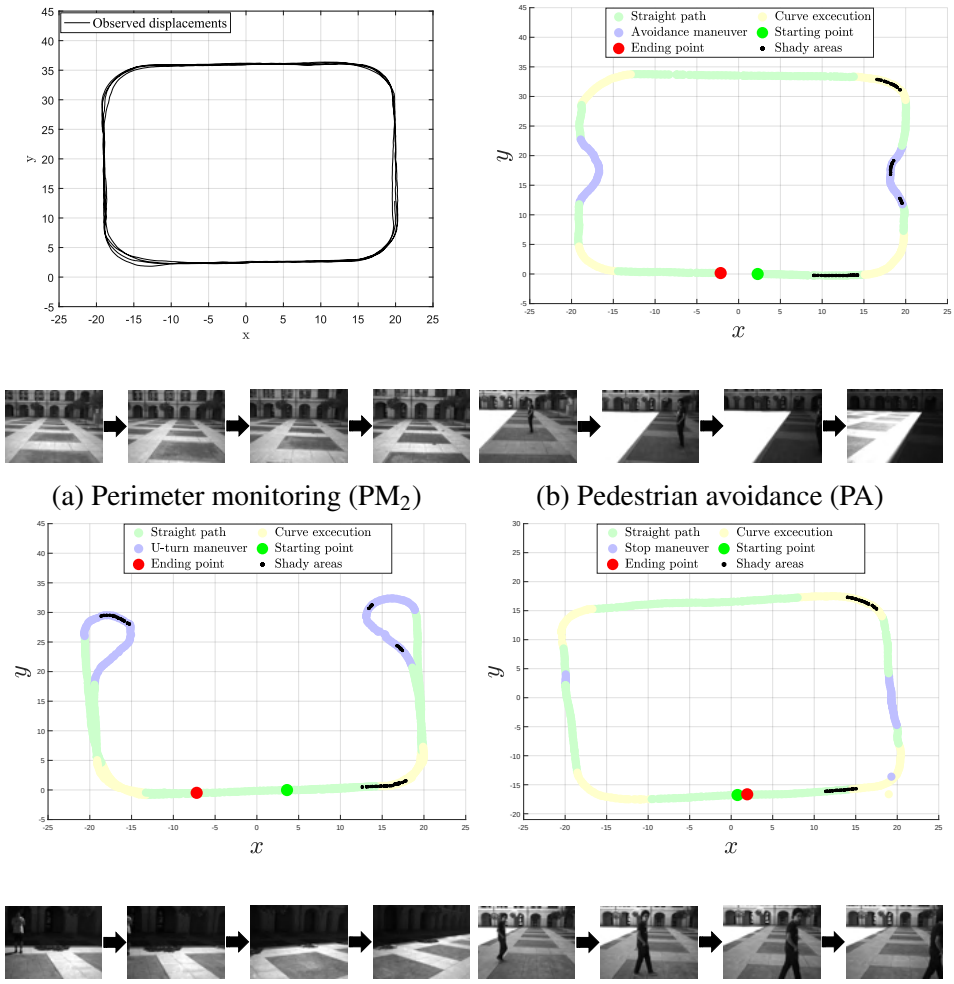
Scenario I - perimeter monitoring (PM), 13,533 frames: the vehicle follows a rectangular trajectory along a closed building.

Scenario II - emergency stop maneuver (ES), 13,724 frames: the vehicle executes the PM task, encounters two pedestrians crossing its path at each lap, performs an emergency stop, and then continues the PM task when pedestrians exit its field of view.

Scenario III - pedestrian avoidance maneuver (PA), 7,247 frames: two obstacles (stationary pedestrians) in different locations interfere with the PM task. The vehicle performs an avoidance maneuver and continues the PM task.

Scenario IV - U-turn maneuver (U-turn), 9,874 frames: while executing the PM task, the vehicle encounters two static pedestrians in different locations. It performs a U-turn motion and continues the PM in the opposite direction compared to the training data.

The temporal evolution of the four above-described maneuvers from a first-person perspective is shown in Fig. 4.16a, 4.16b, 4.16c, and 4.16d, below the odometry plotting for a single lap. The PM scenario is used for training and validation; the other scenarios are for testing. We aim to detect the abnormal maneuvers in the testing scenarios caused by the



(a) Perimeter monitoring (PM_2)

(b) Pedestrian avoidance (PA)

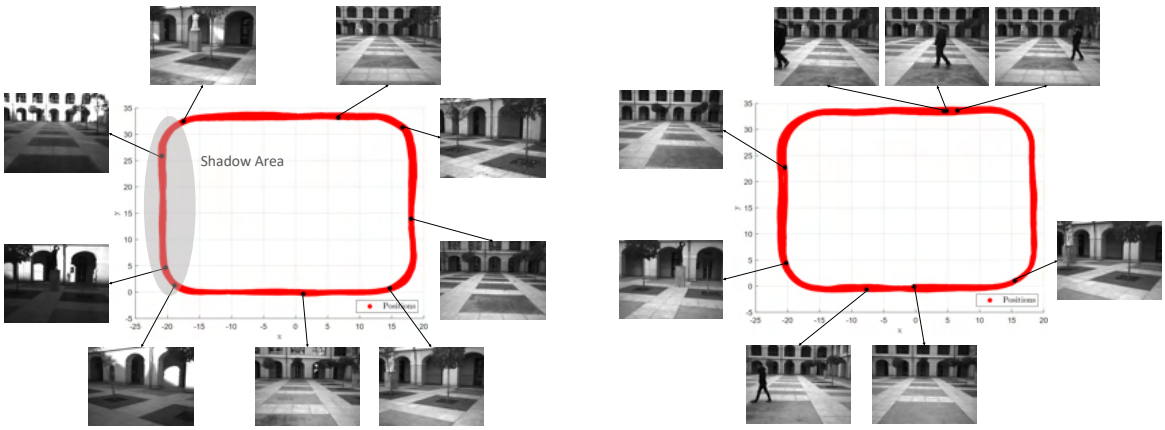
(c) U-turn

(d) Emergency stop (ES_2)

Fig. 4.16 ICab vehicle tasks used to evaluate the proposed method. Perimeter monitoring is adopted in the training phase whereas the other three tasks are employed for testing purposes. © 2021 IEEE.

presence of pedestrians. Training and testing data were captured at different hours of the day and camera orientations, resulting in differences in lighting conditions and element positions with respect to the camera. Icab images have 640x480 resolution.

The three anomaly tasks are shown in Fig. 4.16b, 4.16c, 4.16d, where the maneuvers are highlighted with different colors. Blue zones refer to the instances where the abnormal maneuver is executed; green and yellow zones correspond to linear paths and curves respectively. Shady areas are shown too, as they constitute a problem in computer vision and anomaly detection: if very dark or bright areas are present, these could be potential anomalies when the training set does not include similar dark areas. In our case, although shadows are present



(a) PM_1 (training). (b) ES_1 (testing).

Fig. 4.17 PM_1 and ES_1 sequences. Odometry information is displayed in red. Some image examples are provided. © 2023 IEEE.

in the training phase, they only cover a few local parts of the frame compared to testing. Consequently, they are noted as abnormal areas in the ground truth.

We also observe that two sets of experiments taken at different times of day and season are present for the PM case and for the ES case. A set of PM is taken at the same season (but at a different time of day) as a set of ES. We denote these sets as PM_1 (6,849 frames) and ES_1 (6,558 frames). The other two *PM* and ES experiments are performed at different times of day and season. We name them from here on PM_2 (6,684 frames) and ES_2 (7,166 frames). This latter couple is the one shown in the general Fig. 4.16. Conversely, Fig. 4.17 displays the PM_1-ES_1 experiments performed on the same day, with the odometry values in red and some image examples. In this case, a shadow area is present on one side of the training courtyard, whereas it is absent in the testing case.

This dataset is employed for evaluation in all methodology chapters of the thesis, as it presents simple, spatially and temporally contained, repeated anomalies of different types, allowing us to study how they can be detected at the different levels of the hierarchy.

4.2.2 UAH-DriveSet dataset

This is a dataset [190] by the University of Alcalá for driver behavior analysis composed of various car sensory data from six drivers performing two routes (motorway and secondary road) with three types of behaviors (normal, drowsy, and aggressive). The dataset provides GPS, accelerometer, and video data of resolution 960x540 pixels.

Fig. 4.18 displays some examples of images from this dataset, all belonging to the "secondary road" scenario.

Fig. 4.18 Image examples from UAH-DriveSet dataset, all belonging to the "secondary road" scenario.

A subpart of the "motorway road" and "secondary road" scenario is used in the evaluation of the approaches proposed in Chapters 6 and 7, respectively.

4.2.3 Egocart dataset

The Egocart dataset [206, 207] is a benchmark dataset for indoor video localization, captured with a shopping cart moving in an empty retail store. It is composed of a set of training and testing image trajectories and their corresponding positions. The training set has $13,360$ frames, and the testing set has $6,171$. Egocart images have $1280x720$ resolution. The ground truth position is obtained using Structure from Motion techniques [228].

Spera et al. [206, 207] evaluated several state-of-the-art localization methods on this dataset.

Fig. 4.19 displays some example frames from this dataset, and the training and testing odometry data.

Note that this dataset is a benchmark localization dataset, and not an anomaly detection one. It is employed in Chapter 8 of this thesis to test the localization capabilities of the proposed method. However, some comments regarding the detection of anomalies are proposed too.

(a) (b) (c)

Fig. 4.19 Training and testing Egocart odometry (a), and some examples of images of the Egocart dataset.

(a) (b) (c) (d)

Fig. 4.20 Examples of normal and abnormal images of the subway dataset: (a) subway entrance normal image; (b) subway exit normal image; (c-d) subway exit abnormal images: in (c) a person enters the platform from the wrong direction, whereas in (d) he exits the platform when there is no train in the subway.

(a) (b) (c)

Fig. 4.21 Normal (a) and abnormal (b-c) frames in the Avenue dataset. In the abnormal frames, a person throws paper in the air (b), and someone walks on the lawn (c). Note: the illustrated red box does not correspond to the ground truth provided by the authors of the dataset, but was plotted in this figure just to approximately highlight where some examples of anomalies are located.

4.2.4 Fixed-camera datasets: Subway and Avenue

All datasets considered until now were extracted from a moving vehicle. In this Section, instead, we introduce two real-world datasets from static cameras. In place of being mounted on a vehicle, the camera in these datasets records the scene from an overhead vantage point. Both datasets presented in this section are anomaly detection benchmark datasets.

The Subway Dataset [1] : this dataset contains two sets of video streams from surveillance cameras positioned at the entrance and the exit of a subway, with people moving in the corresponding flow direction. The Entrance dataset is composed of $144,249$ frames (1 hour and 36 minutes) and the Exit dataset contains $64,900$ frames (43 minutes). In this thesis, the first 18 and 5 minutes, respectively, were used for training. Abnormal events correspond to people moving in wrong directions, loitering, etc..

The Avenue Dataset [151] : this dataset contains videos from a surveillance camera positioned in front of a campus avenue, with $15,328$ training and $15,324$ testing frames

(16 normal and 21 abnormal videos). Abnormal events correspond to people walking in unexpected zones, throwing paper, running, etc. Some examples are shown in Fig. 4.21.

The Subway and Avenue datasets are employed to evaluate the approach proposed in Chapter 5 only. The subsequent methodology chapters necessitate positional data of the moving vehicle, which is incompatible with these static camera datasets.

4.3 Data from the Carla simulator (CarlaABN)

In addition to the real-world datasets described above, we use data from the Carla simulator, an open-source simulator for autonomous driving research [53].

We automatically generate 50 trajectories of a car randomly moving in a town (Town 2), extracting a total of $22,776$ frames. These frames are subdivided into $17,138$ training and $5,638$ validation frames. The resolution of the frames is $288x162$. Then, we manually extract a set of abnormal scenarios:

- A Sudden Deceleration (SD), 50 frames: the car travels at normal speed along a road traversed during training, and suddenly decelerates, a movement that was not performed during training. This is a motion anomaly.
- An Out-Of-Street motion (OOS), 40 frames: the car crosses the center line of a known road, and starts driving on the wrong side of the road. This is both a visual anomaly (unseen areas of the map are explored) and a motion anomaly (due to the crossing of the center line and moving in the wrong direction).
- An Abnormal Turn at an Intersection (ATI), 70 frames: during training, the car always turned left at a specific intersection. Conversely, during the ATI scenario, it moves straight at the same intersection. This is a motion anomaly and an anomaly at a higher level of the DBN hierarchy, as the car moves from the known zone 1 (before the intersection) to the unexpected known zone 2 (straight after the intersection) instead of moving to the expected known zone 3 (at the left after the intersection). See Fig. 4.22 for an approximate representation of the zones 1, 2, and 3.

Fig. 4.22 displays the extracted training, validation (red), and testing (blue) trajectories. The three scenarios are marked with the letters “a” (SD), “b” (OOS), and “c” (ATI). A frame example for each scenario is displayed in Fig. 4.23; the same letters are adopted for the scenarios as in Fig. 4.22.

From here on, we denominate this dataset CarlaABN, so that it can be easily referenced in the thesis.

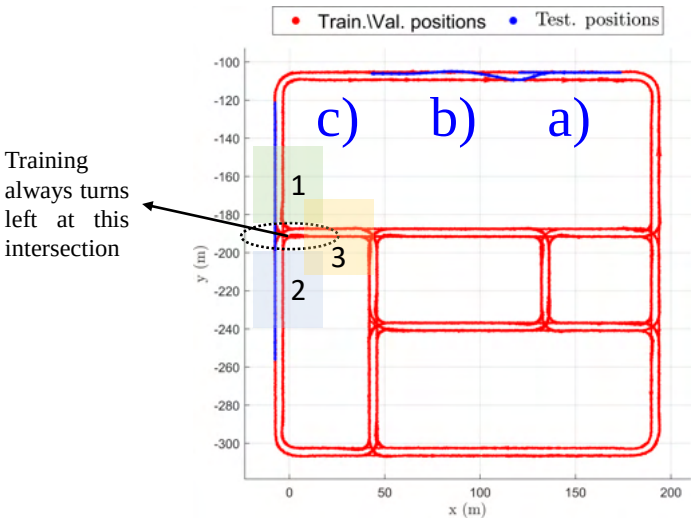


Fig. 4.22 Training, validation and testing positions in the CarlaABN dataset.

(a) (b) (c)

Fig. 4.23 Frames of the CarlaABN dataset belonging to the three abnormal situations. (a): SD; (b): OOS, after crossing the center line; (c): ATI, before crossing the intersection.

The advantage of using the simulator is that it allows us to easily extract precise scenarios with simple, known, spatially and temporally contained anomalies. For each of the above-described scenarios, we know at which level of the DBN hierarchy to expect high anomaly signals. Therefore, we can test our approaches to verify at which level of the DBN anomalies are detected.

The data extracted with the Carla simulator is adopted for evaluation in Chapter 8.

4.4 Datasets summary

Table III summarizes the main characteristics of the datasets explored in this chapter.

In the first three columns, the table defines the available data types, the data types employed in the self-awareness methods presented in this thesis, and the data types used to build the positional ground truth. Some datasets have a high variety of provided information,

Table III Summary of the main characteristics of the datasets employed in this thesis. The term "abs. pos." stands for absolute positioning.

	Data types available	Data types used for the self-awareness methods in this thesis	Sensors used to build the positional GT	Fixed (F) or moving (M) camera	Real-world (RW) or simulated (S) data
Drone (Section 4.1)	Camera, abs. pos., IMU readings, lidar PCs, battery level, etc.	Camera, abs. pos. (GPS)	Lidar	M	RW
ICab (Section 4.2.1)	Camera, abs. pos., IMU readings, etc.	Camera, abs. pos. (GPS)	GPS	M	RW
UAH-DriveSet (Section 4.2.2)	Camera, abs. pos., IMU readings, etc.	Camera, abs. pos. (GPS+IMU)	GPS+IMU	M	RW
Egocart (Section 4.2.3)	Camera, abs. pos.	Camera, abs. pos. (SfM)	Camera (SfM)	M	RW
Avenue (Section 4.2.4)	Camera	Camera	-	F	RW
Subway (Section 4.2.4)	Camera	Camera	-	F	RW
CarlaABN (Section 4.3)	Camera, abs. pos.	Camera, abs. pos. (GPS)	Actual pos. from simulator	M	S

so we only highlight a subpart of the data. For example, the employed DJI FLY 2S drone provided a great amount of raw and elaborated data that was behind the scope of this thesis, such as battery level, temperature, and wind speed. Similarly, the UAH-DriveSet dataset includes a lot of processed information, such as detected lanes, vehicles, and lane-changing events. It is also worth noting that this table summarizes the available data after preprocessing but before applying the self-awareness models described in the next chapters.

The positional information is extracted in the different datasets with a variety of sensors. In the drone dataset, we extracted it with the VIO, with the VIO and ArUcos, and with the lidar. The positioning from the VIO and ArUcos is employed to train the self-awareness

models, whereas the positioning from the lidar is used as ground truth. This is the only real-world dataset case in which the odometry used as training input and as ground truth are different, for the reasons explored in Section 4.1.3. In the iCab dataset, the positioning is available through the GPS and the wheel odometer. The GPS positioning is employed in this thesis. In the UAH-DriveSet dataset, the GPS position has a sampling rate of 1 Hz and the IMU readings of 10 Hz. In this thesis, they are combined to obtain a positioning with a sampling rate of 10 Hz. The Egocart positional data is extracted by the original authors [207] through Structure from Motion (Sfm) techniques. Finally, in the CarlaABN dataset, a simulated GPS sensor is adopted, and the real vehicle position in the simulator corresponds to the ground truth.

Finally, the last two columns of the table define if the dataset camera was fixed or moving, and if the data is real-world or simulated.

4.5 Conclusions

The objective of this chapter was twofold: *i)* to describe the extraction of an indoor drone dataset for anomaly detection and localization, while also proposing a method for localizing the drone from onboard and external sensors; *ii)* to introduce all the other evaluation datasets adopted in this thesis, which include both real-world and simulated data.

In the first part, we proposed a traditional Signal Processing method to localize a door flying in an indoor environment from its onboard sensors (camera and IMU) and from an external sensor (lidar). The onboard localization was performed by combining a sparse positioning obtained from detecting ArUco markers in the environment with a continuous positioning gained from the VIO. The fusion was performed through a Kalman Filter. On the other hand, lidar localization was made possible by tracking the drone in the lidar PC, leveraging its high reflectivity. The lidar localization is aligned with the onboard localization by identifying the plane of the ground and the plane of the wall inside the environment.

Moreover, in Section 4.1.2, we explain how the extracted drone dataset will be employed for the evaluation of the self-awareness approaches proposed in this thesis.

In the second part of this chapter, we have introduced all the evaluation datasets that are employed in this thesis. These include three real-world First-Person Viewpoint (FPV) datasets, endowed with both camera and odometry information (iCab, UAH-DriveSet, and Egocart), two real-world fixed camera datasets (Subway and Avenue), and one FPV simulated dataset extracted using the Carla simulator. Except for the UAH-DriveSet and Egocart datasets, all these datasets are specifically tailored for anomaly detection. Additionally, we have noticed that certain datasets, including iCab and CarlaABN, feature spatially and

temporally isolated abnormal scenarios. In these instances, we can readily pinpoint the specific abnormal signals that are expected to register high within the DBN hierarchy.

Whereas in this chapter all the datasets are introduced, a more detailed analysis of some of them will be found in the methodology chapters. This detailed analysis is related to aspects of the proposed self-awareness methods and, therefore, would be out of place in this introductory chapter.

Consequently, this chapter still does not describe or employ any of the proposed self-awareness approaches central to this thesis. In the next chapters, the self-awareness methods of this thesis are presented.

Chapter 5

Multilevel anomaly detection Through Variational Autoencoders and Bayesian Models for self-aware Embodied Agents

This is the first chapter of this thesis proposing a self-awareness approach based on the framework discussed in Chapter 3. We tackle the anomaly detection application, and we only employ camera data. As already discussed in Chapter 3, anomaly detection constitutes a fundamental step in developing self-aware autonomous agents capable of continuously learning from new situations, as it enables to distinguish novel experiences from already encountered ones.

This chapter adopts DBNs and NNs to perform anomaly detection in video data across multiple levels of abstraction. Two methods are presented and compared. Both exploit VAEs to reduce the dimensionality of camera data and employ an extension of the MJPF. However, whereas the first method trains the VAE on camera frames only and uses the innovation of a Null Force Filter (NFF) to build the GS, the second method trains the VAE on camera frames and Optical Flows (OFs) between consequent images.

Section 5.1 introduces the discussed problem and the motivation behind the proposed approach, also given the framework discussed in Chapter 3. Then, the two versions of the proposed approach are described in detail in Sections 5.2 and 5.3, respectively. We also outline the advantages of the proposed approach (Section 5.4) and a general comparison between the two versions with their advantages and disadvantages (Section 5.5). As the adopted framework aims at performing incremental learning, we introduce how the extracted anomalies can be exploited for this purpose (Section 5.6). Afterward, Section 5.7 discusses results, further comparing the two approaches. Finally, Section 5.8 draws some conclusions.

5.1 Introduction

As examined in Section 3.3, the automatic detection of anomalies in video information is a key element for generating robust autonomous systems that can adapt themselves to unknown situations and incrementally learn predictive models from them.

Indeed, six basic capabilities of self-aware systems have been distinguished: initialization, memorization, prediction, anomaly detection, new model creation, and decision-making. Anomaly detection constitutes the fourth capability, and supports the recognition of novel situations, triggering the creation of new models. The proposed method is based on hierarchical probabilistic models that facilitate inferring future instances of video sequences and detecting anomalies, encompassing the capabilities from *initialization* to *anomaly detection*.

Self-awareness is an important feature that autonomous systems should own to emulate human-like behavior and reasoning. As already pointed out in Chapter 3, recent research [183] has leveraged bio-inspired brain theories, such as the ones of Friston [77], that define a probabilistic hierarchical framework to endow artificial agents with awareness of their surroundings and of their own state. In particular, Friston [77] proposes the use of Bayesian dynamical filters to develop artificial agents that execute processes similar to the ones the brain performs when presented with known and unknown situations. The use of this type of filter allows us to model uncertainty, which is inherent in the representation of the world and in the prediction of how it will change. These filters also enable the construction of a hierarchical representation of the world through conditionally connected random variables that can be linked across time and modality (i.e., proprioceptive and exteroceptive variables). In particular, the Bayesian filters suggested by Friston are Generalized State Bayesian filters, which represent the state through generalized coordinates of motion (see Section 3.2.2).

Recent research has started to introduce brain-based deductions into the aforementioned probabilistic hierarchical models, along with the six basic capabilities of self-aware systems. In particular, as mentioned in Section 3.4.2, the anomaly detection step has been applied on low dimensional exteroceptive [17] and proprioceptive [115] data. In [180], a method for developing self-aware models on video data was proposed using GANs and it covered the first four self-awareness capabilities. A GAN [85] is a state-of-the-art method for handling image data and has been shown to obtain impressive results for different applications such as image inpainting, segmentation, super-resolution, and anomaly detection. However, GANs do not offer an explicit probabilistic representation of the data. Conversely, despite giving a blurrier reconstruction of the data, VAEs (see Section 2.6) allow an explicit probabilistic representation of images, and can therefore be easily incorporated into a Bayesian framework.

The capability of the VAE to explicitly learn the data distribution allows us to find anomalies at the different abstraction levels of the DBN. These anomalies can be calculated

using probabilistic distances between the learned data distributions, as seen in Section 3.7.4. The message-passing capabilities of hierarchical Bayesian models can consequently be exploited. Anomalies using probabilistic distances such as Kullback-Leibler divergence and Bhattacharyya distance can be defined and can be applied coherently at the different abstraction levels.

Therefore, the work in [180] lacked a probabilistic nature and kept the treatment of data at a high-dimensional level. DBN message-passing capabilities could not be properly exploited and the interpretability of the model was limited. We remind the reader that interpretability is one of the five desirable characteristics for self-awareness systems, identified in Section 3.3. Additionally, we desire for the architecture to be multi-sensorial, and to handle high levels of the hierarchy homogeneously between different sensory modalities. This is not possible with GANs, which do not provide a probabilistic high-level representation.

Instead, the method proposed in this chapter reduces high-dimensional video data to low dimensionality, allowing at the same time probabilistic reasoning. In this way, two worlds are combined: the one of DBNs and the one of NNs. The presented method is based on the latent representation of a VAE, which is employed to obtain a low-dimensional state of the video at each time instant in a probabilistic fashion. Using the latent state values, clusters of similar image appearance and motion are identified, facilitating a semantic representation of video data.

In this chapter, we present two alternative methods for building the part of the state related to the image motion and, consequently, for dividing the information into clusters. The first method trains the VAE on camera frames only and computes the innovation of an NFF on the VAE latent states. This innovation constitutes the part of the GS accounting for the image motion information. On the other hand, the second method extracts the OF between consequent video frames, and trains the VAE on both OF and camera frames; the part of the latent state corresponding to the OF captures information related to the agent's motion.

In each cluster, a prediction model consisting of a fully connected NN is employed to learn a nonlinear dynamical model that allows estimating a future image given the current one and its motion. These NNs encode the dynamical rules that emerge from the data. Accordingly, two representation levels, i.e., discrete and continuous, are learned to make inferences in video sequences and detect anomalies. During the online testing phase, a PF (see Section 2.4.4) coupled with a set of UKFs [221] (see Section 2.4.3) are employed for predicting at the discrete and continuous levels, respectively, and for detecting anomalies. By combining the VAE, PF, KF (for the update) and UKF (for the prediction) in the two versions of the proposed approach, two filters inspired by the original MJPF [17] are presented. Both image-level, state-level, and discrete-level anomalies are calculated.

To summarize, this chapter aims to develop a self-awareness approach for video anomaly detection based on the framework presented in Chapter 3. A GDBN generative model is learned, with a multi-level representation employing generalized coordinates of motion.

The novel contributions of this chapter are:

- A multi-level probabilistic model that represents video sequences into latent states, facilitating the prediction at continuous and discrete hierarchical levels. This is performed through two novel ways of combining VAE, KFs (update equations), UKFs (prediction equations), and PF, and a set of NNs that non-linearly models the dynamics of latent space information, thus encoding the dynamical rules emergent from the data.
- An approach that is compatible with low and high-dimensional data, potentially allowing the detection of anomalies also in a multi-sensory framework with homogeneous handling of the high levels of the DBN hierarchy.
- The detection of new situations that autonomous systems may employ to continuously learn predictive models without forgetting previously learned experiences.
- A comparison between the two methods for building the GS and for joining the VAE with the KF/UKFs. Two filters based on the MJPF are proposed. We define the first filter as Adapted MJPF (AMJPF), and the second as OF Adapted MJPF (OF-AMJPF). Both filters contain, as a submodule, a UKF modified to work with a VAE as an observation model, and with an NN as a prediction model. This submodule is slightly different in the two cases. In the first method, we name it Adapted KF (AKF) and, in the second method, we name it OF Adapted KF (OF-AKF). We ascertain that the incorporation of the OF enhances the anomaly detection performance, albeit increasing the computation time and introducing a model-based dimension into the predominantly data-driven framework, as we will discuss in the later sections.

5.2 First Version of the Proposed Approach

We start by describing in detail the first version of the approach.

The proposed method is composed of two basic steps, namely *training* and *testing*; during the former (Section 5.2.1 and Fig. 5.1), algorithms are trained based on observed data; whereas the latter (Section 5.2.2 and Fig. 5.2) uses learned algorithms to detect anomalies on new data.

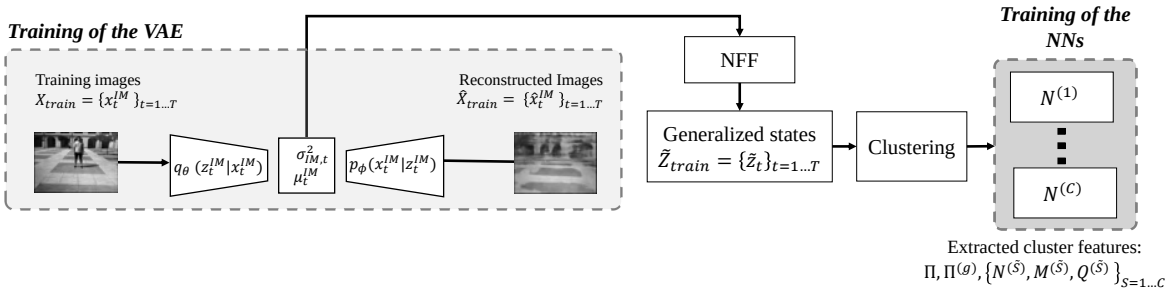


Fig. 5.1 Training phase for the first version of the proposed approach. The VAE is trained to reconstruct a set of training images. The latent states (i.e., the bottleneck features) are then extracted and the GSs are derived and used for clustering. An NN is trained for each cluster.

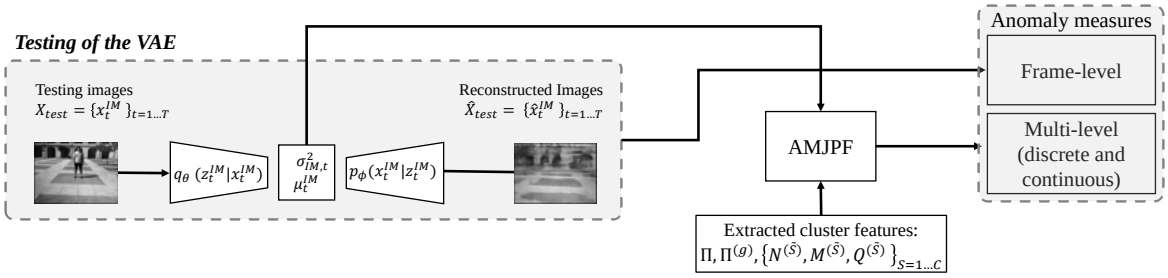


Fig. 5.2 Testing phase for the second version of the proposed approach. The encoder of the VAE is used to extract the latent state corresponding to each testing image. The latent state is given as input to the AMJPF, which detects anomalies by using cluster information.

5.2.1 Training phase

Variational Autoencoder

As a first step of the training phase, a VAE is used for describing the images in a latent space that has significantly reduced dimension compared to the original image size. A VAE has been used instead of a normal Autoencoder because it facilitates representing images in the latent state probabilistically by using a mean μ and variance σ^2 to approximate each latent variable. This enables probabilistic reasoning and inference, and allows us to detect anomalies at the latent feature level. Additionally, as we are interested in producing predictive models that can work for multisensory data regardless of their dimensionality, the VAE turns out to be an excellent choice for representing and treating video sequences as small-dimensional data, potentially enabling a more homogeneous way of making algorithms for data fusion with multimodal information.

We first train the VAE by using a set of training images X^{IM}_{train}. The images are given as input to a VAE through the encoder $q^{IM}_{\theta}(z^{IM}_t|x^{IM}_t)$. The VAE is then trained to reconstruct the images through the decoder $p_{\phi}(x^{IM}_t|z^{IM}_t)$.

Consequently, we input again X^{IM}_{train} to the VAE and obtain a set of latent features described by μ^{IM}_{train} and $\sigma^2_{IM,train}$.

Let μ^{IM}_t be the value of μ^{IM} for the image x^{IM}_t at time t. Consistent with the conventional notation in VAE literature [122], we denote as $\sigma^2_{IM,t}$ the variance vector that the VAE associates to μ^{IM}_t, and we denote as z^{IM}_t a sample from the Gaussian distribution $\mathcal{N}(\mu^{IM}_t, \sigma^2_{IM,t})$.

The notation with the addition of the "IM" index might seem a bit complex, but we adopt it to distinguish the image observation and latent state from the OF observation and latent state that will be employed in the second method.

Null Force Filter and Generalized states

The framework underlying this thesis adopts concepts inspired by Friston, such as the use of Generalized States. As discussed in Section 3.2.2, a set of GSs comprises information about the states per se and about their higher-order dynamical features, i.e., their motion, velocity, acceleration, etc.

Starting from the set of μ^{IM}_{train}, considered as the state of training images, we could build a set of GSs containing several time-order derivatives. This work only uses the first-order derivative since no abrupt dynamics are considered.

An NFF is applied on μ^{IM}_{train}. The GS is the extracted GE on the filter, as described in Section 3.7.1. We define the GS at time t as:

$$\tilde{z}_t = [\mu^{IM}_t \;\; \dot{\mu}^{IM}_t]^{\intercal} \tag{5.1}$$

Note that $\dot{\mu}^{IM}_t$ approximates the first-order derivative of μ^{IM}_t.

We obtain a set of GSs for the training set, defined by:

$$\tilde{Z}_{train} = [\mu^{IM}_{train} \;\; \dot{\mu}^{IM}_{train}]^{\intercal} \tag{5.2}$$

Clustering and neural networks

After obtaining the GSs related to the training video sequences, we use a Growing Neural Gas [78] clustering algorithm to cluster GSs into groups that carry similar information. Since we use the values of μ^{IM}_t and $\dot{\mu}^{IM}_t$ to perform the clustering process, obtained clusters take into consideration the encoded image and also its dynamics compared to the next frame. This facilitates recognizing and clustering different ways of moving. However, it is also worth

noting that this separation of different motion types is very limited because the VAE's latent state can be complex in the way it changes. Therefore, similar latent space differences do not necessarily correspond to similar velocities.

For each cluster $\tilde{S}$, with $\tilde{S} = 1 \dots C$, the following features are extracted:

- the cluster centroid $M^{(\tilde{S})}$, i.e., the mean of the cluster;
- the cluster covariance $Q^{(\tilde{S})}$;
- a transition matrix Π defining the overall probabilities of moving from one cluster to the others;
- a set of Temporal Transition Matrices (TTMs) $\Pi^{(g)}$, containing the probability of moving from one cluster to the other ones, given that g time instants have been spent in the current cluster, where $g = 2 \dots G$, with G being the maximum time spent in a cluster.
- a fully connected neural network $N^{(\tilde{S})}$ defining the dynamics of the GSs is learned for each cluster. This is the continuous predictive model.

Each of these elements was already present in the original architecture for odometry data presented in [17], based on the framework examined in Section 3.7. The novelty in this learned vocabulary is the addition of the NNs, necessary to make an accurate prediction on the type of GS obtained in this chapter, which is an output of the VAE, and is less smooth and more complex than odometry information.

For training each $N^{(\tilde{S})}$, the value of each μ_t^{IM} is taken as input and the corresponding $\dot{\mu}_{t+1}^{IM}$ as output, where $[\mu_t^{IM}, \dot{\mu}_t^{IM}]^\intercal \in \tilde{S}$. Moreover, to include the uncertainty of the Gaussian latent spaces encoded in σ^2, $2L$ additional inputs and outputs are used, where L is the dimension of the VAE's latent state μ_t^{IM}. Such $2L$ points, together with the initial mean $\mu_t^{IM,0} = \mu_t^{IM}$, permit to completely capture and define the Gaussian $\mathcal{N}(\mu_t^{IM}, \sigma_{IM,t}^2)$, as described in the UKF's paper [221] and in Section 2.4.3:

$$\begin{aligned} \mu_{\mathrm{t}}{}^{IM,i} &= \mu_{\mathrm{t}}{}^{IM} + (\sqrt{(L+\lambda)\Sigma_t^{IM}})_{\mathrm{i}} \qquad \textbf{if} \;\; i = 1 \dots L \\ \mu_{\mathrm{t}}{}^{IM,i} &= \mu_{\mathrm{t}}{}^{IM} - (\sqrt{(L+\lambda)\Sigma_t^{IM}})_{\mathrm{i-L}} \quad \textbf{if} \;\; i = L+1 \dots 2L, \end{aligned} \tag{5.3}$$

where λ is a scaling parameter and $\Sigma_t^{IM} \sim I_L \sigma_{IM,t}^2$, being I_L the identity matrix of dimension L.

The $\mu_t^{IM,i}$ values calculated in Eq. (5.3) are the so-called sigma points associated with (μ_t^{IM}, $\sigma_{IM,t}^2$). A corresponding group of sigma points is defined in the same way for (μ_{t+1}^{IM},

$\sigma^2_{IM,t+1}$) and each value of $\mu^{IM,i}_{t+1} - \mu^{IM,i}_t$ is given as additional output for the training of the NNs.

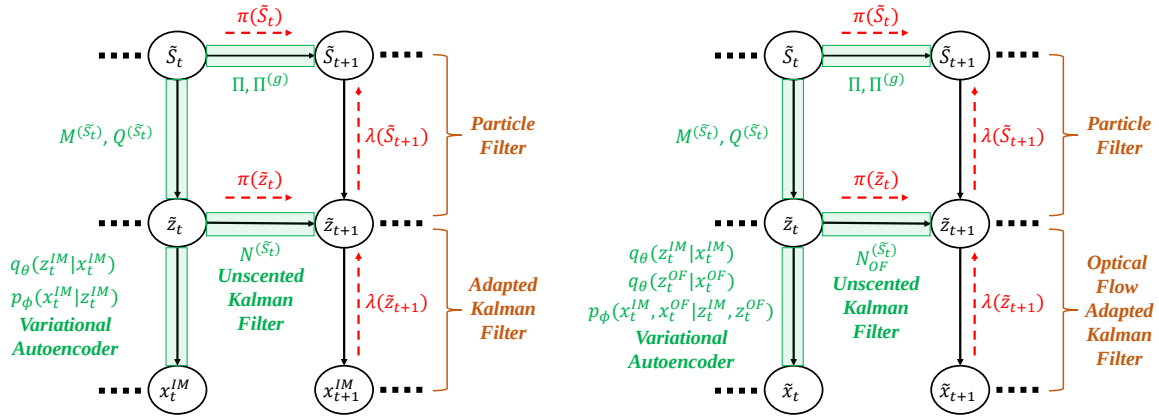
To summarize, each $N^{(\tilde{S})}$ performs the following approximation:

$$\dot{\mu}^{IM,i}_{t+1} \sim N^{(\tilde{S})}(\mu^{IM,i}_t) + w^i_t, \tag{5.4}$$

where $\dot{\mu}^{IM,i}_{t+1}$ and $\mu^{IM,i}_t$ are calculated based on the $[\mu^{IM}_t, \dot{\mu}_t{}^{IM}]^\intercal$ assigned to the considered cluster and w^i_t is the residual error after the network convergence.

Each $N^{(\tilde{S})}$ learns a sort of *quasi-semantic* information based on a particular image appearance and motion detected by the cluster $\tilde{S}$, facilitating the estimation of future latent spaces, i.e., predicting following frames. The predictions can be employed to detect whether new observations are similar to previously learned situations encoded in the set of NNs. If the predictions from the NNs are not compliant with the observations, an anomaly is detected, and models should be adapted to learn new situations, thereby generating new semantic information.

Learned Dynamic Bayesian Network



(a) DBN of AMJPF. (b) DBN of OF-AMJPF.

Fig. 5.3 DBNs of the AMJPF and of the OF-AMJPF. It can be noticed how, in case (a), only the image x^{IM}_t is employed as observation to the filter, and the VAE is trained to reconstruct it. In case (b), the Generalized Observation $\tilde{x}_t$, composed of image and OF, is used as observation; the VAE is trained to reconstruct both images and OFs.

Fig. 5.3a shows the DBN learned in this version of the approach. It can be noticed how this is a three-level DBN as the one discussed in Sections 2.3 and 3.7.

Section 3.7 described the general framework underlying this thesis, and examined how the observation model of the DBN could be either built with an observation matrix H (in the

low-dimensional case) or with a Deep Learning generative model (in the high-dimensional case). In this chapter, we implement this idea by adopting a VAE, which allows us to obtain z_t^{IM} from x_t^{IM} through the encoder $q_\theta(z_t^{IM}|x_t^{IM})$, and, to obtain x_t^{IM} from z_t^{IM} through the decoder $p_\phi(x_t^{IM}|z_t^{IM})$.

Furthermore, in Section 3.7, we noticed that the link $P(\tilde{z}_{t+1}|\tilde{z}_t)$ could either be represented by linear or by non-linear models. In this chapter, the non-linear models $N^{(\tilde{S}_t)}$ are employed. We will see in the next section how UKFs can be exploited in these models.

The other two links displayed in green in the figure are learned in the same way for high-dimensional and low-dimensional data. As seen in Section 3.7, the cluster mean $M^{(\tilde{S})}$ and covariance $Q^{(\tilde{S})}$ represent the link $P(\tilde{z}_t|\tilde{S}_t)$, and the transition matrices represent the link $P(\tilde{S}_{t+1}|\tilde{S}_t)$. Therefore, on the one hand, low levels of the hierarchy are not homogeneous across sensory modalities due to the diverse characteristics of the sensors. We saw in Section 3.7, that for low-dimensional data, linear prediction and observation models are built, using the matrices $U^{(\tilde{S})}$, and H. In contrast, in this chapter, for high-dimensional data, non-linear models are employed. On the other hand, as desired, high levels of the hierarchy are expressed homogeneously in the two cases.

5.2.2 Testing Phase

During the testing phase, each testing image is processed by the VAE. Then, a filter that we denote as AMJPF is used to perform future frame prediction and to detect anomalies in video sequences.

Adapted Markov Jump Particle Filter

As discussed in Section 2.4.6, an MJPF can be represented as a Probabilistic Switching Graphical Model [127, 209]. It was enhanced in [17] with anomaly detection capabilities, according to the self-awareness framework described in Chapter 3. The MJPF uses a set of KFs at the continuous state level and a PF at the discrete state level [17].

This chapter tackles a problem that requires a non-linear model for prediction purposes and a non-linear observation model, solved by a set of NNs (each of them associated with a detected cluster) and by a VAE, respectively. Accordingly, a set of standard KFs at the state level do not provide a good solution due to the aforementioned non-linearities. This work uses a set of modified KFs with predictions following the same logic of the UKFs and employs the encoded information of the VAE for updating purposes.

An overview of the MJPF can be found in Section 2.4.6 and a more detailed description of how it is incorporated in the adopted self-awareness framework is proposed in Section 3.7.

This section, instead, focuses on the parts that have been modified in the AMJPF compared to the standard MJPF.

For both architectures (MPJF and AMPJF), at each time instant t, two stages are performed: *prediction* and *update*. During prediction, the next cluster $\tilde{S}_{t+1}$ (discrete level) and the next GS $\tilde{z}_{t+1}$ (continuous level) are estimated for each particle, i.e., $p(\tilde{S}_{t+1}|\tilde{S}_t)$ and $p(\tilde{z}_{t+1}|\tilde{z}_t)$, respectively. Similarly to the standard MPJF, predictions at the discrete level in the AMJPF are performed by using, for each particle, the transition matrices Π and $\Pi^{(g)}$. On the other hand, predictions at the continuous level in the AMPJF are performed by the neural network $N^{(\tilde{S}_t)}$ associated with the selected discrete state $\tilde{S}_t$. Since non-linear models are considered for predicting continuous-level information, a UKF performs the estimations by taking $2L$ additional sigma points as in Eq. 5.3. Therefore, the prediction for each sigma point follows the equation below:

$$\tilde{z}^i{}_{\mathrm{t+1}} = f(\tilde{z}^i{}_{\mathrm{t}}) = A\tilde{z}^i{}_{\mathrm{t}} + BN^{(\tilde{\mathrm{S}})}(\mu_t^{IM,i}) + w^i{}_{\mathrm{t}}, \tag{5.5}$$

where A and B are two matrices used to map the previous state $\tilde{z}_t^i$ and the new velocity computed by $N^{(\tilde{S})}(\mu_t^{IM,i})$ on the new state $\tilde{z}_{t+1}^i$. Here, $A = [A_1 A_2]$ with $A_1 = [I_L 0_{L,L}]^\intercal$, $A_2 = 0_{2L,L}$, and $B = [I_L I_L]^\intercal$. The mean and covariance of the predicted state are then calculated using the UKF formulas for the propagation of a Gaussian random variable through a non-linear model.

Note that, when the state is at the border of the clusters, some particles predict to remain in the original cluster, while others predict to move to the following one. Consequently, the NNs of both clusters are used in their respective particles to perform state-level prediction. This enables reasonable prediction at the borders of the clusters too.

The update phase is performed when a new measurement (image) is observed. At the discrete level, particles are resampled based on a measure of the anomaly. This is, for the moment, in contrast to the procedure in Section 3.7.3, where $\lambda(\tilde{S})$ was computed through the described D_B distances. In subsequent chapters, instead, when not explicitly states, the resampling is performed coherently with the process described in Section 3.7.3. At the state level, a modified version of the KF update is performed. This update takes into consideration that μ_t^{IM} and $\sigma_{IM,t}^2$ for image x_t^{IM} can be used as the mapped observation on the state space. Consistently, $\sigma_{IM,t}^2$ can approximate the covariance matrix, such that $\Sigma_t^{IM} \sim I_L \sigma_{IM,t}^2$, representing the uncertainty while encoding images.

Alg. 1 describes the overall AMJPF steps; Algs. 2 and 3 (that are called during the execution of Alg. 1) are related to the Adapted-KF's prediction and update step, respectively. They concern the prediction and update of the sole continuous level and not of the dicrete level. Note that the Adapted-KF is not a true KF, but follows its reasoning.

Coherently with the Bayesian notation already seen in Chapter 2, in the algorithms, a term such as "$\tilde{z}_{t|t-1}$" can be expressed as "the GS estimated at time t, given observations up to time $t-1$". In addition, an apex n is used in some of the variables to identify that they are associated with the *n-th* particle out of N in the PF. In most discussions in this chapter, instead, this apex is omitted for simplicity of notation.

Multi-level anomaly extraction

For conducting anomaly detection, we can exploit the capability of hierarchical Bayesian models, like the MJPF, to perform message passing between the DBN nodes. It should be recalled that three types of messages can be distinguished: *intra-slice*, *top-down* and *bottom-up*. Top-down messages travel from upper nodes to lower ones. Combined with intra-slice messages, they have a predictive capability, as is the case with $\pi(\tilde{S}_t)$ and $\pi(\tilde{z}_t)$ displayed in Fig. 5.3. Bottom-up messages, moving from lower nodes to upper ones, have a diagnostic capability: they are used to adjust the expectations (predictions provided by top-down messages) given a sequence of observations. The combination of predictive and diagnostic messages allows us to perform anomaly detection at different abstraction levels. Three levels of abstraction can be identified: image-level, state-level, and discrete-level. The corresponding anomalies are defined in the paragraphs below.

1) Kullback-Leibler-Divergence Anomaly (KLDA) at the Discrete Level [132]: This anomaly is defined as a distance between messages entering node $\tilde{S}_t$, namely the predictive support $\pi(\tilde{S}_t)$ and the diagnostic support $\lambda(\tilde{S}_{t+1})$. Differences between the probability profiles of the predictive support and the evidence indicate that involved components of the generative model do not fit current observations. In other words, they produce an awareness signal indicating whether and how much the current agent's surroundings behave differently compared to rules learned previously. Since these two messages are discrete probability distributions, the Kullback Leibler-Divergence (KLD) is employed to calculate the difference between them. KLD is an asymmetric divergence. We use its symmetric version, as shown in Eq. 3.25, which we report here for improved readability:

$$KLDA = D_{\mathrm{KL}}(\pi(\tilde{\mathrm{S}}_{\mathrm{t}})||\lambda(\tilde{\mathrm{S}}_{\mathrm{t+1}})) + D_{\mathrm{KL}}(\lambda(\tilde{\mathrm{S}}_{\mathrm{t+1}})||\pi(\tilde{\mathrm{S}}_{\mathrm{t}})) \tag{5.13}$$

To define $\lambda(\tilde{S}_{t+1})$, we use the Bhattacharyya distance between two Gaussian distributions: one coming directly from the GS $\tilde{z}_{t+1}$ and its associated covariance Σ_{t+1} and the other coming from the mean $M^{(\tilde{S})}$ and covariance $Q^{(\tilde{S})}$ values of cluster $\tilde{S}$, for $\tilde{S} = \{1,\ldots,C\}$. The entire process for the calculation of the KLDA is defined in Section 3.7.3.

Algorithm 1 AMJPF.

Input: Π, $M^{(\tilde{S})}$, $Q^{(\tilde{S})}$, $N^{(\tilde{S})} \leftarrow$ Vocabulary, with $\tilde{S} = 1 \dots C$
$x_t^{IM} \leftarrow$ Images $t = \{1, ..., T\}$
N $\leftarrow$ Total number of particles
for *t = 1 to T* $\leftarrow$ *Time evolution* **do**
  Get latent state of the images using VAE's encoders:
  $\mu_t^{IM}, \sigma_{IMt}^2 = q_\theta^{IM}(x_t^{IM}) \leftarrow$ Image latent state
  $\Sigma_t^{IM} \sim I_L \sigma_{IM,t}^2 \leftarrow$ Image latent state covariance
  for *n = 1 to N* $\leftarrow$ *Particles* **do**
    $W_n = \frac{1}{N} \leftarrow$ weight of the particle
    if *t == 1* $\leftarrow$ *Initial iteration* **then**
      Define $\dot{\mu}_t^{IM}$ and $\dot{\sigma}_{IM,t}^2$ through initial prior density $P(\dot{z}_t^{IM})$
      Join (μ_t^{IM}, $\sigma_{IM,t}^2$) and ($\dot{\mu}_t^{IM}$, $\dot{\sigma}_{IM,t}^2$) to form ($\tilde{z}_t$, Σ_t)
      $\tilde{z}_{t|t}^n = \tilde{z}_t \leftarrow$ current state
      $\Sigma_{t|t}^n = \Sigma_t \leftarrow$ current covariance
      Estimate $\tilde{S}_t^n$ from $P(\tilde{z}_t|\tilde{S}_t)$
    end
    else
      ANOMALY DETECTION:
      $\lambda(\tilde{z}_t^n) = P(\tilde{x}_t|\tilde{z}_t)$
      $\lambda(\tilde{S}_t^n) = D_B(\lambda(\tilde{z}_t^n), P(\tilde{z}_t|\tilde{S}_t))$
      Anomaly indicator at the observation level:
      $d\_MSE_t^{IM,n} = MSE(x_t^{IM}, \hat{x}_t^{IM})$
      $MSE_t^{IM,n} = MSE(x_t^{IM}, x_{t|t-1,n}^{IM})$
      Anomaly indicator at the continuous level:
      $Err_t^n = \tilde{z}_{t|t-1,n} - \tilde{z}_t$
      Anomaly indicator at the discrete level:
      $KLDA_t^n$ from $\lambda(\tilde{z}_t^n)$, $\lambda(\tilde{S}_t^n)$ as in Eq. 5.13.
      UPDATE:
      Update Belief in the hidden variables:
      $\tilde{z}_{t|t}^n, \Sigma_{t|t}^n = AKF_{update}(\tilde{z}_{t|t-1}^n, \Sigma_{t|t-1}^n, \mu_t^{IM}, \Sigma_t^{IM})$,
      where AKF_{update} is the update of Alg. 3.
      Update the particles' weight based on the chosen anomaly A_{dist} (e.g., $MSE_t^{IM,n}$) :
      $W_n = \frac{W_n}{A_{dist}}$
    end
    PREDICTION:
    Prediction at the discrete level:
    $\tilde{S}_{t+1}^n \sim \Pi(\tilde{S}_t^n)$
    Prediction at the continuous level:
    $\tilde{z}_{t+1|t}^n, \Sigma_{t+1|t}^n = AKF_{pred}(\tilde{z}_{t|t}^n, \Sigma_{t|t}^n)$
    where AKF_{pred} is the prediction of Alg. 2.
    Prediction at the image level through the VAE's decoder:
    $x_{t+1|t}^{IM,n} = p_\phi(\mu_{t+1|t}^{IM,n})$.
  end
  KLDA calculation.
  SIR resampling
end
Output: MSE_t^{IM}, Err_t, $KLDA_t$

Algorithm 2 Prediction phase of the AKF and OF-AKF (AKF_{pred} and $OF\text{-}AKF_{pred}$).

Input: $\tilde{z}_{t|t}, \Sigma_{t|t}$

Calculation of the sigma points $\tilde{z}^i_{t|t}$ and of their respective weights $\tilde{W}^{i,m}$ and $\tilde{W}^{i,c}$ as described in [221].

The sigma points $\tilde{z}^i_{t|t}$ are propagated through the prediction model f, i.e., the NN of the cluster $N^{(\tilde{S})}$ (see Eq. 5.5 for the AKF, and Eq. 5.20 for the OF-AKF):

$$\tilde{z}^i_{t+1|t} = f(\tilde{z}^i_{t|t}) \tag{5.6}$$

The weighted mean $\tilde{z}_{t+1|t}$ of the propagated sigma points $\tilde{z}^i_{t+1|t}$ is calculated:

$$\tilde{z}_{t+1|t} = \sum_{i=0}^{2L} \tilde{W}^{i,m} \tilde{z}^i_{t+1|t} \tag{5.7}$$

The weighted covariance $\Sigma_{t+1|t}$ of the propagated sigma points is found:

$$\Sigma_{t+1|t} = \sum_{i=0}^{2L} \tilde{W}^{i,c} \{\tilde{z}^i_{t+1|t} - \tilde{z}_{t+1|t}\}\{\tilde{z}^i_{t+1|t} - \tilde{z}_{t+1|t}\}^\intercal \tag{5.8}$$

Output: $\tilde{z}_{t+1|t}, \Sigma_{t+1|t}$

The KLDA measurement allows us to define an anomaly at the highest level of the proposed hierarchy. Consequently, this measurement identifies anomalies in the flow of events, instead of pinpointing pixel-wise abnormalities.

2) Continuous-level anomaly: This anomaly corresponds to the distance between the predictive support $\pi(\tilde{z}_t)$ and the diagnostic support $\lambda(\tilde{z}_{t+1})$. In practice, this can be calculated as the absolute distance between $\tilde{z}_{t+1|t}$ and $\tilde{z}_{t+1}$, or with the $Db1$ and $Db2$ defined in Eqs. 3.30 and 3.29. The anomaly can be either calculated only on the particle with the highest weight in the PF or on all particles. In this second case, it is then averaged over the number of state components ($2L$) and the number of particles (P). In the results section, the first method is chosen.

3) Image-level anomalies: We can distinguish at least two image-level anomalies: the direct reconstruction anomaly and the prediction anomaly at the observation level. The *direct reconstruction anomaly* corresponds to the calculation of the MSE between the reconstructed image ($\hat{x}^{IM}_t$) and the observed image (x^{IM}_t). In Alg. 1 and 4, we differentiate the direct reconstruction anomaly from the prediction anomaly at the observation level by identifying it with a "$d\_$" in front (e.g., $d\_MSE^{IM,n}_t$). On the other hand, the *prediction anomaly at the observation level* (which we also denominate as *image-level prediction anomaly*) is calculated through two phases. The state prediction $\mu^{IM}_{t+1|t}$, related to each particle n, is fed to

Algorithm 3 Update phase of the AKF (AKF_{update}).

Input: $\tilde{z}_{t|t-1}, \Sigma_{t|t-1}, \mu_t^{IM}, \Sigma_t^{IM}$

The subpart of the covariance $\Sigma_{t+1|t}$ related to $\mu_{t|t-1}^{IM}$ is extracted:

$$\Sigma_{t|t-1}^{IM} = \Sigma_{t|t-1}\Big|_{\{row:1...L, col:1...L\}} \tag{5.9}$$

The Kalman Gain (KG) is obtained. The VAE's encoder provides the state-level measurement covariance (Σ_t^{IM}) related to μ_t^{IM}:

$$K_t = [\Sigma_{t|t-1}^{IM}; I_L](\Sigma_{t|t-1}^{IM} + \Sigma_t^{IM})^{-1} \tag{5.10}$$

The estimated mean state $\tilde{z}_{t|t}$ is updated based on the KG K_t. The VAE's encoder directly provides the projection of the new observation at the state level (μ_t^{IM}):

$$\tilde{z}_{t|t} = \tilde{z}_{t|t-1} + K_t(\mu_t^{IM} - \mu_{t|t-1}^{IM}) \tag{5.11}$$

The estimated covariance $\Sigma_{t|t}$ is updated based on K_t:

$$\Sigma_{t|t} = \Sigma_{t|t-1} - K_t(\Sigma_{t|t-1}^{IM} + \Sigma_t)K_t^{\mathsf{T}} \tag{5.12}$$

Output: $\tilde{z}_{t|t}, \Sigma_{t|t}$

the VAE's decoder, and the prediction backpropagated to the observation level is obtained, i.e., $\hat{x}_{t+1|t}^{IM,n}$. Again, this value can be either computed only on the particle with highest weight or it can be averaged over all particles. From a message-passing viewpoint, this anomaly is the distance between the observation at $t+1$ and the predictive message $\pi(\tilde{z}_t)$ forward propagated from the continuous level towards the node $\tilde{x}_{t+1}$.

In Section 3.7.3, we observed how the PF resampling can be performed through the message coming from the observation (see Eq. 3.12). In this chapter, we perform resampling of the particles using the anomalies. We consider both resampling using the image-level anomalies and resampling using the state-level anomalies. The first case corresponds to a message moving directly from node $\tilde{x}_{t+1}$ to node $\tilde{S}_{t+1}$, whereas the second case also considers a message through node $\tilde{z}_{t+1}$. We will show how the results using the state-level anomaly for resampling are comparable to the ones using the image-level anomaly. The former case allows us to reduce the computation time, as there is no need to perform a passage through the VAE's decoder at each time instant t for each particle n.

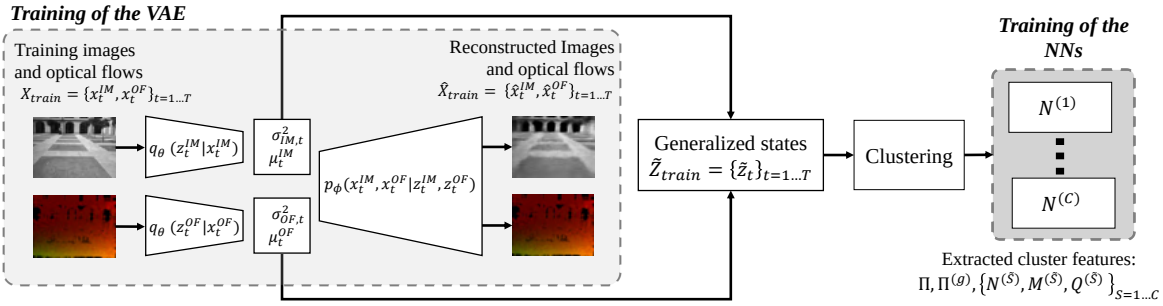


Fig. 5.4 Training phase for the second version of the proposed approach. The VAE is trained to reconstruct a set of training camera images and the OFs between subsequent images. The latent states (i.e., the bottleneck features) are then extracted and employed as GSs. Clustering is performed on the GSs. An NN is trained for each cluster. © 2021 IEEE.

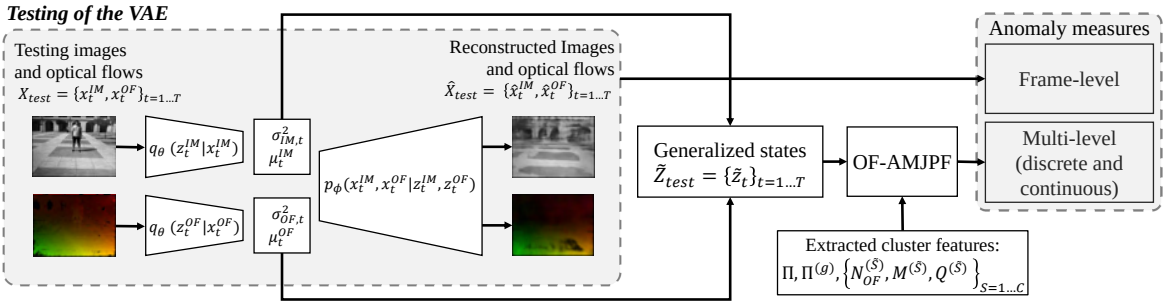


Fig. 5.5 Testing phase for the second version of the proposed approach. The encoders of the VAE are used to extract the GS corresponding to each testing image. The GSs are given as input to the Adapted MJPF, which detects anomalies by using cluster information. © 2021 IEEE.

5.3 Second Version of the Proposed Approach

This section explains the second version of the proposed method. As the first one, it can be divided into a *training* and a *testing* phase. The training phase is examined in Section 5.3.1 and illustrated in Fig. 5.4; the testing phase is discussed in Section 5.3.2 and displayed in Fig. 5.5.

5.3.1 Training phase

Variational Autoencoder

As in the first version, the input to the training phase of the method is the dataset of training images:

$$X_{\text{train}}^{\text{IM}} = \{x_{\text{t}}^{\text{IM}}\}_{\text{t=1...T}}, \tag{5.14}$$

where x_t^{IM} is the camera frame at time instant t, and T is the total number of time instants of the dataset. As a first step of the algorithm, the set of dense OFs corresponding to the set of images is extracted [65]. These are RGB images of the same dimension as the camera frames. We express the set of OFs as:

$$X_{\text{train}}^{\text{OF}} = \{x_{\text{t}}^{\text{OF}}\}_{\text{t=1...T}}, \tag{5.15}$$

where each x_t^{OF} is calculated between the images x_{t-1}^{IM} and x_t^{IM}. The first OF value, x_0^{OF} is set to a null-motion image. We define with $\tilde{X}_{train}$ the combination of these two sets, i.e.,

$$\tilde{X}_{\text{train}} = [X_{\text{train}}^{\text{IM}} X_{\text{train}}^{\text{OF}}]^{\intercal}, \tag{5.16}$$

Therefore, in this case, the observation can be defined as generalized too. This type of observation vector is a Generalized Observation (GO).

Both the images and the OFs are given as input to a VAE through the encoders $q_\theta^{IM}(z_t^{IM}|x_t^{IM})$ and $q_\theta^{OF}(z_t^{OF}|x_t^{OF})$, respectively. The VAE is then trained to reconstruct the images and OFs through the decoder $p_\phi(x_t^{IM}, x_t^{OF}|z_t^{IM}, z_t^{OF})$.

As in the first version of the method, the VAE enables the generation of a latent state having significantly reduced dimension compared to the original image and OF size. We first train the VAE using $\tilde{X}_{train}$. Then, we input again $\tilde{X}_{train}$ to the VAE and extract two sets of latent features, i.e., μ_{train}^{IM}, $\sigma_{IM,train}^2$ for the images, and μ_{train}^{OF}, $\sigma_{OF,train}^2$ for the OFs. The corresponding values at each time instant t can be defined as μ_t^{IM}, $\sigma_{IM,t}^2$, μ_t^{OF}, and $\sigma_{OF,t}^2$.

Generalized states

The proposed latent states extracted from the VAE contain information regarding both the image content and the motion between consecutive frames. Compared to the first version of the approach, a more reliable motion information is captured, as the OF is employed. By combining the two parts of the latent state, μ_t^{IM} and μ_t^{OF}, a GS is therefore obtained. The GS at time t can be written as $\tilde{z}_t = [\mu_t^{IM} \;\; \mu_t^{OF}]^{\intercal}$. The full set of GSs extracted from the training

camera images and OFs can then be formulated as:

$$\tilde{Z}_{train} = [\mu^{IM}_{train} \;\; \mu^{OF}_{train}]^{\intercal} \tag{5.17}$$

Additionally, the variance of each GS, $\tilde{z}_t$, can be defined as $\sigma^2_t = [\sigma^2_{IM,t} \;\; \sigma^2_{OF,t}]^{\intercal}$.

Clustering and neural networks

After obtaining the GSs related to the training video sequences, we use a Growing Neural Gas (GNG) [78] algorithm to cluster GSs into groups that carry similar information. Other clustering methods are potentially applicable, but the GNG is chosen because it does not need the number of clusters to be defined a priori. Conversely, the training of the GNG can be stopped when a specific condition is met. Therefore, a chosen discriminator or a chosen group of discriminators (e.g., the distance from the cluster center, the covariance of clusters) allows one to decide when to stop the algorithm.

Once clustering is performed, the same vocabulary discussed for the first version of the method is extracted. However, the predictive NNs and the overall prediction model are different, as the two subparts of the GS can no longer be directly derived one from another. Indeed, μ^{OF}_t does not correspond to $\mu^{IM}_t - \mu^{IM}_{t-1}$ as was the case with $\dot{\mu}^{IM}_t$ in the first version of the approach. Conversely, μ^{IM}_t and μ^{OF}_t now lie on two separate VAE latent spaces that are not directly relatable.

To perform the training of each $N^{(\tilde{S})}_{OF}$, we give as input to the NN the value of every GS $\tilde{z}_t$ assigned to cluster $\tilde{S}$. The obtained GS-level prediction $\tilde{z}_{t+1|t}$ is then passed to the decoder $p_\phi(x^{IM}_t, x^{OF}_t | z^{IM}_t, z^{OF}_t)$. The outputs of the decoder are the corresponding observation-level predictions, i.e., $\hat{x}^{IM}_{t+1|t}$ (image prediction) and $\hat{x}^{OF}_{t+1|t}$ (OF prediction). These predictions are compared with the actual image and OF values, i.e., x^{IM}_{t+1} and x^{OF}_{t+1}. The related MSE loss is used to optimize the parameters of $N^{(\tilde{S})}_{OF}$.

Summarizing, each $N^{(\tilde{S})}_{OF}$ performs the following approximation:

$$\tilde{z}_{t+1} \sim N^{(\tilde{S})}_{OF}(\tilde{z}_t) + w_t, \tag{5.18}$$

where w_t is the network's residual error after convergence.

Learned Dynamic Bayesian Network

Fig. 5.3b shows the DBN learned in this version of the approach. A three-level DBN is adopted, as in the first version. Most of the elements of the two DBNs are the same. However, the observation model is different: instead of only observing the images, GOs are provided

as input. The observation model is composed of a couple of encoders $q_\theta(z_t^{IM}|x_t^{IM})$ and $q_\theta(z_t^{OF}|x_t^{OF})$ and of a decoder $p_\phi(x_t^{IM},x_t^{OF}|z_t^{IM},z_t^{OF})$.

Moreover, the two NN models are expressed differently in the two cases (see Eqs. 5.4 and 5.18).

5.3.2 Testing Phase

During the testing phase, each testing image and OF is processed by the VAE, and GSs are calculated. Then, a filter that we denote as OF-AMJPF is used to perform predictions and to detect anomalies.

Adapted Markov Jump Particle Filter

Alg. 4 describes the overall OF-AMJPF steps.

The OF-AMJPF follows an algorithm similar to the one of the AMJPF. However, it incorporates specific adaptations tailored to the distinct origins and attributes of the GS, and to the distinct approach for training the NNs.

Firstly, since the NNs are trained to directly predict the full GS at the next time instant, instead of only half of it, the UKF prediction Eq. 5.3 is changed as follows:

$$\begin{aligned} \tilde{z}_{\mathrm{t}}{}^{i} &= \tilde{z}_{\mathrm{t}} + (\sqrt{(L+\lambda)\Sigma_{\mathrm{t}}})_{\mathrm{i}} \qquad \textbf{if} \;\; i = 1...2L \\ \tilde{z}_{\mathrm{t}}{}^{i} &= \tilde{z}_{\mathrm{t}} - (\sqrt{(L+\lambda)\Sigma_{\mathrm{t}}})_{\mathrm{i-2L}} \quad \textbf{if} \;\; i = 2L+1...4L, \end{aligned} \tag{5.19}$$

where λ is a scaling parameter and $\Sigma_t \sim I_{2L}\sigma_t^2$, being I_{2L} the identity matrix of dimension $2L$. With L, also in this case, we identify the dimension of the latent state part related to the camera frames only.

Secondly, the prediction of a sigma point i follows the equation below:

$$\tilde{z}^{i}{}_{\mathrm{t+1}} = f(\tilde{z}^{i}{}_{\mathrm{t}}) = N^{(\tilde{\mathrm{S}})}(\tilde{z}^{i}{}_{\mathrm{t}}) + w^{i}{}_{\mathrm{t}}, \tag{5.20}$$

Alg. 4 describes the overall OF-AMJPF steps. The same continuous-level prediction algorithm (Alg. 2) used in the first version of the approach is adopted. However, notice that Eq. 5.6 is different in the two cases. On the other hand, the continuous-level update algorithm adopted by the OF-AKF is different from the one of the AKF and is Alg. 5. These two continuous-level OF-AKF algorithms are employed in Alg. 4.

Algorithm 4 OF-AMJPF.

Input: Π, $M^{(\tilde{S})}$, $Q^{(\tilde{S})}$, $N_{OF}^{(\tilde{S})} \leftarrow$ Vocabulary, with $\tilde{S} = 1 \dots C$
x_t^{IM}, $x_t^{OF} \leftarrow$ Images and OFs $t = \{1, ..., T\}$
N $\leftarrow$ Total number of particles
for *t = 1 to T* $\leftarrow$ *Time evolution* **do**
  Get latent state of images and OFs using VAE's encoders:
  $\mu_t^{IM}, \sigma_{IM,t}^2 = q_\theta^{IM}(x_t^{IM}) \leftarrow$ Image latent state
  $\mu_t^{OF}, \sigma_{OF,t}^2 = q_\theta^{OF}(x_t^{OF}) \leftarrow$ OF latent state
  $\tilde{z}_t = [\mu_t^{IM} \;\; \mu_t^{OF}]^\intercal$, $\sigma_t^2 = [\sigma_{IM,t}^2 \;\; \sigma_{OF,t}^2]^\intercal \leftarrow$ GS
  $\Sigma_t \sim I_L \sigma_{IM,t}^2 \leftarrow$ Latent state covariance
  for *n = 1 to N* $\leftarrow$ *Particles* **do**
    $W_n = \frac{1}{N} \leftarrow$ weight of the particle
    if *t == 1* $\leftarrow$ *Initial iteration* **then**
      $\tilde{z}_{t|t}^n = \tilde{z}_t \leftarrow$ current state
      $\Sigma_{t|t}^n = \Sigma_t \leftarrow$ current covariance
      Estimate $\tilde{S}_t^n$ from $P(\tilde{z}_t|\tilde{S}_t)$
    end
    else
      ANOMALY DETECTION:
      $\lambda(\tilde{z}_t^n) = P(\tilde{x}_t|\tilde{z}_t)$
      $\lambda(\tilde{S}_t^n) = D_B(\lambda(\tilde{z}_t^n), P(\tilde{z}_t|\tilde{S}_t))$
      Anomaly indicator at the observation level:
      $MSE_t^{IM,n} = MSE(x_t^{IM}, x_{t|t-1}^{IM,n})$
      $d\_MSE_t^{IM,n} = MSE(x_t^{IM}, \hat{x}_t^{IM})$
      $MSE_t^{OF,n} = MSE(x_t^{OF}, x_{t|t-1,n}^{OF,n})$
      $d\_MSE_t^{OF,n} = MSE(x_t^{OF}, \hat{x}_t^{OF})$
      Anomaly indicator at the continuous level:
      $Err_t^n = \tilde{z}_{t|t-1}^n - \tilde{z}_t$
      Anomaly indicator at the discrete level:
      $KLDA_t^n$ from $\lambda(\tilde{z}_t^n)$, $\lambda(\tilde{S}_t^n)$ as in Eq. 5.13.
      UPDATE:
      Update Belief in hidden variables:
      $\tilde{z}_{t|t}^n, \Sigma_{t|t}^n = OF\text{-}AKF_{update}(\tilde{z}_{t|t-1}^n, \Sigma_{t|t-1}^n, \tilde{z}_t, \Sigma_t)$,
      where $OF\text{-}AKF_{update}$ is the update of Alg. 5.
      Update particles' weight based on chosen anomaly A_{dist} (e.g., $MSE_t^{IM,n}$) :
      $W_n = \frac{W_n}{A_{dist}}$
    end
    PREDICTION:
    Prediction at the discrete level:
    $\tilde{S}_{t+1}^n \sim \Pi(\tilde{S}_t^n)$
    Prediction at the continuous level:
    $\tilde{z}_{t+1|t}^n, \Sigma_{t+1|t}^n = OF\text{-}AKF_{pred}(\tilde{z}_{t|t}^n, \Sigma_{t|t}^n)$
    where $OF\text{-}AKF_{pred}$ is the prediction of Alg. 2.
    Prediction at the image level through the VAE's decoder:
    $x_{t+1|t}^{IM,n}, x_{t+1|t}^{OF,n} = p_\phi(\tilde{z}_{t+1|t}^n)$.
  end
  KLDA calculation.
  SIR resampling
end
Output: MSE_t^{IM}, MSE_t^{OF}, Err_t, $KLDA_t$

Algorithm 5 Update phase of the OF-AKF ($OF\text{-}AKF_{update}$).

Input: $\tilde{z}_{t|t-1}, \Sigma_{t|t-1}, \tilde{z}_t, \Sigma_t$

The Kalman Gain (KG) is obtained. The VAE's encoders provide the state-level measurement covariance (Σ_t):

$$K_t = \Sigma_{t|t-1}(\Sigma_{t|t-1} + \Sigma_t)^{-1} \tag{5.21}$$

The estimated mean state $\tilde{z}_{t|t}$ is updated based on the KG K_t. The VAE's encoder directly provides the projection of the new observation at the state level ($\tilde{z}_t$):

$$\tilde{z}_{t|t} = \tilde{z}_{t|t-1} + K_t(\tilde{z}_t - \tilde{z}_{t|t-1}) \tag{5.22}$$

The estimated covariance $\Sigma_{t|t}$ is updated based on K_t:

$$\Sigma_{t|t} = \Sigma_{t|t-1} - K_t(\Sigma_{t|t-1} + \Sigma_t)K_t^{\mathsf{T}} \tag{5.23}$$

Output: $\tilde{z}_{t|t}, \Sigma_{t|t}$

Multi-level anomaly extraction

All anomalies described in Section 5.2.2 for the AMJPF can be adopted also in the OF-AMJPF.

Moreover, we have to consider that, in the OF-AMJPF, a GO is present. Consequently, the two discussed image-level anomalies can be defined for the OF too and not only for the camera frames. Through the VAE's decoder, we can take the observed OF (x_t^{OF}), reconstruct it ($\hat{x}_t^{OF}$), and calculate an OF direct reconstruction anomaly. Similarly, from the state level prediction related to each particle ($\tilde{z}_{t+1|t}^n$), we can calculate a GO reconstruction ($\hat{x}_{t+1|t}^n$). Each GO reconstruction is composed of the camera frame reconstruction and of the OF reconstruction. In this way, the OF prediction anomaly at the observation level can be obtained too.

5.4 Advantages of using the proposed approach

The use of the AMJPF or of the OF-AMJPF provides the following advantages inside the multi-modal self-aware framework described in Chapter 3:

- it allows us to parameterize the different variables of the problem and their relationships through the use of a DBN.
- it provides homogeneity with low-dimensional sensory modalities, which were considered in previous work [17, 115]. In both of these works and in this chapter, a form of MJPF is used, allowing message-passing and coherent anomaly detection at multiple

levels. The high levels of the DBN hierarchy are expressed in the same way across modalities. This means that clear and reusable semantics are adopted. Therefore, even if multi-modality is not used in this chapter, it could be easily reached. However, also note that one significant different is still present between the proposed approach and the method in Section 3.7 for low-dimensional data: the use of neural networks instead of piece-wise linear prediction models for state prediction.

- it permits the adoption of probabilistic distances, leading to a coherent definition of probabilistic anomalies at the different levels of the DBN.
- the AMJPF and OF-AMJPF are coherent with the analyzed self-awareness theories and framework: both are GS Bayesian filters (the family of filters proposed by Friston) allowing us to model uncertainty, to build a hierarchical representation of information and to extract multi-level anomalies, which could be consequently used to trigger a new training phase.

5.5 Comparison between the two versions of the approach

From here on, we will refer to the first version of the approach as FV (*First Version*) and to the second one as SV (*Second Version*). The main change inserted in the SV compared to the FV is the use of the OF to represent the GS' time-derivative information instead of the difference between consequent image encodings. This leads to a more direct and effective clustering of the vehicle motion and to more accurate discrete-level predictions. We will observe in the results section that better anomaly detection performance is obtained with the SV.

On the other hand, however, the SV introduces an evident model-driven element to the approach. The dense OF used in this chapter is calculated through the Farneback method [65]. Therefore, a model of motion estimation is adopted instead of having the networks directly learn the connection between the motion and appearance state parts. Furthermore, the calculation of the OF increases the computational complexity of the method.

Both versions comply with three of the remaining self-awareness characteristics introduced in Chapter 3. They use Bayesian inference and are hierarchical. They employ DBNs, which are interpretable models, and they further enhance this capability through hierarchical anomalies that identify to which level of the DBN the abnormal event is related. Additionally, in both versions of the approach, the dynamical rules emerge from the data. They are initially learned from the normal model. The anomalies signal that new rules are emerging. Through

an incremental learning step (not covered in this thesis), these emergent rules could be learned too.

It must, however, be observed that the model is still not multi-sensorial. This capability will be covered in Chapters 7 and 8.

5.6 Incremental Learning

The problem of incremental learning is not tackled in this thesis, but in this section, we provide some insights on future work in this direction.

Our architecture can be extended for learning new scenarios: when abnormal situations are detected, the input data related to them can be used for building a new model. In such cases, the initial VAE (that we define here as VAE_0) is copied and its weights are fine-tuned using the set of abnormal images (and OFs), generating thus a VAE_1. The rest of the training process is repeated, building the GS through the bottleneck features related to the abnormal images (and OFs).

During the new testing phase, VAE_0 and VAE_1 are used simultaneously. The minimum anomaly signal between the two available VAEs is considered as the system's anomaly measurement. In this way, a hierarchy of models $\{\mathscr{M}_i\}$ can be built incrementally as anomalies are detected, where $i = 0...V$, being V the number of trained VAEs. This would allow the system to adapt itself to unseen situations as they are experienced by fine-tuning previously learned VAEs and their associated NNs. Emergent rules can be, therefore, incrementally learned.

In situations in which the VAE is already able to recognize and reconstruct all possible image scenarios based on the appearance, but video dynamics/motions are new, the creation of a new VAE is not necessary. It is sufficient to build new clusters and to train new NNs to recognize abnormal dynamics/motions. Hence, the method's modularity and its capability to interpret which model parts are inadequate emerge as distinct advantages.

It can be observed that, in this way, future predictions are improved, leading to the minimization of anomalies and GEs (and, therefore, of free energy).

5.7 Experimental Results

5.7.1 Employed Evaluation Datasets

In this section, we adopt three evaluation datasets: the Icab dataset [157] (see Section 4.2.1), the Subway dataset [1] (see Section 4.2.4), and the Avenue dataset [151] (see Section 4.2.4).

First, we analyze and compare in detail the two versions of the approach on the Icab dataset. We observe that the SV displays higher performance. The PM scenario (i.e., PM_1 and PM_2) of the Icab dataset is used for training; the other scenarios (i.e., PA, ES, and U-turn) are adopted for testing. We aim to detect, in the testing scenarios, the abnormal maneuvers caused by the presence of pedestrians. Furthermore, as observed in Section 4.2.1, training and testing data were captured at different hours of the day and camera orientations, resulting in differences in lighting conditions and objects positions relative to the camera. Strong lighting differences account for additional anomalies.

Then, we test the SV on the Subway and Avenue anomaly detection datasets. These are static-camera datasets. The results show that, while the approach was developed for vehicular datasets, it is also applicable to static-camera surveillance datasets.

5.7.2 Performance evaluation metrics

We calculate the anomaly indicators described in Sections 5.2.2 and 5.3.2. To evaluate the goodness of the results we build the Receiver Operating Characteristic (ROC) curve for each anomaly signal and we compute evaluation metrics such as the Area Under the Curve (AUC) and the maximum ACCuracy (ACC).

A ROC curve plots the values of True Positive Rate (TPR) and False Positive Rate (FPR) at different thresholds, where [91]:

$$TPR = \frac{TP}{TP+FN}, \tag{5.24}$$

$$FPR = \frac{FP}{FP+TN}, \tag{5.25}$$

where TP stands for "True Positive", FN for "False Negative", FP for "False Positive", and TN for "True Negative". In anomaly detection, TPR and FPR consist in the percentage of times in which abnormalities are correctly identified and in which they are incorrectly assigned, respectively. A threshold that allows perfect discrimination would pass from the left upper corner of the ROC curve, i.e. $FPR = 0$ and $TPR = 1$. To make an example of the meaning of this type of diagram, let us consider the ROC curves in Figs. 5.7b and 5.7a, ignoring for the moment the meaning of the illustrated anomalies. Note how the curves always start in $(0,0)$. This point corresponds to a threshold above the maximum abnormality value: all values are assigned to the normal class, resulting in no false positives, but also in no abnormality being detected (true positives). The curves always end in $(1,1)$, the point that corresponds to a zero threshold: all data are assigned to the abnormal class. The other points of the curve correspond to thresholds in the middle.

The AUC of the ROC is widely employed to measure the performance of classification methods [165]. Another metric adopted here is the maximum ACC: we calculate the ACC at the different thresholds and select the threshold with the highest accuracy. The ACC corresponds to:

$$ACC = \frac{TP+TN}{TP+TN+FP+FN}, \tag{5.26}$$

There are different ways to find an optimal threshold, such as finding the threshold that maximizes the accuracy, or the point on the ROC curve with equal value of sensitivity and specificity, or the point with the minimum distance from the left-upper corner of the unit square, or the point where the Youden's index has maximum value [90].

5.7.3 Implementation details

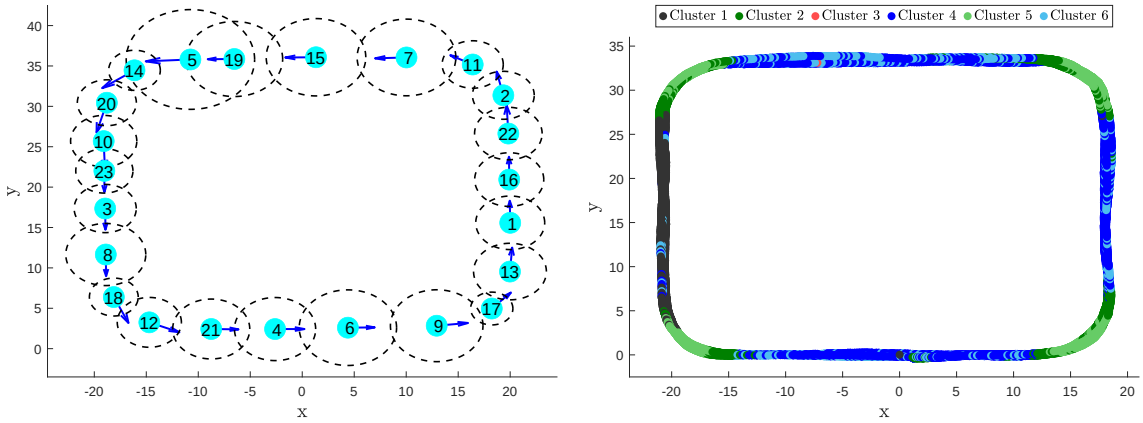
Table I Layers of the VAE's encoder. *NI* stands for the number of input channels of the image. © 2021 IEEE.

Layer	**Output**	**Activation**	**Kernel**	**Stride**
1.*Input*	120*x*240*xNI*	*Identity*	–	–
2.*Conv*2*D*	87*x*117*x*16	*ReLU*	7,7	2,2
3.*Conv*2*D*	41*x*56*x*32	*ReLU*	7,7	2,2
4.*Conv*2*D*	18*x*25*x*64	*ReLU*	5,5	2,2
5.*Conv*2*D*	7*x*11*x*128	*ReLU*	5,5	2,2
6.*Conv*2*D*	2*x*4*x*128	$ReLU + Flatten$	5,5	2,2
7.FC_μ, FC_σ	32	*Identity*	–	–

To adjust to the possible camera orientations, we perform data augmentation using 9 different crops of the images inside a central window. This is performed for the iCab case only, as in the Subway and Avenue datasets the camera is static and the orientation does not change.

As we desire to perform a fair comparison between the two cases, the implementation details of the FV and of the SV are as similar as possible, such that both cases: *i)* use the same layers in the two VAEs and in the prediction NNs; *ii)* employ the same image-level loss during training; *iii)* use GNG as a clustering method; *iv)* have the same dimension of GSs (64) and the same number of clusters (6).

The encoding layers of the VAEs are summarized in Table I; the decoder is symmetrical. The learning rate is set to $5e^{-5}$ and weight decay to 0.01. The Adam optimizer was used. To



(a) Odometry clustering.

(b) Video Clustering.

Fig. 5.6 Odometry and video Clustering. © 2021 Elsevier, http://www.elsevier.com.

perform prediction, we used fully connected NNs composed of 5 layers with ReLu activation functions. The network was implemented and trained using the PyTorch library.

5.7.4 Clustering visualization

As a first step of the presented framework, a normality model must be learned. In both the FV and in the SV, the overall model includes the VAE encoder and decoder, the discrete superstates with their statistical properties (i.e., mean value and covariance matrix), the transition matrices describing the links between superstates, and the NNs capturing the prediction model inside each cluster.

Both in the camera approach proposed in this chapter and in the odometry approach presented in [17], clustering must be performed. We compare here the clustering that can be obtained through the use of odometry data and of video data (taking the SV as an example). In this section, we only refer to the iCab dataset, as the Subway and Avenue datasets don't have any significant odometry information.

Fig. 5.6 displays the performed clustering on the odometry and video data. Observing the odometry case (5.6a), the mean values $M^{(\tilde{S})}$ for each cluster can be visually recognized as the mean value of the position data of the clusters (shown through the cyan dots) and the mean values of the speed data of the clusters (displayed through the blue arrows). Additionally, the dashed circles are proportional to the standard deviation of the clusters' positions, which can be extracted from $Q^{(\tilde{S})}$. For better visualization purposes, instead of plotting the cluster extension with the accurate ellipse shape, we use a circumference of radius equal to the maximum standard deviation along the two directions.

Instead, Fig. 5.6a represents clustering in the video case. As opposite sides of the courtyard display similar visual appearances, while the same motions are performed, symmetry is present. A shadow area is assigned a unique separate cluster (*cluster* 1).

From the figure, it can be observed how, in contrast to odometry clustering, video clustering can join areas that are very far away in space. By clustering odometry and video separately, two very different clustering results can be obtained. In Chapters 7 and 8 we will explore two other ways of handling the clustering of data. In those two chapters, odometry and video data are elaborated together.

5.7.5 Comparison between M1 and M2

Definition of normality

The PM frames (and their related OFs) are used as x_{train}^{IM} (and x_{train}^{OF}) to perform the training phase described in Sections 5.2.1 and 5.3.1. Clustering experiments were performed by varying the total number of clusters C; and the case $C = 6$ was chosen due to its accuracy at representing scenes while employing a low number of clusters.

Since NNs are used to make predictions and each NN is associated with a cluster, the number of training samples per cluster is a relevant parameter to consider, for avoiding underfitting issues. Accordingly, each cluster has an average number of $9,022$ samples, with the smallest one containing $2,924$ samples in the FV and 412 samples in the SV.

Pedestrian Avoidance

In this task, the vehicle avoids a static pedestrian (see Fig. 4.16b). Four shady areas are also present, which are abnormal areas too.

Fig. 5.7 displays anomaly signals at the different levels of the DBN and the corresponding ROC curves. These abnormalities are the ones described in Section 5.2.2. The arrangement of the subfigures follows a hierarchical structure that partly mirrors the DBN hierarchy. At the bottom, in Fig. 5.7c, are direct reconstruction anomalies. In the middle, in Fig. 5.7b, are anomalies pertaining to GS predictions backpropagated at the image level, i.e., the prediction anomalies at the observation level discussed in Section 5.2.2. Finally, at the top, in Fig. 5.7a, are the discrete-level KLDAs. In all three cases, the FV results are in the plot above, and the SV results are in the plot below. ROC curves associated with the first two cases are showcased together in Fig. 5.7e (as they all are observation-level anomalies); ROC curves related to discrete-level anomalies are displayed in Fig. 5.7d. Below the ROC curves, the AUC and ACC values for each anomaly are displayed.

Let us start from the top a more detailed description of the plots.

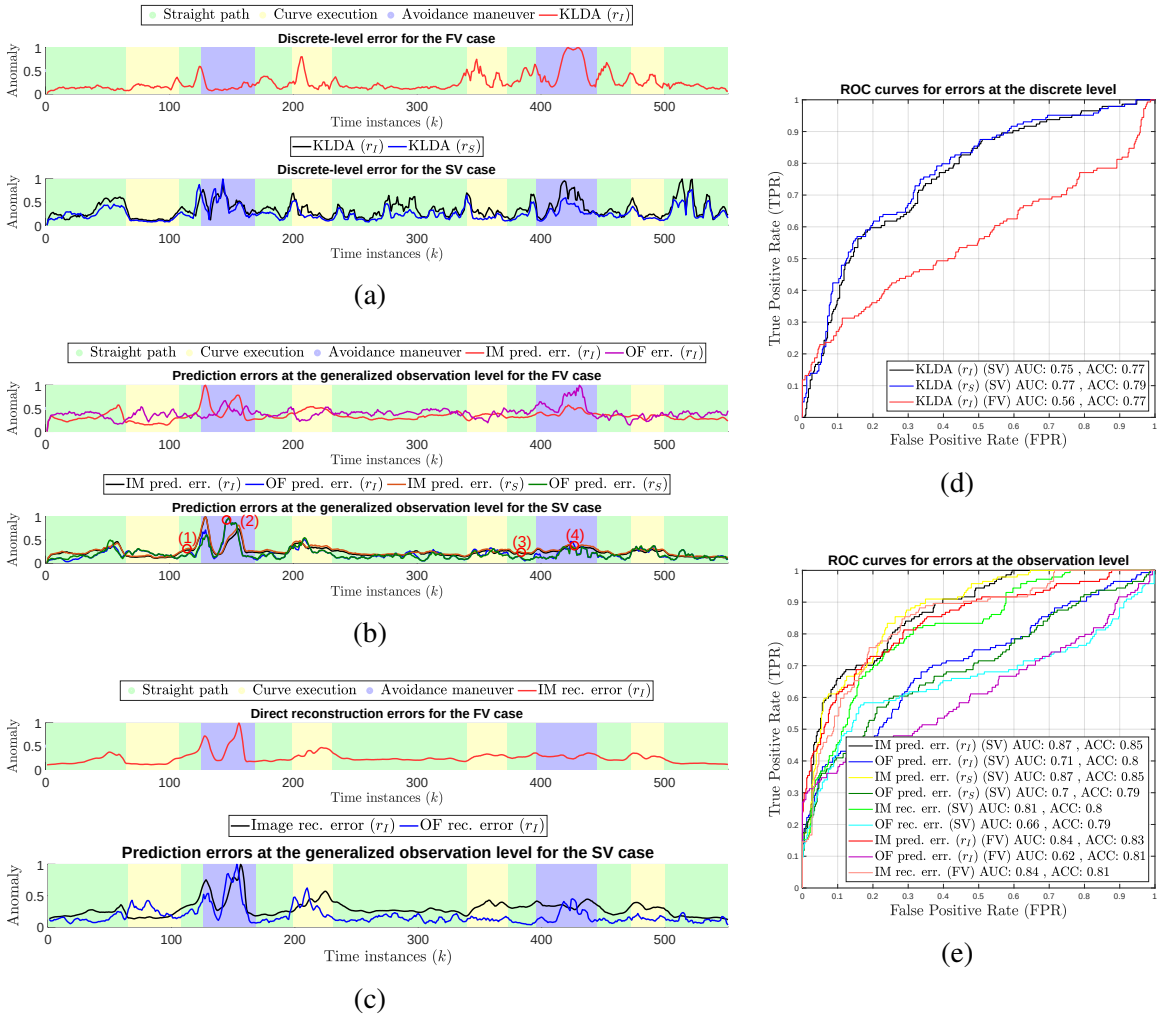
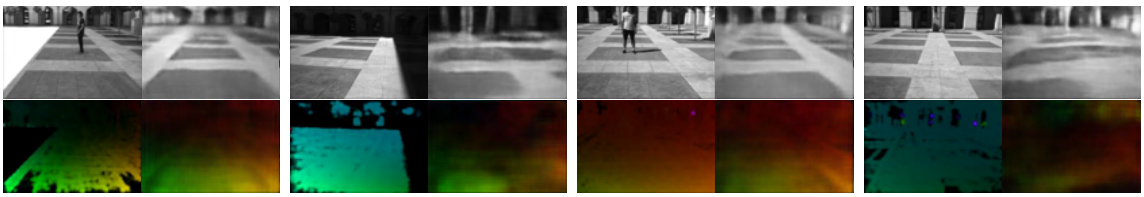


Fig. 5.7 Multi-level anomalies (a,b,c) and corresponding ROC curves (d,e) in the Pedestrian Avoidance task. © 2021 IEEE.



(a) Point (1). (b) Point (2). (c) Point (3). (d) Point (4).

Fig. 5.8 Time instants in PA numerically defined in Fig. 5.7b: for each one, the real value of the frame or OF is on the left and the corresponding reconstruction of the prediction is on the right. (a)-(c): static pedestrian; (b)-(d): avoidance maneuver. © 2021 IEEE.

Fig. 5.7a shows the anomaly at the discrete level. Note that SV discriminates both anomaly zones, whereas FV detects only the first one. In the SV, a resampling of image-level anomalies (r_I) and state-level anomalies (r_S) is considered; in the FV, only r_I is shown. r_I and r_s produce similar results. The anomaly at the discrete level signals that an abnormal passage between clusters is performed. Indeed, when avoiding the pedestrian, the vehicle executes an abnormal sequence of actions and a passage in an unexplored area.

From the image-level prediction anomalies in Fig. 5.7b, it can be observed that avoidance maneuvers always present a high anomaly in both FV and SV cases with either r_I or r_s. However, the pedestrian presence does not generate significant anomaly values. This is due to the mixing effect of the pedestrian with the background, especially in the second pedestrian case, where he wears a white shirt on a light-colored background (see Fig. 5.8c). Moreover, Fig. 5.8 shows the image-level comparisons between the real camera frames and OFs and the corresponding predictions for some time instants numbered in Fig. 5.7b. Similar ROC curves are obtained from the two cases, with SV performing slightly better.

Note that, for the sake of a full comparison, we report an OF prediction error result also in the FV case. This is obtained by calculating the OF with the Farneback method between the predicted image and the previous one, and by then computing the MSE between this OF and the real one. Therefore, the calculation of this anomaly is different and more time-consuming than the one performed with the SV.

Finally, Fig. 5.7c displays the anomalies coming from a direct reconstruction method, i.e., inputting images (and OFs, in the SV) to the VAE and calculating the MSE between them and the outputted reconstructions. Note that since the VAE architectures of the FV and of the SV are similar (allowing a fair comparison), both VAEs generate similar anomaly values in the image reconstruction. Comparing the AUC and ACC obtained with the direct reconstruction anomalies and the prediction anomalies, it can be observed that the anomaly detection is slightly improved in the latter case. Also, note that a direct reconstruction can miss anomalies due to the generalization abilities of Convolutional Autoencoders (the VAE in our case) [149].

Observation-level anomalies signal us that abnormal observations are present in the scene and that the car is moving abnormally. For example, in Figs. 5.8a and 5.8c, note that the pedestrian does not get reconstructed and, therefore, a high image-level abnormality corresponds to it. In Fig. 5.8b and 5.8d, the vehicle turns to the right, which was never performed during training, resulting in an unseen OF that cannot either be reconstructed or predicted accurately. On the other hand, motions seen during training, but not expected when traveling in a certain area, would generate a high prediction anomaly at the observation level, but not a high direct reconstruction anomaly, as we will see in an example in the ES scenario.

Pedestrian avoidance through U-turn

In this task, the vehicle avoids a static pedestrian with a U-turn maneuver and then moves away from the pedestrian (see Fig. 4.16c). This maneuver introduces new situations later on, e.g., curving in the opposite direction, and already known behaviors, e.g., moving linearly in regions with similar symmetries to those in the training set. The second and third executed curves are abnormal, as they are performed in the opposite direction compared to the training set. As in the previous case, some shadow areas are present in this scenario too.

The image-level prediction anomalies are shown in Fig. 5.9b. The performance of FV and SV are comparable at this level, but SV offers slightly better results. Regions corresponding to the U-turn maneuver, curves in the opposite direction, and shady areas tend to display high anomalies. The pedestrian goes mostly undetected due to his "camouflage" with the background (see Fig. 5.10a).

Fig. 5.9c displays the direct reconstruction anomalies. For this task, despite the discrete-level anomalies being more peaked, direct reconstruction is sufficient to detect all abnormal areas. Indeed, most of the abnormal areas display observations and motions that are new to various degrees, instead of including known but unexpected observations or motions. For example, in Fig. 5.10c, the vehicle turns right while moving toward the center of the courtyard. During training, the vehicle never turned right nor was oriented toward the center of the courtyard.

Fig. 5.9a shows the discrete-level anomalies. As before, this anomaly detects abnormal sequences of coupled appearances and motion, that, in this scenario, correspond to the U-turn maneuver and to the execution of curving actions in the opposite direction compared to training. The reader should, however, also take into consideration that, if the VAE is fed data significantly different from training, the latent state will be less meaningful and this will impact the higher-level anomalies too, potentially causing their values to increase.

Emergency Stop maneuver

In this task, the vehicle performs an emergency stop maneuver and allows a pedestrian to cross in front of it. Fig. 4.16d shows the anomaly zone (crossing pedestrian) in blue. Additionally, after avoiding the first pedestrian, the driver performs an abnormal stabilization maneuver, moving first right and then left to center the car between the white lines of the courtyard (see Fig. 5.12b).

Fig. 5.11a displays the discrete-level anomalies. The FV detects the pedestrian anomalies but not the abnormal sequence of maneuvers. The SV case can detect the stabilization maneuver too.

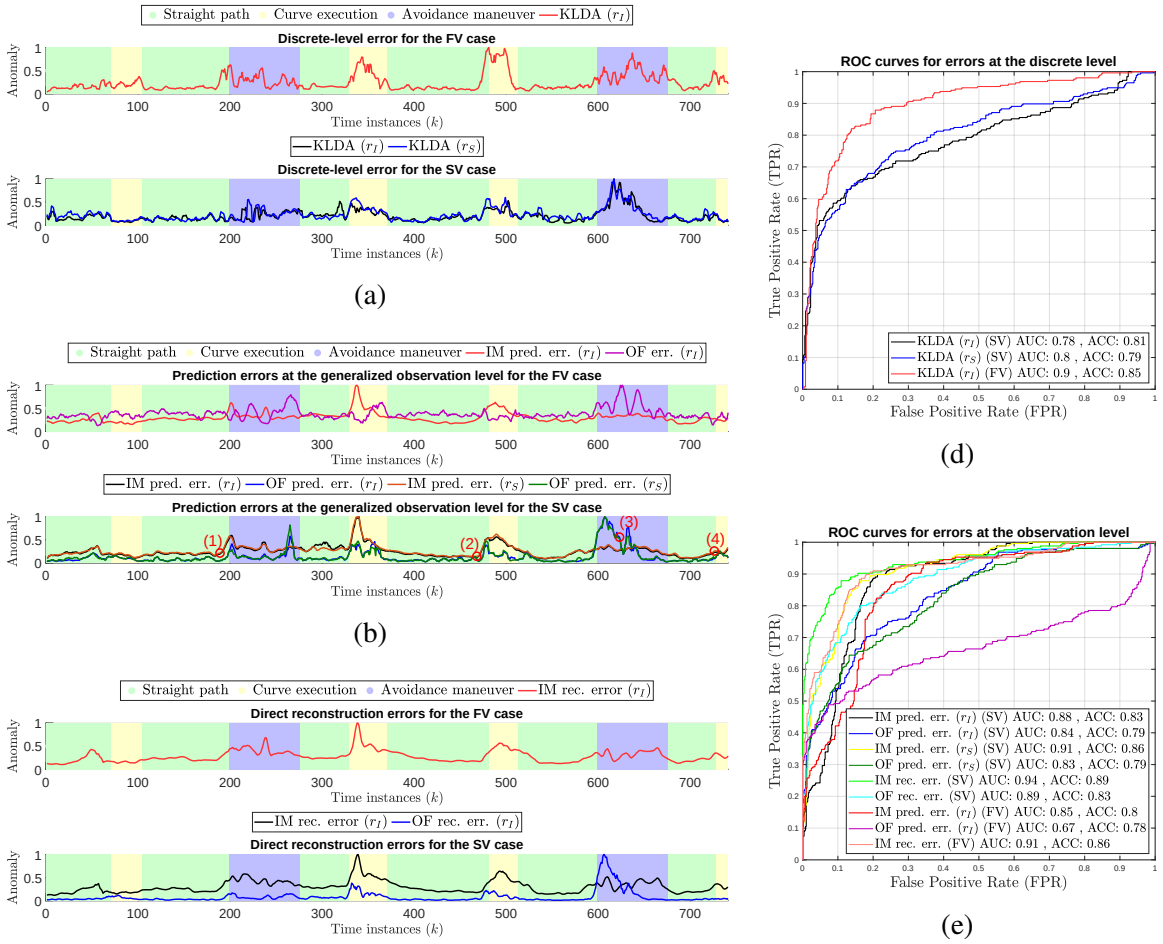
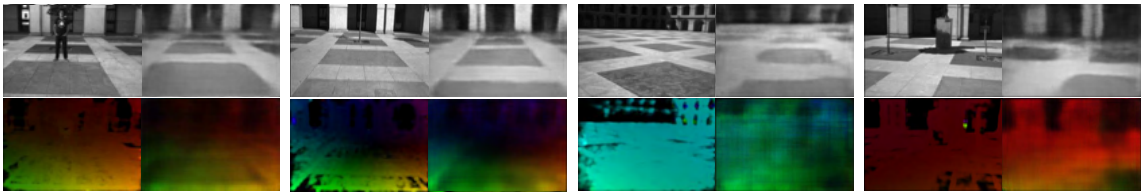


Fig. 5.9 Multi-level anomalies (a,b,c) and corresponding ROC curves (d,e) in the U-turn task. © 2021 IEEE.



(a) Point (1). (b) Point (2). (c) Point (3). (d) Point (4).

Fig. 5.10 Time instants in U-turn numerically defined in Fig. 5.9b. (a): static pedestrian; (b): normal straight motion on the opposite side of the courtyard; (c): U-turn; (d): normal curve. © 2021 IEEE.

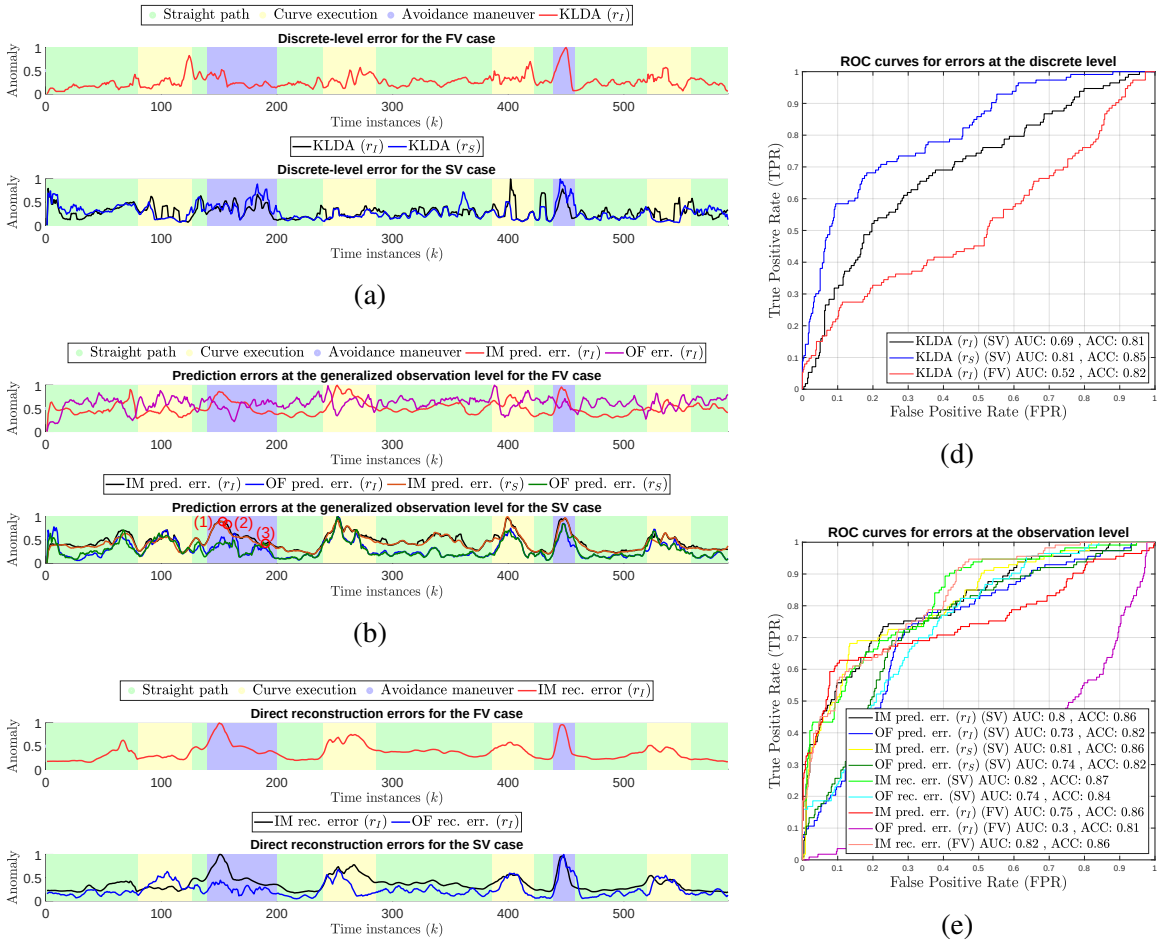
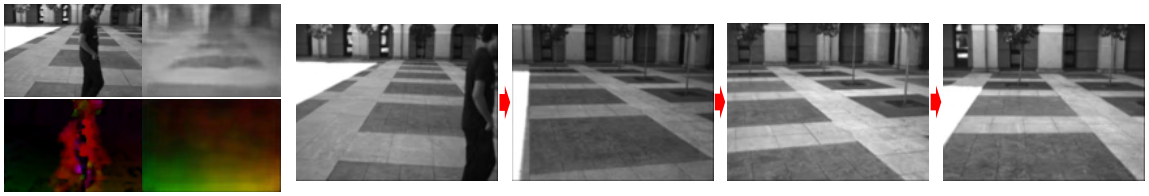


Fig. 5.11 Multi-level anomalies (a,b,c) and corresponding ROC curves (d,e) in the Emergency Stop task. © 2021 IEEE.



(a) Point (1). (b) Some images from point (2) to (3).

Fig. 5.12 Time instants in ES numerically defined in Fig. 5.11b. (a): crossing pedestrian (b): Some images in the sequence going from (2) to (3) in Fig. 5.11b. © 2021 IEEE.

Table II Execution time with 10 and 50 particles for the SV and FV using resampling through image-level (r_I) and state-level (r_S) anomalies. Additionally, the AUC values of the different anomalies for PA, U-turn, and ES are shown (with r_S). The AUC values denoted for shortness as "Image" and "OF" refer to the observation-level prediction anomalies. © 2021 IEEE.

	N. part.	time SV r_I (ms)	time SV r_S (ms)	time FV r_I (ms)	time FV r_S (ms)	AUC_{PA}	AUC_U	AUC_{ES}
Image	10	$57+OF$	$38+OF$	$48(+2OF)$	$37(+2OF)$	0.86	0.88	0.8
OF						0.72	0.84	0.73
KLDA						0.77	0.74	0.69
Image	50	$211+OF$	$123+OF$	$170(+2OF)$	$123(+2OF)$	0.87	0.91	0.81
OF						0.70	0.83	0.74
KLDA						0.77	0.80	0.81

Fig. 5.11b shows the image-level prediction anomalies due to the pedestrians for both the FV and SV cases, with the SV having improved performance.

Finally, Fig. 5.11c exhibits the direct reconstruction anomalies. This sequence, compared to the PA and U-turn ones, displays more anomaly cases related to unexpected but known motions or sequences of motions in known environments. Examples are the deceleration and acceleration before and after seeing the pedestrian, and the stabilization motion. In Fig. 5.12a, we observe the vehicle decelerating in front of the pedestrian. The deceleration motion is known, and therefore, we would expect that the predicted OF could be more or less reconstructed. Instead, as we are in an area where we expect to move forward, the networks predict an OF encoding a forward motion. The anomaly is thus detected.

In this task, the ROC curves of some of the signals display very low performance; this is due to the inability of most of the signals to capture sequential anomalies, which, instead, is the main objective of the KLDA anomaly. For example, direct reconstruction captures the pedestrian anomaly accurately, but not the stabilization anomaly. By combining the different anomaly signals, the best performance could be obtained. This anomaly fusion is, however, not tackled in this thesis.

5.7.6 Computation Time

The algorithm is implemented in Python using the PyTorch library. The experiments were carried out using an Intel® CoreTM i7K 8700K Processor and a dual NVIDIA® GeForce® GTX 1080 Ti with 11 GB RAM GDDR5X each.

The computational time in Table II refers to the time required to analyze, predict, and detect abnormalities in an instance of visual data. Experiments with 10 and 50 particles

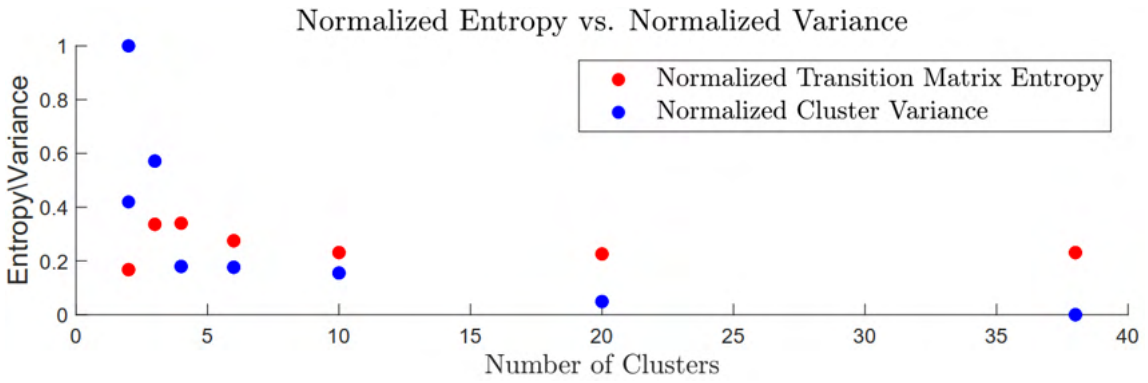


Fig. 5.13 Entropy vs. variance in various GNG training stops.

were performed for the SV and FV. The AUC values refer to the case of using the SV with r_s. Increasing the number of particles improves the accuracy of the method (in particular for the discrete-level anomaly) while requiring more time. When using the SV, a single OF calculation is necessary at the beginning; when using the FV and considering also the OF anomaly, the OF must be calculated twice. The reader should be reminded, however, that the calculation of the OF in the FV is not necessary and was only introduced in the Results section to provide a complete comparison between the two versions of the approach.

The OF calculation on our platform requires 18ms for images of size 240x180, and 88ms for images of size 640x480. For the VAE encoding, the images and OFs were first reduced from 640x480 to 240x160. Additionally, it is important to note how the resampling performed with r_s (in contrast to r_I) does not impact the performance while significantly increasing the speed, as it is not necessary to use the decoder of the VAE to calculate the image-level anomalies for each particle. All the experimental results reported in this work are obtained through an AMJPF (or OF-AMJPF) that uses 50 particles.

5.7.7 Clustering experiments

We perform different clustering experiments on the SV to show how clustering affects the proposed method. We use the GNG algorithm because it adds new clusters during its execution. Consequently, it is not necessary to define the number of clusters a priori, and the clustering process can be stopped when a condition is met.

In Fig. 5.13 we display two examples of discriminators that could be used to stop the clustering: *i)* the variance of the first component of the PCA (Principal Component Analysis) over the GSs belonging to each cluster, averaged over the number of clusters; *ii)* the normalized entropy of the transition matrix. On the x-axis is the corresponding number of clusters. We desire both proposed discriminators to be low and we can stop the clustering

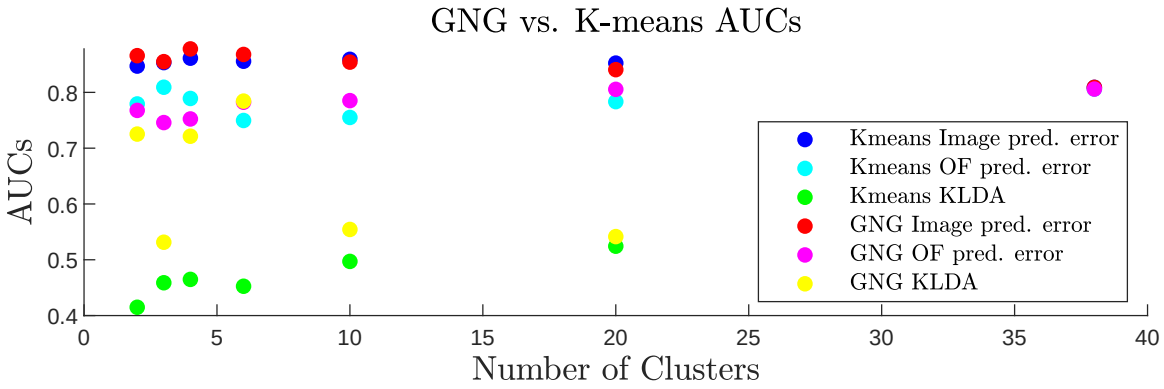


Fig. 5.14 GNG vs. Kmeans clustering AUCs. © 2021 IEEE.

process when they reach a trade-off point. In Fig. 5.14, we show the AUC value for the different proposed anomalies in the above-described GNG experiments. Additionally, we compare these results with the ones obtained when the K-means clustering algorithm is used, setting the same number of clusters as the ones reached with the GNG. We observe that the best GNG results are obtained with 4/6 clusters, around the point in which entropy and variance stabilize. Also, note that K-means generally performs worse. For further comparisons with different clustering methods, we report the reader to [106], as clustering is not the topic of this chapter. Further methods for comparing the clustering levels, inspired by [106], will be adopted in Chapters 7 and 8.

5.7.8 Ablation study

We perform a study to see how the blocks of the method contribute to the performance gain.

Firstly, we consider applying the traditional MJPF on the VAE latent state obtained from the sole camera information (and no OF). This means that the GS prediction is performed with linear models built based on the GNG clustering and not with the neural networks. A KF is employed instead of the AKF.

Conversely, we then remove the linear prediction models (and the KF) and use the prediction neural networks (and the AKF). This case corresponds to the FV.

Finally, we calculate the OF and use it as additional input to the VAE to build the GS. This case corresponds to the SV.

In this way, we can observe how the use of the prediction NNs and of the OF encoder and decoder influence the performance. We display in Tables III, IV and V the AUC obtained with r_S resampling and 50 particles on the PA, U-turn and ES tasks, respectively.

It can be noted that both the addition of the prediction NNs and of the OF tend to positively affect the performance on both anomaly indicators. The only exception is found

Table III Ablation study on PA. The AUC (r_S, 50 particles) with image prediction error and KLDA is shown. In parentheses, we show the improvement of one cell of the table compared to the same cell in the previous row.

	AUC img. pred. err.	AUC KLDA
MJPF	0.72	0.56
+ prediction NNs and AKF (FV)	0.84 (+0.12)	0.58 (+0.02)
+ OF (SV)	0.87 (+0.03)	0.77 (+0.19)

Table IV Ablation study on U-turn. The AUC (r_S, 50 particles) with image prediction error and KLDA is shown. In parentheses, we show the improvement of one cell of the table compared to the same cell in the previous row.

	AUC img. pred. err.	AUC KLDA
MJPF	0.92	0.62
+ prediction NNs and AKF (FV)	0.82 (−0.10)	0.74 (+0.12)
+ OF (SV)	0.91 (+0.09)	0.80 (+0.06)

Table V Ablation study on ES. The AUC (r_S, 50 particles) with image prediction error and KLDA is shown. In parentheses, we show the improvement of one cell of the table compared to the same cell in the previous row.

	AUC img. pred. err.	AUC KLDA
MJPF	0.65	0.55
+ prediction NNs and AKF (FV)	0.69 (+0.04)	0.66 (+0.11)
+ OF (SV)	0.81 (+0.12)	0.81 (+0.15)

on the U-turn case, with the image prediction anomaly, for which a better performance is obtained on the base case. However, this can be explained by the nature and location of the anomalies in this task and by the weakness of the basic MJPF on image data, which is highly non-linear. The basic MJPF tends to perform badly on curve areas, as it cannot handle well the high non-linearity of these regions (more will be discussed on this topic in Chapter 6). Therefore, curving areas tend to be associated with high anomaly indicators. In the U-turn task, conversely to the other tasks, all curving areas correspond to abnormal zones and most abnormal zones are curving areas. Therefore, the basic MJPF shows a better performance on the image prediction anomaly than the other two methods despite learning a worse model.

In the table, we report in parentheses the improvement of one cell of the table compared to the same cell in the previous row. It is worth noting that the addition of the prediction NNs and of the OF improves both the image prediction anomaly and the KLDA. The KLDA

is particularly affected by the use of the OF, as this allows the creation of a more directly explainable GS, leading to better clustering.

5.7.9 Discussion of comparison between FV and SV

From the obtained results, it can be observed that both methods, the FV and the SV, generate a hierarchy of anomalies from less semantic to more semantic levels. The use of the OF as the second part of the GS allows us to increase the performance of the method, due to the OF's ability to directly capture the features related to the type of instantaneous movement of the vehicle and of the environment. Some shortcomings of simple reconstruction methods can be overcome in certain cases, like the possibility that the reconstruction model generalizes to new data or the inability to detect abnormal sequences of motions.

5.7.10 Analysis of the method on the Avenue and Subway datasets

In this section, we propose some additional results on the well-known Subway and Avenue benchmark datasets (see Section 4.2.4). We perform testing using the SV only.

Despite not being an FPV dataset for Autonomous Systems, the Subway dataset is still apt for use with our method, as agents inside the video move and act together with a certain flow and rules of actions. In the following section, we propose an in-detail description of results obtained on the Subway Exit dataset, where a clearer evolution of the flow of events is present, due to the presence of the train entering and leaving the station.

On the other hand, the Avenue dataset is the least apt for our method among the three adopted ones. This is because the actors interact independently, and, aside from a general left-to-right motion in a specific section of the images, there is a lack of explicit flow or governing rules of motion. Therefore, in this dataset, the significance of the highest hierarchical level diminishes. Lower hierarchical levels can be considered only.

Consequently, we don't examine the results on the Avenue dataset in detail but only report the obtained performance in Table VI.

Table VI shows AUC performance for these datasets, with various ground truth (GT) definitions in the case of the Subway dataset. It is worth noting that, as the different anomaly measures in the hierarchy can refer to different types of abnormal events (as observed for the U-turn data too), better results can be obtained by combining them. The combinations mentioned in the Table refer to the simple method of summing the normalized anomalies.

In the next sections, we first analyze in detail the results on the Subway Exit dataset, and then we provide a comparison on the Avenue and Subway datasets with other state-of-the-art methods.

Table VI Table displaying AUC value for direct image reconstruction anomaly, image prediction anomaly and KLDA, for Subway Exit/Entrance dataset and Avenue dataset. *combining the three anomalies, AUC = 0.8825 **combined with OF reconstruction anomaly, AUC = 0.872.

	GT	img. rec. err.	img. pred. err.	KLDA
Exit	Fig. 5.17 original	0.882	0.896	0.775
	Fig. 5.17 additional	0.865	0.879	$0.818(*)$
	[121]	0.902	0.910	0.818
Entrance	[121]	0.731	0.732	0.604
	[48]	0.727	0.737	0.626
Avenue	[151]	$0.862(**)$	0.851	0.671

Analysis of the results obtained on the Subway Exit data

Fig. 5.15 displays the color-coded clustering on the training part of the Exit dataset. Different clusters are formed based on the position of the train, its presence, and its absence. Passengers crossing in front of the camera or leaving the platform form a passenger-related cluster (in green in Fig. 5.15).

Fig. 5.16 shows different normal/abnormal events found in the testing part of the dataset. The normal ones are: a passenger passing in front of the camera, from left to right (1), the train entering the station (6), and passengers leaving the train and platform (7). The abnormal ones, shaded in red, correspond to: the train leaving the station (2), a passenger moving in front of the camera, from right to left (3), passengers going to the platform (4), a passenger leaving the platform in absence of the train (5). Since there is not a clear consensus regarding the dataset portion for training and regarding the ground truth, we tried to remain faithful to the indications of the original author, training on the first 5 minutes of the dataset. Accordingly, Fig. 5.17 displays in dark red the ground truth defined in a window around the original authors' point-wise definitions [1], which correspond to events of type (4). As, however, events of type (2), (3), and (5) are also absent in the training part, we show them in light red in Fig. 5.17. Table VI reports AUC values using the ground truths in Fig. 5.17 and the one defined in [121].

Fig. 5.17 displays the anomalies at the different levels of the hierarchy. Additionally, the different events (A) and the clustering assignment (C) are shown, color-coded as in Fig. 5.16 and in Fig. 5.15, respectively. It can be observed how the image prediction anomaly makes the results slightly better. The use of the KLDA allows us to identify some normal behaviors that were considered abnormal by the pixel-wise anomalies approach; the clearest example is around time instant 22500. Due to the train occupying a big portion of the

Fig. 5.15 First component of the PCA of the training GSs in subway Exit dataset, and corresponding color-coded clusters assignment (C), with examples of related images. © 2021 IEEE.

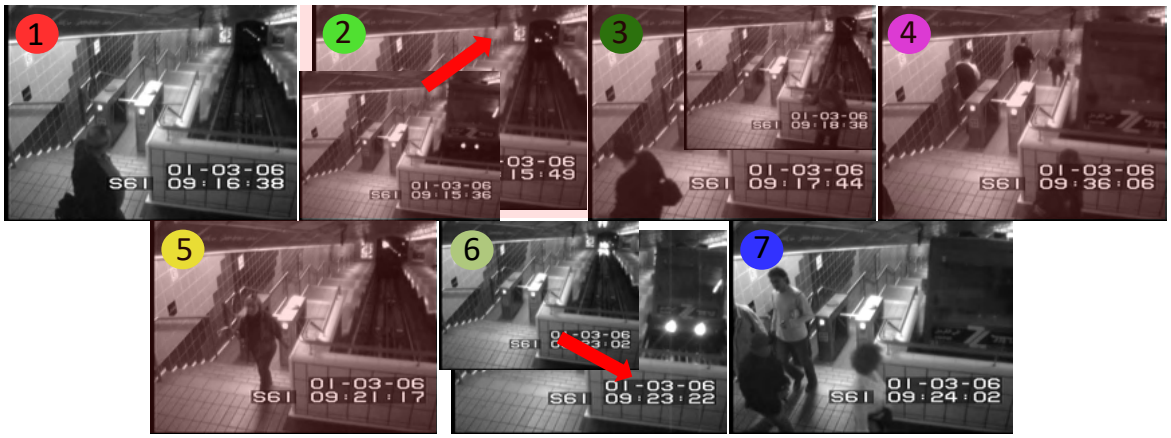


Fig. 5.16 Examples of normal/abnormal events in the testing set of subway Exit dataset. © 2021 IEEE.

image, it generates a high pixel-wise reconstruction and prediction anomaly (despite it being reasonably reconstructed and predicted). However, the succession of events at the conceptual level is normal, and the event does not generate a high KLDA. A similar situation is found for the event in which passengers leave the train and platform, around time instants 23000 and 61500.

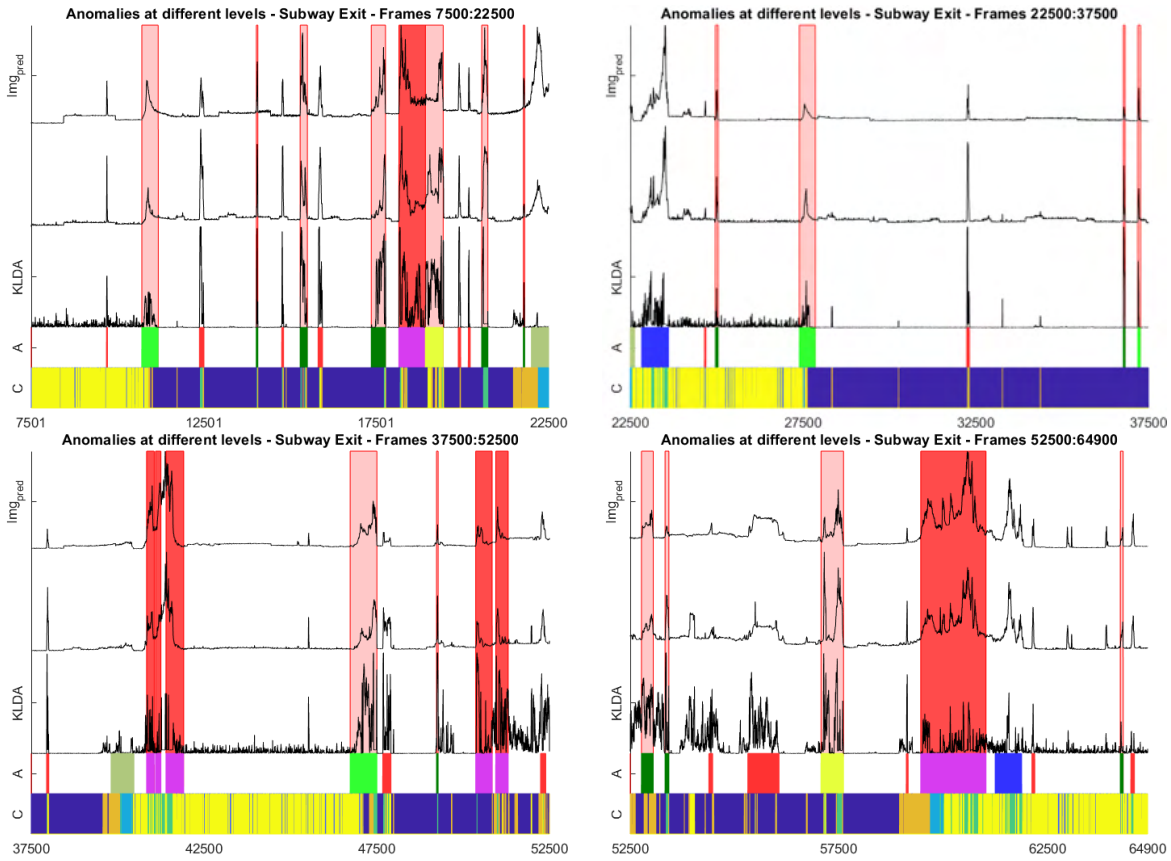


Fig. 5.17 Anomalies at the different levels for the Subway Exit frames. From above to below, in each plot: image reconstruction anomaly; image prediction anomaly; KLDA; abnormal/normal event (A), following the colors defined in Fig. 5.16; cluster assignment (C), following the colors defined in Fig. 5.15. The labels on the x-axis represent the time instants.

5.7.11 Performance comparison with other state-of-the-art methods

Lastly, we compare the anomaly detection results obtained on the Subway and Avenue datasets with other state-of-the-art methods, to show that we compete with them, while also providing a self-awareness method for vehicular data that has desirable characteristics such as interpretability.

Table VI reported a summary of the results obtained with our method. It is worth noting that some anomalies seem to provide better performance than others. However, some anomaly signals are expected to be high in certain situations and not in others. We have just seen an example of this with the KLDA and the observation-level anomalies in the Subway Exit dataset. Therefore, the different anomalies could be fused to improve the performance.

The table shows that better results are obtained in the Subway Exit dataset compared to the Subway Entrance. The reason is that the Exit dataset is governed by clearer flow-like rules

Table VII Comparison of anomaly detection methods. The columns "Avenue", "Exit" and "Entrance" report the AUC of the methods on the respective dataset. The year of publication is reported, and methods are listed according to it. In the sixth column, we analyze if further supervision is given to the method, except for knowing that training data are normal. Methods marked with a "✓" do not use further supervision. Supervision in the other methods is constituted by the use of pre-trained object or interaction detection networks. The last column defines if the method tackles the problem of interpretability/explainability.

Method	Avenue	Exit	Entrance	Year	No additional supervision	Interpretability /Explainability
[93]	0.702	0.807	0.943	2016	✓	✗
[66]	0.630	-	-	2016	✓	✗
[154]	0.817	-	-	2017	✓	✗
[245]	0.771	-	-	2017	✓	✗
[193]	-	0.902	0.904	2017	✓	✗
[39]	0.803	0.940	0.847	2017	✓	✗
[149]	0.849	-	-	2018	✓	✗
[204]	0.892	0.946	0.902	2019	✓	✗
[48]	0.823	0.932	0.806	2020	✓	✗
[187]	0.829	-	-	2020	✗	✗
[210]	0.896	-	-	2020	✗	✗
Ours	0.862	0.910	0.732	2021	✓	✓
[144]	0.866	-	-	2021	✓	✗
[213]	0.753	-	-	2021	✗	✓
[241]	0.875	-	-	2021	✗	✗
[214]	0.883	-	-	2022	✗	✓
[224]	0.883	-	-	2022	✓	✗
[197]	0.869	-	-	2022	✓	✗
[244]	0.849	-	-	2022	✓	✗
[203]	0.860	-	-	2023	✗	✓

than the Entrance dataset: the train is fully visible, and when it reaches the station, pedestrians are expected to exit the platform; on the other hand, when the train is absent, pedestrians are not expected to move inside this one-way platform. Conversely, in the Entrance dataset, only a small angle of the train appears in the camera, and crowd movements are more complex. Therefore, a cluster-based representation of the entire scene is more appropriate in the Exit scenario than in the Entrance.

For the sake of comparison with other state-of-the-art datasets, we select, for each dataset, our anomaly signal that provided the best results (typically, the image prediction error). Table VII compares this result with methods proposed in other approaches. The ground truth

adopted for the Exit and Entrance datasets is the one in [121]. Methods are illustrated in order of publication year. We analyze methods published both before and after ours.

We can observe how our method provides results that are competitive with the ones of the state of the art. Up to its publication date, our method displayed among the best results on the Avenue dataset. It also demonstrates a better performance than some methods published in the following years. Competitive results are obtained on the Subway Exit dataset too. Conversely, our method has poor results on the Subway Entrance dataset, for the reasons already discussed.

All methods in the table are DL-based, using either Convolutional Autoencoders (e.g., [93, 48]), Deep Gaussian Mixture Models (e.g., [66, 210]), CNNs (e.g., [193, 149, 224, 214, 203]), CNNs and RNNs (e.g., [154]), Spatiotemporal Autoencoders (e.g., [245, 39]), Attention-based Autoencoders (e.g., [204, 244]), Conditional VAEs (e.g., [144]), Graph Convolutional NNs (e.g., [241]), GANs (e.g., [204]), or fully connected NNs (e.g., [187]). The methods in [210] (the one displaying the highest performance on the Avenue dataset), [187], [213], [241], [214], and [203] present some supervision, as they work on objects' trajectories, after employing a pre-trained object detector. Therefore, they are not in line with the self-aware philosophy of this thesis, as they suppose that notions such as objects or pedestrians are known, instead of learning concepts and rules through experience. However, for completeness, we also report results from this type of methods.

Among the methods in the table, only [213], [203], and [214] tackle the problem of interpretability/explainability. In [213], this is performed by encoding, for each pair of objects in a frame, an Object Interaction Vector (HOI) containing the box size, object class score, and interaction probabilities. HOIs of the training data are employed to train a GMM. Each HOI of the testing data is compared with the GMM and is considered abnormal if it falls below a threshold probability under the GMM. The closest mixture component is identified, and a saliency map is built, which allows the recognition of the part of the HOI responsible for the anomaly. For example, it is possible to distinguish if the anomaly is due to an object, or to an interaction between objects, and which kind of interaction. However, this approach uses both a pre-trained object detector and a pre-trained interaction classifier, which deviates from the underlying philosophy of this thesis.

The method in [214] is proposed by the same authors of [213]. The authors observed how the use of the pre-trained models was a limitation confining the detection to a limited set of known classes and preventing the generalization to unknown situations. Therefore, they proposed an encoder-decoder architecture (with a vector quantization module) that derives anomaly detection saliency maps and provides an interpretation of them. However, object and interaction detectors are still employed in the final part of the method. The values of the

saliency maps are summed inside the bounding boxes where objects are detected to derive a per-box anomaly score. The anomaly explanation is again obtained from analyzing the classes of objects and interactions that correspond to high anomaly scores. Compared to the previous method, the anomaly is detected also without the pre-trained detectors, but these are necessary for the explainability of the method. Therefore, this work does not align with the philosophy of this thesis either. Firstly, it still relies on pre-trained models, moving the problems noticed by the authors from the overall method to only the second part of it (the anomaly explanation). Secondly, it leans more towards offering post-hoc explanations for anomalies rather than embracing an interpretable model as a primary approach.

The most recent among the papers in the table, i.e., [203], published in 2023, also uses pre-trained models. First, a set of networks for object and motion detection are pre-trained, taking as input spatio-temporal regions of the video. Then, the training data embeddings from these networks are used to learn a set of training exemplars. During testing, embeddings of spatio-temporal regions are compared with all exemplars in the same area. Those embeddings displaying a high distance from all the corresponding exemplars are classified as anomalies. Since embeddings encode human-interpretable information, they can be used to explain the anomalies. Therefore, the philosophy of this approach is very similar to the one of [214], but again diverges from the one discussed in this thesis.

In this thesis, we adopt DBNs, which are interpretable models by nature, as it is possible to express what is happening at their different levels. Furthermore, we can locate at which level of the DBN an anomaly is detected, by analyzing which anomaly signal displays high values compared to training. Notably, unlike other anomaly explanation methods, our approach does not rely on pre-trained models.

5.8 Conclusions

This chapter presented an approach for predicting future time instances and detecting anomalies in visual data. Using a VAE, we were able to bring high-dimensional data acquired from a camera to a low-dimensionality compatible with other sensory data acquired from the vehicle (e.g., position, steering angle). Two versions of a filter joining a VAE, KFs, UKFs, and a PF were introduced, i.e., the AMJPF and the OF-AMJPF. In this way, the two worlds of NNs and DBNs were combined.

The insertion of the OF for the generation of the GSs was shown to improve the accuracy of the method (especially at the discrete level) compared to the version of the method directly learning the motion information from camera data latent states. However, the

calculation of the OF increased the computation time and added a model-based component to a predominantly data-driven framework.

Anomaly signals at multiple levels of the DBN hierarchy were extracted. When faced with a novel situation, the method could detect at which level anomaly signals were high compared to training. These signals can already suggest the cause of an anomaly at a general level. For example, an anomaly at the discrete level could be caused by an abnormal sequencing between clusters; a high direct reconstruction anomaly could be caused by a new object, or by a known object in an abnormal position (camera frame anomaly), or by a new type of motion (OF anomaly); a high prediction reconstruction anomaly accompanied by a low direct reconstruction anomaly signals the presence of known objects in unexpected situations (camera frame anomaly) or known motions in unexpected situations (OF anomaly). More anomaly types could be introduced in the future. Observation anomalies could be analyzed further at the frame level, for example, to identify if the detected anomalies are vast and sparse (as for a new environment) or concentrated (as for new objects).

The method is aimed at autonomous vehicles. However, we have also evaluated it on static-camera benchmark datasets for anomaly detection. The method is competitive against state-of-the-art approaches, especially in environments with clear flow-like motion rules. A fusion method to combine the different anomaly signals should however be derived to have a more significant comparison. As anomaly signals at different levels can display high values in distinct situations, a rule to combine them should be deduced. The fusion of anomalies and the introduction of further anomaly signals are not tackled in this thesis.

Furthermore, our method is integrated within a multi-modal architecture designed for autonomous vehicles. The insertion of observations coming from other sensory data, e.g., positional and control data, into the proposed probabilistic framework, will be considered in the next chapters. By allowing algorithms to handle multi-modal data when making predictions and detecting anomalies, it is possible to generate more realistic agents that associate multiple heterogeneous signals to familiar concepts as humans and other animals do. Therefore, first, in Chapter 6, we analyze low-dimensional data and propose a method that can be applied to it. In Chapters 7 and 8, novel approaches for combining low-dimensional and high-dimensional data are proposed. Moreover, these two chapters also cover one inhomogeneity in the treatment of high-dimensional data compare to low-dimensional data in our method, i.e., the use of neural networks for state prediction instead of piece-wise linear models.

Chapter 6

Interpretable anomaly detection using a Generalized Markov Jump Particle Filter

This chapter presents a method for interpretable anomaly detection on low-dimensional data. As we have explored in previous chapters, when performing anomaly detection on the trajectory (or other sensory data) of an autonomous vehicle, it is fundamental to interpret the found anomalies. The approach is based on the self-awareness framework presented in Chapter 3 and proposes an extension of the MJPF with the main aim of improving the interpretability. This is achieved by decomposing the evolution of an agent's state into its motion towards the objective it is trying to reach (the attractor) and into the oscillation around this first motion.

Furthermore, we consider to be provided with data from different types of driver behavior. We train multiple models, one for each type, and we extract the salient discriminatory features of each. The proposed anomaly detection method is extended to perform behavior classification.

Section 6.1 introduces the considered problem. In Section 6.2, we describe the proposed method for anomaly detection, and in Section 6.3 we discuss its extension for driver behavior classification. Section 6.4 discusses the obtained results. Finally, Section 6.5 draws some conclusions.

6.1 Introduction

The learning of a model describing how the state of an agent evolves across time has many purposes: using the model to perform short-time or long-time prediction; classifying an agent based on what model it follows; performing anomaly detection distinguishing when the rules

of the model are broken or when they are respected. Such applications are of interest in a variety of fields, from autonomous driving, to self-aware radios [132], to video surveillance [152], to weather forecasting [201], to medical image analysis [242].

In Section 3.5, we have seen that also the concept of interpretability is important when discussing self-aware agents. Interpretable approaches should include causality, decomposability of the individual parameters, and algorithmic transparency. DBNs possess all these characteristics. In Chapter 5, we have also discussed how we can leverage the hierarchical nature of DBNs to provide a basic explanation of anomalies, and we have proposed an approach for camera data. Indeed, in the field of anomaly detection, it is relevant not only to identify the time and location of an abnormal event but also to ascertain which model rules were violated. Hierarchical models are apt for this purpose, as they allow us to distinguish variables that are more directly related to the sensory observation (at the lowest levels of the hierarchy) or to more conceptual representations (at the highest levels of the hierarchy). Therefore, this allows distinguishing where the anomaly is located in the hierarchy too, which, in turn, identifies on which models we should execute incremental learning.

Friston proposed the use of Hierarchical Generalized State Filters (i.e., hierarchical filters employing Generalized States) and adopted the concept of linear attractors [77]. The motion among attractors can be described through flow functions. The flow can be decomposed in a curl-free component and in an orthogonal divergence-free component. Accordingly, in this chapter, two types of motions are considered: the one in the direction of the attractor and the one in the orthogonal direction. In [104], Iqbal et al. the model associated with each cluster is connected with a linear attractor inspired by Friston. Similarly, in this chapter we identify the learned clusters as the attractors. The first type of motion can be described as the motivation that the agent pursues and how it moves toward the attractor. In contrast, the second type of motion can be connected to the modality with which the agent reaches the attractor along the orthogonal direction, e.g., smoother when the agent is an expert in its task, and oscillating when it is uncertain. Therefore, the features related to the two directions of motion can be used to identify a particular behavior and to determine the abnormality of that behavior compared to an unknown one. When multiple behavior models are known, the discriminatory features of each can be extracted and anomalies used for classification. This has been used, for example, in driver behavior analysis [37, 232, 25]. One of its main objectives is to distinguish dangerous drivers from safe ones. Oscillating and uncertain drivers fall into the risk category of driving behaviors [37]. Drivers that are oscillating or drifting from their lane could be drowsy. A drowsy driver can be alerted from the autonomous vehicle, thus reducing the peril of a traffic accident [234]. Driver behavior classification can be executed by finding thresholds where typical driver behavior is located. Driving behavior

classification can be performed by setting hard rules that classify the different behaviors. For example, an aggressive driver might be recognized by fixing rules on acceleration, speed, or braking; a drowsy driver can be detected by hard-coding a test to detect an oscillating or drifting motion between lanes. However, setting thresholds on these rules can be difficult. Therefore, in recent years, research has studied methods that learn to classify driver behavior in a data-driven way instead of hard-coding rules [234].

In this chapter, we propose an extension of the MJPF presented in [17], to create more precise rules describing the evolution of the state of an agent and with the objective of treating the study of the evolution along the direction of motion together with the evolution along its orthogonal direction. Tests are conducted on two-dimensional real data by employing the iCab dataset [157] and the UAH DriveSet dataset [190].

The main contributions of this chapter are the following:

- the tracking of parameters at the base of the vehicle's motion and their use to predict the next state. This allows to improve the interpretability of the model, increasing the decomposability. This chapter extends the previous work to build a more flexible and interpretable model based on a four-level DBN. We call this model Generalized-MJPF (G-MJPF).
- the extraction of the features related to the direction perpendicular to motion using the concept of *vorticity* and their usage to define an anomaly related to driver experience/uncertainty.
- the recognition of discriminatory features of a behavior class and the use of the extracted anomalies for behavior classification purposes.

6.2 Method Description: anomaly detection

In this section, we explain the main method, aimed at anomaly detection. Its extension to driver behavior classification is described in Section 6.3.

The method can be split into two parts: the training and the testing phase. During the training phase, two filters are learned incrementally, a rotational MJPF (R-MJPF) and a basic MJPF, which are joined to form the G-MJPF. During the testing phase, we apply the G-MJPF on the testing dataset to detect anomalies relative to the learned model. As anomalies identify where the learned model is not coherent with the object's actual motion rules, a new filter could be learned using the extracted model errors. A scheme of the method is illustrated in Fig. 6.1.

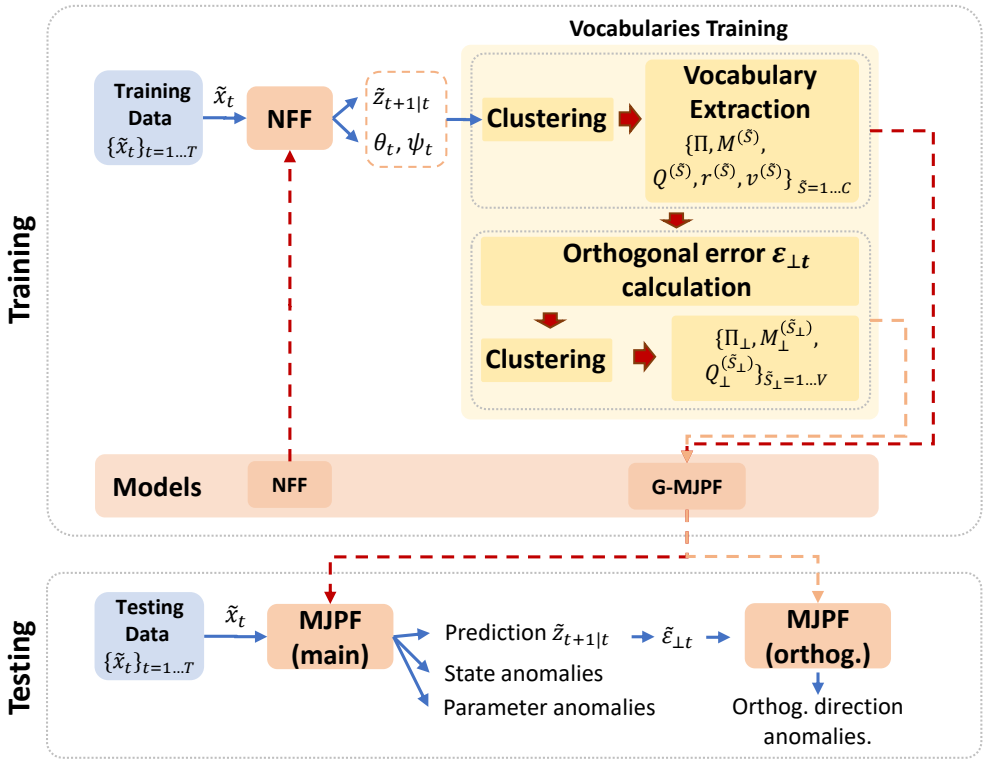


Fig. 6.1 Training and Testing phases of the G-MJPF. © 2021 IEEE.

6.2.1 Training phase

Null Force Filter and Generalized States

Let us suppose to be given a training dataset composed of T consequent observations $\{x_t\}_{t=1...T}$ from a sensor. In this chapter, we assume to be working with low-dimensional observations, such as positional data from a GPS.

As an initial step, we track the evolution of the GSs $\{\tilde{z}_t\}_{t=1...T}$ associated with the observations, using an NFF. We remind the reader that an NFF is a KF supposing that the agent is not affected by any force and that it continues in its motion with unmodified speed compared to the previous time-step. The GS at time t can be defined as:

$$\tilde{z}_t = [z_t \ \dot{z}_t]^\intercal = [z_t \ v_{\|t}]^\intercal, \tag{6.1}$$

where $\dot{z}_t$ approximates the first-order derivative of z_t. When we observe positional data, $\dot{z}_t$ corresponds to the velocity in the direction towards the attractor. For this reason, in this chapter, we also use the alternative notation $v_{\|t}$ for it. The symbol "$\|$" identifies that the motion is towards the attractor and not orthogonal to it. We can link the GSs and the observation through a transition matrix H, as in Eq. 3.3, which we report here for

completeness and readability:

$$\dot{z}_t = \frac{x_t - Hz_{t-1}}{\Delta t} \tag{6.2}$$

The dynamic model supposed by the NFF can be expressed through the following equation:

$$\tilde{z}_{t+1} = A\tilde{z}_t + \omega_t, \tag{6.3}$$

where $A = [A1, A2]$, with $A1 = [I_{d,d}, 0_{d,d}]^\intercal$ and $A2 = [0_{d,d}, 0_{d,d}]^\intercal$. $I_{d,d}$ is the identity matrix with d rows and columns, where d is the observation dimension. Instead, $0_{d,d}$ is a dxd matrix of zeros. In the case of the trajectory data, $d = 2$ for $2D$ data and $d = 3$ for $3D$ data. In this chapter, we assume to be working with 2D observations.

Therefore, at each time step in which the NFF is applied, we can extract the desired motion parameters $\tilde{\mu}_t$, which allow us to correct our model, coherently with the free energy principle defined by Friston [76]. One possible way of describing the correction is by using the velocity part of the latent state, as we have already seen in Eq. 3.10:

$$\tilde{z}_t = A\tilde{z}_{t-1} + BU^{(\tilde{S}_t)} + w_t, \tag{6.4}$$

where $B = [0_{d,d} I_{d,d}]$ is the control matrix projecting the velocity information $U^{\tilde{S}_t}$ on the GS.

In this chapter, instead, we propose to perform the correction through a different model. Eq. 6.3 could, therefore, also be corrected as:

$$\tilde{z}_{t+1} = A\tilde{z}_t + (\Phi + \Psi)\tilde{z}_t + \dot{\Psi} + \omega_t, \tag{6.5}$$

Let us now discuss what Φ, Ψ and $\dot{\Psi}$ are. Firstly, Φ represents a rotational correction and Ψ an acceleration correction. $\Phi + \Psi$ can be modeled as:

$$\Phi + \Psi = \begin{bmatrix} 0 & 0 & 1 - cos\theta_t + \psi x_t & 0 \\ 0 & 0 & 0 & 1 - sin\theta_t + \psi y_t \\ 0 & 0 & 1 - cos\theta_t & 0 \\ 0 & 0 & 0 & 1 - sin\theta_t \end{bmatrix}, \tag{6.6}$$

being θ_t the rotation angle and being $\psi_t = [\psi x_t, \psi y_t]$, i.e. the accelerations to add along the d dimensions.

Secondly, $\dot{\Psi}$ can instead be modeled as $\dot{\Psi} = [0_{d,d}, \psi_t]^\intercal$. We extract the rotation angle first, and then we estimate the acceleration from the remaining error present in the model. In this way, we have extracted the parameters that define our rule of motion over $\tilde{z}_t$, i.e., $\tilde{\mu}_t = \{\theta_t, \psi_t\}$.

Clustering of the Generalized State and of the motion parameters

After performing testing with the NFF, and obtaining the GSs and motion parameters along the direction of attraction, i.e., $\{\tilde{z}_t, \tilde{\mu}_t\}$, we use the GNG [78] algorithm to cluster them. Therefore, clusters group together similar states characterized by similar positions and rules of motion.

We associate to each cluster $\tilde{S} = 1 \dots C$ a mean value $M^{(\tilde{S})}$ and a covariance $Q^{(\tilde{S})}$. A transition matrix Π describes the probability of transitioning between clusters. The model can also include the Temporal Transition Matrices containing the probability of moving between clusters, having already spent a certain number of time instants in the current cluster.

Additionally, for each cluster $\tilde{S}$, the rotation center $r^{(\tilde{S})}$ and the mean velocity magnitude $|v|^{(\tilde{S})}$ are extracted. The rotation center $r^{(\tilde{S})}$ is obtained by supposing to move with a θ_t equal to the mean $\theta^{(\tilde{S}_t)}$ of the cluster. Additionally to the center of rotation, also the orientation of the rotation is saved for each cluster, i.e., $o^{(\tilde{S})}$. The orientation identifies if the agent moves clockwise or counterclockwise compared to the rotation center.

Redefinition of the prediction model

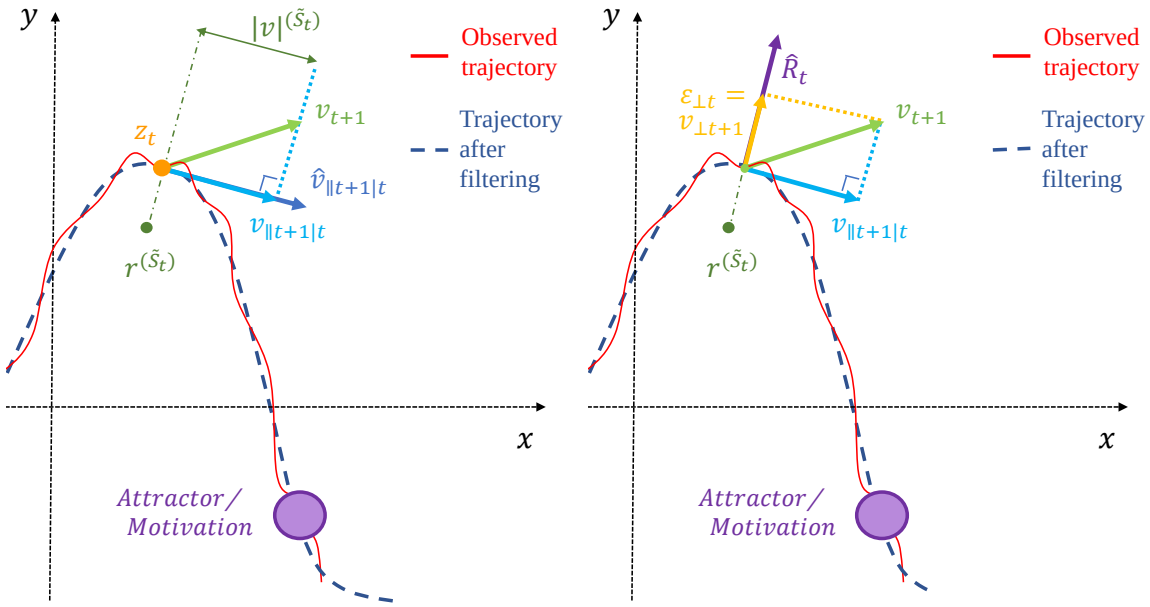
The prediction model can be reformulated again, so that, during testing, the newly-built model can be employed to perform prediction. The reader can refer to Fig. 6.2a to understand the process described in the next paragraphs. The figure illustrates an example of the trajectory of an agent (i.e., the line in red) moving towards its motivation (i.e., the circle in violet). The dotted blue line is the trajectory after the filtering process.

We initially determine the normal to the line connecting the rotation center of the current cluster ($r^{(\tilde{S}_t)}$) and the current position (z_t). This normal is defined as $\hat{v}_{\|t+1|t}$ in the figure. Then, we multiply it by the mean velocity magnitude of the current cluster ($|v|^{(\tilde{S})}$), obtaining $v_{\|t+1|t}$.

The prediction equation can consequently be reformulated through a non-linear function f written as follows:

$$\tilde{z}_{t+1} = f(\tilde{z}_t, r^{(\tilde{S})}, |v|^{(\tilde{S})}) + \omega_t, \tag{6.7}$$

This model allows considering the possibility of having a primary direction of rotation while not depending on the velocity at the previous time instant, conversely to Eq. 6.5, which made the model sensible to noise and less robust for anomaly detection purposes. It is worth noting that $r^{(\tilde{S})}$ is extracted after clustering and not while clustering, to avoid a noisy measurement of it. If clusters with similar motions and in close positions happen to have different rotation points, clustering can be refined, considering $r^{(\tilde{S})}$ as an additional clustering input.



(a) Direction towards attractor.

(b) Orthogonal direction.

Fig. 6.2 Geometric representation of the variables relevant for the prediction along the direction towards the attractor (a) and along the orthogonal direction (b).

Extraction of the features for the orthogonal space

Through the definition of the clusters and their respective type of motion, the features related to the motivation that guides the agent have been extracted. Based upon the attractor's theory of Friston [77], also the motion orthogonal to the direction towards the attractor can be modeled. These features are not related to an attractor but rather to the type of agent performing the motion, e.g., how expert or uncertain it is. We define this oscillation in the direction perpendicular to motion with the name of *vorticity*, inspired by the homonym concept used in fluidodynamics.

For each time instant t of the training data, based on the assigned clusters, prediction is performed using $r^{(\tilde{S})}$ and $v^{(\tilde{S})}$ as defined in Eq. 6.7. Consequently, the predicted velocity of the agent towards the attractor is found, i.e., $v_{\|t+1|t}$. We direct the reader to Fig. 6.2b to better understand the passages described in the next paragraph.

The error related to the prediction performed with Eq. 6.7 is extracted and projected along the direction $\hat{R}_t$, which is orthogonal to $v_{\|t+1|t}$. We define this orthogonal error as $\varepsilon_{\perp t}$. A Generalized Error (GE) is defined from it by considering its first-order time derivative $\dot{\varepsilon}_{\perp t}$:

$$\tilde{\varepsilon}_{\perp t} = [\varepsilon_{\perp t}, \dot{\varepsilon}_{\perp t}]^{\intercal} \tag{6.8}$$

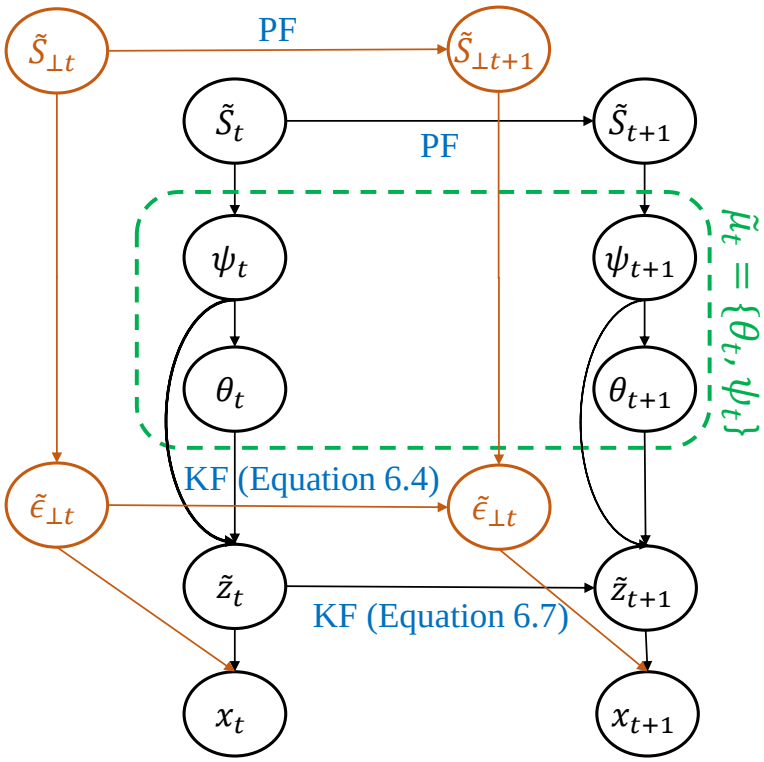


Fig. 6.3 DBN structure of the proposed G-MJPF. © 2021 IEEE.

Clustering of the features for the orthogonal space

Clustering using GNG is then performed on the GEs. Consequently, a set of V clusters $\tilde{S}_\perp = 1 \dots V$ is extracted, each associated with a mean value $M_\perp^{(\tilde{S})}$ and a covariance $Q_\perp^{(\tilde{S})}$. A transition matrix $\Pi_\perp$ is calculated too. Temporal transition matrices can be computed as well.

In this way, we have built also a model for the orthogonal error.

Learned Dynamical Bayesian Network

We have now acquired the vocabulary for our G-MJPF, a process akin to learning a DBN. Fig. 6.3 displays the learned DBN: in black, we show the variables related to the motion along the direction to the attractor, highlighting in green the parameters at the base of motion; in orange, we display the variables connected with the normal to the direction towards the attractor; the writings in blue define the filter used to perform prediction at the considered level. It is worth noting that the level related to ψ_t is reported below the one related to θ_t, as we suppose for the rotation angle θ_t to be extracted first, and for the acceleration ψ_t to be derived as the remaining error.

6.2.2 Testing phase

During the testing phase, anomaly detection is performed on a testing dataset. In [17], as discussed in Section 3.7, a MJPF was used for anomaly detection. In this chapter, we propose a modified version of the MJPF, with more precise clustering and general motion rules, allowing the tracking of motion along the direction of attraction and along its normal. In the following description, we will consequently concentrate on the differences between the basic MJPF and the G-MJPF.

Generalized Markov Jump Particle Filter

The G-MJPF is composed of two subparts: an R-MJPF and a basic MJPF. At each time instant, firstly, the R-MJPF is employed and, secondly, its prediction error is provided as input to the basic MJPF.

As in the basic MJPF, two steps are performed: *prediction* and *update*.

During the prediction phase, at each time instant t, based on the cluster associated with each particle of the PF, we perform a KF prediction of the GSs $\tilde{z}_{t+1|t}$ with the prediction model in Eq. 6.7. The KF is not optimal in this case, because the model is non-linear, but we adopt it as a first solution. As in the basic MJPF, prediction at the cluster level is performed using the transition matrix Π, which, in this case, is built through clustering over both GSs and parameters.

During the update phase, at each time instant $t+1$, the motion parameters θ_{t+1} and ψ_{t+1} are estimated as in the NFF, and the state prediction $\tilde{z}_{t+1|t}$ is corrected based on the sensor observation, similarly to how performed in the basic MJPF, [17]. Anomalies are then extracted. Additionally, the orthogonal error ε_t is found for each particle prediction. The error of the particle with the highest weight is given as input to the parallel MJPF for tracking along the orthogonal direction $\hat{R}_t$. Filtering in this MJPF is performed exactly as in the basic MJPF [17].

Multi-level anomaly detection

During the update phase of the two parallel MJPFs, anomalies can be extracted on all levels of the hierarchical DBN.

Using direct mean subtraction between prediction and update or probabilistic measures based on the Bhattacharyya distance as in Section 3.7.4, anomalies on $\tilde{z}_{||t}$, θ_t, ψ_t and $\tilde{\varepsilon}_t$ can be extracted. At the cluster level, KLDA can be calculated, on both the R-MJPF and on the basic MJPF.

Table I Comparisons between the basic MJPF [17] and the G-MJPF. © 2021 IEEE.

	Prediction model	Clustering	Parameters ($\tilde{\mu}_t$) filtering	Orthogonal ($\tilde{\varepsilon}_{\perp t}$) filtering
Basic MJPF [17]	linear	based only on the GSs	absent	absent
G-MJPF	rotational (R-MJPF)	based on the GSs & motion parameters (R-MJPF)	present for improved interpretability	present for improved interpretability

Comparison between the MJPF and the G-MJPF

Table I summarizes the differences between the basic MJPF and the proposed G-MJPF, from training to testing.

Firstly, let us address the first two columns in the table. In the basic MJPF, clustering is performed solely on the GSs and a linear prediction model is applied. In contrast, the first component of the G-MJPF, i.e., the R-MJPF, involves clustering on the GSs and on the motion parameters and uses a rotational model. This allows for representing the training data with fewer clusters while obtaining a precise prediction. To elucidate this concept, let us consider the example of a moving vehicle. Such a vehicle can either travel in a straight line or follow a curved path. When extracting clusters on velocity only, and using a linear prediction model as in the basic MJPF, the treatment of curvilinear segments offers two distinct approaches. These segments can either be subdivided into many small clusters, each approximating the curving motion with a straight motion along distinct directions, or they can be represented through a single cluster with a less precise prediction model. Conversely, when extracting clusters on the rotation angle too, and adopting a rotational prediction model, a single cluster with precise prediction can be obtained.

The second difference between the two filters pertains to the variables on which clustering and filtering are performed. In the table, this is illustrated by the third and fourth columns. The MJPF performs filtering only on position and velocity. On the other hand, the G-MJPF adopts the motion parameters ($\tilde{\mu}_t$) and the error on the orthogonal direction ($\tilde{\varepsilon}_{\perp t}$) too. By decomposing the motion along its different directions and parameters, it is now possible to more precisely identify the variables associated with high values of the anomaly signals. Therefore, we can pinpoint which aspect of the vehicle motion is generating anomalies. This increases the interpretability of the method.

To summarize, the first two differences are aimed at obtaining a more accurate model, whereas the second two differences are intended to achieve a more interpretable approach.

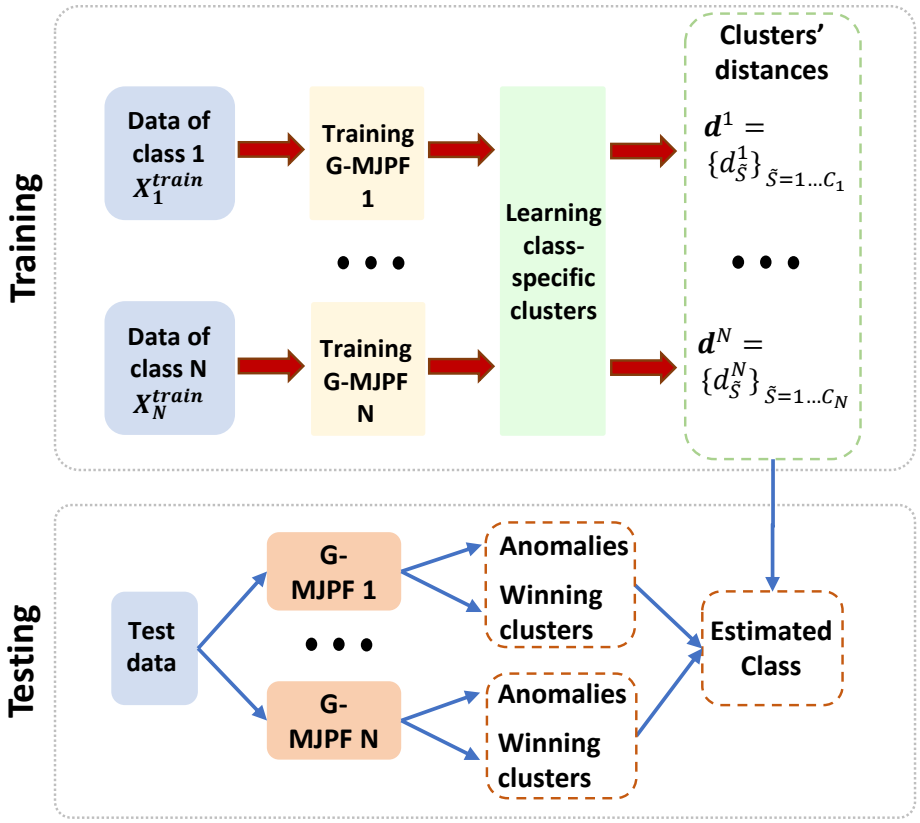


Fig. 6.4 Proposed approach for driver behavior classification. © 2021 IEEE.

6.3 Method Description: Driver Behavior Classification

The method proposed in the previous sections for anomaly detection, can be extended for driver behavior classification. Fig. 6.4 illustrates the approach, which is also subdivided into a training phase and a testing phase.

6.3.1 Training Phase

The objective of driver behavior classification is to identify how a vehicle is moving, choosing among a set of N classes. Therefore, during training, we must be provided with N datasets $X_1^{set} \ldots X_N^{set}$, one per behavior class.

Firstly, a G-MJPF is built on each dataset, following the training phase described in Section 6.2.1.

The found clusters of each set constitute a graph, as observed by Zaal et al. [239]. The graph corresponding to a set of clusters can be derived from the transition matrix Π of the R-MJPF. Each cluster corresponds to a node in the graph. Edges are present between two clusters i and j that correspond to a non-null value in the elements π_{ij} of the transition matrix.

Consequently, we have N graphs $G_1, \ldots, G_N$. A Graph Matching procedure can be performed on each couple of graphs to find which clusters correspond to each other in the two graphs. When performing each comparison, we call one graph the "source graph" and the other one the "target graph". A source graph G_s is compared with each of the N-1 target graphs $G_{ta,1}, \ldots, G_{ta,i}, \ldots, G_{ta,N-1}$. We match the corresponding clusters by finding, for each cluster of G_s, the cluster of $G_{ta,i}$ with the smallest Euclidean distance. This distance is saved. Being C_s the number of clusters of G_s, a vector of C_s euclidean distances is found, that we name as $\mathbf{d}^{(G_s G_{ta})} = \{d_{\tilde{S}}^{(G_s G_{ta})}\}_{\tilde{S}=1\ldots C_s}$.

The calculation of this vector of minimum distances is repeated for each of the N-1 target graphs.

Then, we compute, for each cluster of G_s, the average distance across the N-1 sets $\mathbf{d}^{(G_s G_{ta})}$. It is worth paying attention to the fact that the average is calculated across the N-1 target graphs and not across the C_s clusters. In this way, we obtain a distance value for each cluster of the source graph. These scalars form a vector of normalized cluster distances $\mathbf{d}^{G_s} = \{d_{\tilde{S}}^{G_s}\}_{\tilde{S}=1\ldots C_s}$ of G_s. The highest distances in the set correspond to clusters that are more specific to the corresponding source graph, i.e., to the class.

This procedure is repeated to obtain the distance vector for each of the N graphs.

6.3.2 Testing Phase

When given a new dataset to perform classification, the N G-MJPFs are applied in parallel to it, and the corresponding anomalies at the different levels are extracted. The sequence of winning clusters is also memorized. The winning cluster at time t is the one with the highest weight in the PF. Each anomaly a_t extracted from the i-th G-MJPF is modified as follows:

$$a_t = a_t * (d^i_{\tilde{S}_t} / max(\mathbf{d}^i) * \alpha + \beta), \tag{6.9}$$

where $d^i_{\tilde{S}_t}$ is the distance associated with the i-th G-MJPF and with the cluster $\tilde{S}_t$ assigned to the considered particle of the PF. Furthermore, $max(\mathbf{d}^i)$ corresponds to the maximum distance in the vector $\mathbf{d}^i$. The scalars α and β are used to weight the impact of the distances on the original anomalies.

The use of the cluster's distances allows us to give more importance during the classification to those clusters that have been identified as specific to the particular class. For each G-MJPF, anomalies across levels are normalized and averaged over all time instants. The G-MJPF displaying the lowest final anomaly corresponds to the final estimated class.

The α and β values in Eq. 6.9 are set by performing testing on the training data and by doing a grid search. The N G-MJPFs are tested on the N training datasets as described

above. The anomalies a_t are calculated by adopting different combinations of values for α and β. The obtained class assignments with the different combinations are compared with the ground truth. During training, the ground truth is available, as each trajectory is assigned a driver behavior class. The couple of α and β giving the highest classification accuracy is employed during testing too.

6.4 Results

6.4.1 Employed Evaluation Datasets

To test the proposed method, we use the iCab [157] and the UAH-DriveSet datasets [190], which were introduced in Sections 4.2.1 and 4.2.2, respectively.

The iCab dataset is a dataset primarily developed for anomaly detection. It is employed in this chapter to evaluate the first part of the proposed method, i.e., the anomaly detection, in particular from the point of view of interpretability.

The UAH-DriveSet dataset, instead, is a dataset for driver behavior analysis. As examined in Section 4.2.2, it is composed of various car sensory data from six drivers performing two routes (motorway and secondary road) with three types of behaviors (normal, drowsy, and aggressive). In the drowsy case, the drivers were asked to "simulate slight sleepiness, which normally results in sporadic unawareness of the road scene" [190]. In the aggressive case, the driver was told to "push to the limit his aggressiveness (without putting the vehicle at risk), which normally results in impatience and brusqueness while driving" [190]. Drowsy behavior was observed to often result in lane drifting and lane weaving. Lane drifting happens when the driver drifts from his lane into the adjacent ones or oscillates inside his lane. Lane weaving, instead, is the maneuver of quickly switching back and forth between lanes. Lane weaving was observed to be a behavior typical of aggressive driving too. Other characteristics of aggressive driving were general over-speeding, and brusque motions when braking, accelerating, and turning. However, Arroyo et al. [190], who extracted the dataset, also noticed that the behavior was very dependent on the driver. Whereas some drivers executed dangerous and aggressive maneuvers when asked to perform the aggressive scenario, others were more cautious and implemented the requested aggressiveness only by over-speeding [190].

In this chapter, we used the GPS and accelerometer data of five drivers from the dataset's motorway road. The data of the third driver (out of the six) was excluded from our study because of irregular patterns in the accelerometer data.

Due to GPS having a sampling rate of 1 Hz only, we used a KF combining GPS and accelerometer data (10 Hz), to obtain a 10 Hz estimation of the trajectory data.

We employed MATLAB both for preprocessing the GPS and IMU data, and for training the proposed models.

6.4.2 Anomaly detection on the iCab data

We use the PM data to perform training and we employ the PA, U-turn, and ES data to perform anomaly detection.

Fig. 6.5 displays the anomalies at the different levels of the DBN for the PA (Fig. 6.5a), U-turn (Fig. 6.5b) and ES (Fig. 6.5c) scenarios. Each subfigure is composed of two separate plots: the upper plot illustrates the state innovation and the KLDA (i.e., the state-level error and the discrete level anomaly); the lower plot illustrates the errors at the level of the additional motion parameters $\tilde{\mu}_t = \{\theta_t, \psi_t\}$ and of the orthogonal state $\varepsilon_{\perp t}$. Therefore, the upper plot identifies anomalies at a more general level, detecting if the state or the sequence between the clusters is abnormal. In contrast, the lower plot allows us to analyze more in detail the anomaly type, pinpointing if it is caused by an abnormal rotation or by an abnormal acceleration.

Let us analyze the three cases one by one, starting from the PM scenario (Fig. 6.5a). The blue areas correspond to the pedestrian avoidance maneuver, during which the vehicle performs a semi-circular motion around the pedestrian to avoid it (see Fig. 4.16b). The KLDA anomaly increases in these two areas due to the abnormal sequencing between clusters. However, it cannot explain what element of the vehicle motion is abnormal. Instead, if we analyze the motion parameters, as expected, a high rotational error is present. Moreover, at the beginning and at the end of the rotation, the vehicle slightly oscillates to stabilize the maneuver, generating peaks in the orthogonal error.

In Fig. 6.5b, the U-turn scenario is illustrated. In this case, the blue interval represents the full area from the start of the first U-turn motion - after which the vehicle moves in the opposite direction compared to the training scenario - to the end of the second U-turn motion - after which the vehicle moves again in the same direction as performed during training. In this interval, two separate causes of odometry anomaly can be identified. On the one hand, the vehicle is always moving in the opposite direction compared to training. On the other hand, in four distinct subparts of this area, the vehicle performs abnormal rotation and acceleration actions: it executes the U-turn maneuver at the beginning and at the end of the interval, and it curves right twice, whereas in training it only curved left. These four subparts are highlighted in red in Fig. 6.5a. The trajectory of this scenario is shown in Fig. 4.16c.

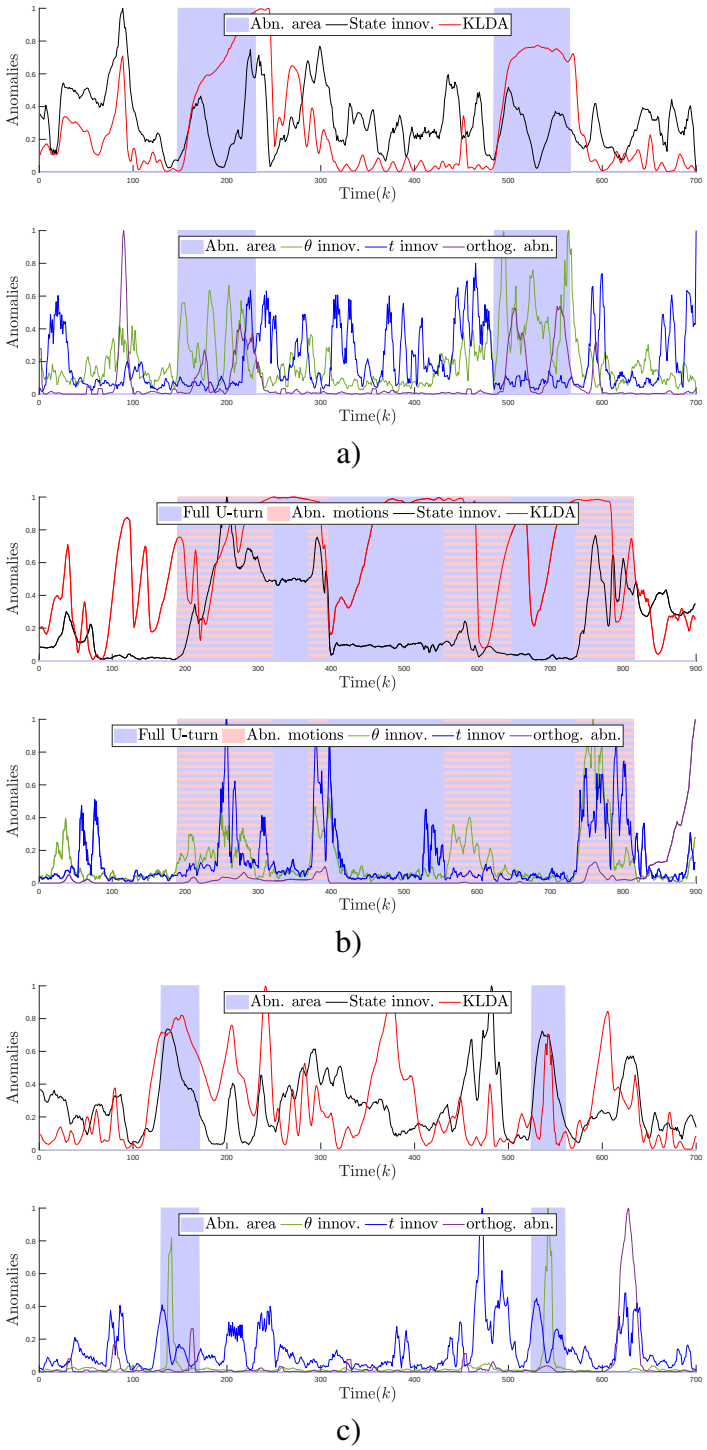


Fig. 6.5 Anomalies at the state and cluster level (above) and at the parameters and orthogonal direction level (below) for the PA (a), U-turn (b), and ES (c) scenarios. © 2021 IEEE.

The KLDA anomaly signal displays high values throughout the blue interval. High rotation errors, instead, are concentrated in the four abnormal red areas. In the rest of the blue interval, the vehicle is moving straight, as during training, and therefore no rotation error is present. The acceleration error has high values in three of the red areas as well.

Finally, Fig. 6.5c shows the ES scenario. The blue intervals correspond to the emergency stop maneuver. The applied methods in this scenario, however, perform worse than in the other two scenarios. This is also due to the vehicle starting and stopping, during the first and last time instants of the training dataset, in the same zone where it is performing the emergency stop during testing. Since the two motions are similar and are performed in the same zone, the algorithm is not able to detect the anomaly. This anomaly is more effectively detected from the camera data, as demonstrated in Chapter 5.

6.4.3 Driver behavior classification on the UAH-DriveSet dataset

As the driver behavior classification dataset does not have a training and a testing set, we first build multiple training sets. Five trajectories are available for each class. Consequently, five training datasets are obtained for each class, by excluding in each dataset one of the five trajectories. The excluded trajectory is then used for testing.

Therefore, we obtain the following training datasets, each composed of four trajectories: $X_{norm}^{train,1}, \ldots, X_{norm}^{train,5}, X_{drowsy}^{train,1}, \ldots, X_{drowsy}^{train,5}, X_{aggr}^{train,1}, \ldots, X_{aggr}^{train,5}$.

We also build the following testing datasets, each composed of one trajectory: $X_{norm}^{test,1}, \ldots, X_{norm}^{test,5}, X_{drowsy}^{test,1}, \ldots, X_{drowsy}^{test,5}, X_{aggr}^{test,1}, \ldots, X_{aggr}^{test,5}$.

For each training dataset, a G-MJPF is trained. We need a G-MJPF of each class to compute the cluster distances $\mathbf{d}^i$, with $i = 1 \ldots N$. Therefore, we associate the datasets extracted above to obtain five experiments: the first one combines $X_{norm}^{train,1}$, $X_{drowsy}^{train,1}$, $X_{aggr}^{train,1}$ (for training), $X_{norm}^{test,1}$, $X_{drowsy}^{test,1}$, and $X_{aggr}^{test,1}$ (for testing); the second one combines $X_{norm}^{train,2}$, $X_{drowsy}^{train,2}$, $X_{aggr}^{train,2}$ (for training), and $X_{norm}^{test,2}$, $X_{drowsy}^{test,2}$, $X_{aggr}^{test,2}$ (for testing), and so on.

Finally, during testing, each trajectory is filtered with the three G-MJPF associated with its experiment.

Classification is performed by employing five errors and anomaly distances on the state along its direction of motion, on the motion parameters, and on the state along the orthogonal direction. The obtained classification accuracy is 73.33%.

Fig. 6.6 shows the confusion matrix associated with the classification results. We observe that the aggressive trajectories are correctly classified three times out of five, whereas the normal and drowsy trajectories are each classified correctly four times out of five. The lower performance on the aggressive trajectories is in line with what was observed by Arroyo et al.

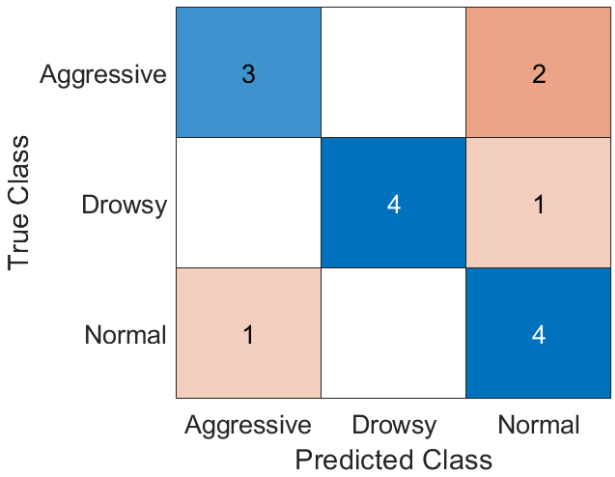


Fig. 6.6 Confusion matrix obtained classifying the considered subpart of the UAH-DriveSet dataset.

[190]: since the difference between the normal and aggressive driving styles highly depends on the driver, classification of this behavior is more difficult.

The comparison with other approaches is not easy. The UAH-DriveSet dataset provides a lot of sensory information: GPS, IMU, and camera raw data, as well as pre-processed camera information. The DriveSafe app [22, 189, 190], which was used to process the dataset, adopts computer vision and pattern recognition techniques to detect lane marks and to track the lane in which the driver is moving. Events such as lane drifting and lane weaving are detected. Therefore, the authors of the dataset, and other researchers [120, 194, 247], proposed methods based on both GPS, IMU, and pre-processed camera information. Conversely, in our study, only the data from GPS and IMU are exploited. An additional difficulty when performing a comparison is the part of the dataset employed to evaluate the method's accuracy. Some authors only use the secondary road part of the dataset, or the motorway part, or both. Some authors only employ the normal and aggressive trajectories, ignoring the drowsy ones. Furthermore, for data-driven methods such as ours, another element that can vary between papers is the part of the dataset used for training and for testing. As a final difference, some researchers classified the trajectories at the end, as in the original paper [190], whereas others classified them after a few minutes of driving. For this reason, we compare the results with the ones obtained by three papers that used only the GPS and/or IMU. Since, however, the training was performed on different data subsets compared to our case, we describe each subset in detail.

In [234], several methods are compared. In all cases, GPS and IMU data are employed; 70% of the UAH DriveSet dataset (from each trajectory) is used for training and the remaining

30% is employed for testing. The sixth dataset trajectory is not used. First, the authors train different classifiers on the entire motorway and secondary road training set and choose the one displaying the best results. Five types of classifiers are evaluated: discriminant analysis [46], decision tree [178], K-nearest-neighbours (KNN) [67], Support Vector Machine (SVM) [42], and ensemble learning [160]. The accuracy for each classifier can vary greatly, depending on its type. For example, The accuracy of SVM ranged from 24.9% (with a cubic kernel) to 67.4% (with a fine Gaussian kernel). A simple Decision Tree was shown to have a 52.7% accuracy; a complex one a 63.1% accuracy. KNN was tested by calculating the distance in various ways; the worst results were obtained with a quadratic distance (51.1%), and the best with a weighted distance (66.8%). The method displaying less variety was the discriminant analysis, which results in an accuracy of 51.0% in the linear case, and in an accuracy of 51.1% in the quadratic case. Finally, ensemble learning, ranged from 51.5% (subspace discriminant) to 81.3% (random forest with Bayesian parameter optimization). Therefore, the best result was obtained with ensemble learning with random forest, with an accuracy of 81.3%. The second best was ensemble learning with subspace KNN, with an accuracy of 69.0%. Then, the authors proposed two modifications. Firstly, they decided to train the classifiers on each driver behavior, and not on the general driver behavior, i.e., to train a classifier for each driver, in each of the three behaviors. During testing, 15 classes can be assigned to a trajectory, as there are 5 drivers and 3 driving behaviors. The behavior of the assigned class is the predicted behavior of the trajectory. The overall learned model is defined in this case the "personalized model". Secondly, the authors decided to split the learning of the classifiers distinguishing the secondary road and the motorway, instead of training everything together. They name the learned model in this case the "road-aware" model. The road-aware model has an accuracy of 86.2% on the motorway and of 80.9% on the secondary road. The accuracy of the personalized model increased to 93.6% on the motorway and to 89.4% on the secondary road. The overall accuracy of the personalized road-aware model is 91.6%.

Another paper [215] proposes a classification method on the GPS data based on an RNN model, using either a Gated Recurrent Unit (GRU) [12] or an LSTM. Both secondary and motorway data are used. Training is performed in two ways. The first method (here called $F1$) excludes the fifth trajectory of the dataset from training and uses it for testing. This is similar to what we performed. However, in our case, all trajectories were used, in turn, as testing trajectories. Furthermore, the authors split all the trajectories (except the fifth) into a part for training (80%) and a part for validation (20%). The second method ($F2$) uses 70% of the data for training, 15% for validation and 15% for testing. Data were either normalized or standardized. One-minute-long, two-minutes-long, and three-minutes-long blocks of data

were employed for classification, instead of classifying the entire trajectory at the end. We examine only the $F1$ three-minute case with normalization, as it corresponds more to our scenario. With GRU, the accuracy was 82.8%; with LSTM is was 90.3%.

Finally, in [14], a method based on a sequence-to-sequence LSTM Autoencoder and hierarchical clustering is presented. First, during training, an LSTM Autoencoder is learned on the training data. Agglomerative hierarchical clustering is executed on the learned representation. A classifier is learned on each cluster. Then, during testing, test data are clustered too. Two distance measures are proposed to find the best pairs of clusters between training and testing. In each testing data point, the corresponding classifier is employed. The method is evaluated on the entire UAH-DriveSet dataset, employing only the IMU data. The authors use 80% of the data for training and 20% for testing. Two classifiers are employed: a stacked LSTM [120] and an MLSTM-FCN [116]. In the first case, an accuracy of 80% was obtained; in the second of 87%, with both proposed distance measures.

Comments for comparing the accuracies of these three methods with our own are difficult to make due to the aforementioned training differences. However, this analysis allows us to have a general idea of the accuracy of other methods. Our method appears to be approximately in line with other methods, performing better than all five classifiers seen in [234] (except ensemble learning with random forest), but worse, to varying degrees, than other methods. However, in the future, more precise and extensive comparisons could be performed, by training on the data subparts considered in each of these works.

6.5 Conclusions

This chapter proposes a method to learn prediction models of the state of an object along the direction toward its motivation and along the orthogonal direction. A G-MJPF is developed to perform anomaly detection and is additionally used for driver behavior classification. The G-MJPF is obtained by joining two filters, an R-MJPF and a basic MJPF. Compared to the basic MJPF, on which it is based, the R-MJPF performs clustering on the GS (position and velocity, in the case of odometry) and on additional motion parameters related to rotation and acceleration. Rotational predictive models are employed instead of the original linear models. Filtering is performed on the GSs and motion parameters and the GEs are extracted and used to build a basic MJPF. This second component of the G-MJPF allows us to filter the movement of the agent along the normal to the direction of motion. In this way, we can examine how the agent oscillates along the motion towards its goal.

The proposed G-MJPF is employed for driver-behavior classification. For each driver behavior class, a G-MJPF is built. A graph is learned for each class, based on the transition

matrix of the learned vocabulary. We identify how much each cluster is specific to the class calculating a set of clusters' distances. During testing, the filters' anomalies are multiplied by the clusters' distances (based on the cluster assigned to each particle), and are used to estimate the behavior class of the trajectory.

The method is tested on two real-world datasets. First, it was shown that the method can explain what type of motion and what direction (toward the objective or in the orthogonal direction) is abnormal. Driver behavior classification was also evaluated and compared to other methods. Further testing could be performed in the future to extend these results.

Therefore, in this chapter, we presented a method adopting the proposed DBN framework on low-dimensional data. In Chapter 5, we examined an approach for high-dimensional data. In the next chapter, both low and high-dimensional sensors are employed.

Chapter 7

A Kalman Variational Autoencoder Model assisted by Odometric Clustering for Video Frame Prediction and anomaly detection

In Chapters 5 and 6, we have proposed an approach based on the self-awareness framework underlying this thesis adopting camera and odometry data, respectively. Instead, in this chapter, we combine the two data types.

We present a method for video-frame prediction and anomaly detection that leverages odometric data. We fuse DBNs and DL methods to learn an appropriate latent space. A basic MJPF is first built using the odometry data. Then, we learn a KVAE-based video model. The KVAE is modified to exploit the cluster vocabulary of the odometry model during its training phase. During the testing phase, anomaly detection is performed leveraging both the odometry and video data.

Section 7.1 introduces the problem of joining multi-sensory data of different dimensionalities. In Section 7.2, the proposed method is described. Results are discussed in Section 7.3. Finally, Section 7.4 draws some conclusions.

7.1 Introduction

The combination of different sensory information to predict upcoming situations is an innate capability of intelligent beings. Consequently, various studies in the Artificial Intelligence field are currently being conducted to transfer this ability to artificial systems.

Humans are intelligent beings who interpret and predict upcoming situations based on current and previous experiences. Additionally, humans can elaborate multi-sensory information, relating one sense to the others. According to Gibson [81], there are two different sensitivity levels, i.e., passive and active perception. Passive perception only considers the stimulation from the environment. In active perception, instead, the attentive actions and reactions of the observer to the environmental inputs generate a stimulus themselves.

Currently, studies are being conducted to transfer to artificial systems the human capability of prediction and correlation over multi-modal observations. Anomaly detection is an application of this prediction capability [149, 152, 166].

A field that can benefit from multi-modal fusion is that of autonomous systems, which are endowed with a multitude of sensors (e.g., cameras, GPS, IMU). For example, for an autonomous car, certain visual information is expected to be accompanied by a particular type of motion or a steering maneuver by a deceleration.

Different sensory information can be fused with trained models in a variety of ways. Models separately obtained from different sensors can be joined together when homogeneity is present. On the other hand, the use of low-dimensional sensors (e.g., positional/motion data) can aid the learning phase in sensors displaying higher dimensionality and intrinsic non-linearities (e.g., video data) [107]. This second option derives from sensory data like vision being a powerful and complicated representation comprised of millions of features that are intractable without further processing. Moreover, image content is typically a highly non-linear function of the system dynamics; therefore, modeling of image data is non-trivial.

The motivation behind this chapter is to develop a data-driven multi-sensorial method based on DL and HDBNs to reduce the dimensionality of a higher-dimensional sensor by leveraging the information of a low-dimensional one.

In the previous chapters, we have leveraged HDBNs for anomaly detection on video and odometry data, separately. In both cases, an extended version of a MJPF [17] was employed. As we have seen, in this type of filter, the state is subdivided into clusters displaying similar content and motion, and a prediction model is assigned to each cluster. In the basic MJPF, linear prediction models are adopted.

To apply SLDS on higher dimensional data, dimensionality reduction is first necessary. Generative neural networks such as VAEs [122] and GANs [85] reduce dimensionality while learning at the same time the underlying distribution of the data. In Chapter 5, the AMJPF and OF-MJPF were introduced. Camera (or camera frames plus OFs) were encoded through a VAE. The obtained latent space could contain strong jumps between the encodings of consequent frames (and OFs), and a neural network predictor needed to be used for modeling the evolution inside each cluster, instead of the original linear models of the basic MJPF.

In this chapter, instead, we propose to learn a smoother latent space. The movement along the space inside a cluster can be measured by a linear approximator, which leads to a higher interpretability of the latent space. Moreover, a higher homogeneity with the self-awareness models for low-dimensional data can be reached, compared to Chapter 5, as we do not use non-linear prediction models.

As we have examined in Section 2.7, a variety of works in the literature treat the problems of combining VAEs and DBNs and learning generative models on sequential data. VAEs that deal with sequential data are called DVAEs [133, 18, 40, 72, 139, 145, 110, 71]. However, to the best of our knowledge, no previous work used DVAEs to combine information from multi-modal data. Among the cited DVAEs, the KVAE [71], described in Section 2.7.1, learns two levels of latent state abstraction. A pseudo-observation model and a prediction model as the ones in a KF [114] are defined by a set of matrices combined at each time instant through a probability vector.

To summarize, this chapter proposes a method using visual and odometric data of a moving agent. A model is built to perform prediction over the odometric data, employing a basic MJPF. The video model is based on the KVAE in [71]. The formulation through the combination of linear models given by the KVAE allows us to reach a better homogeneity with the odometric model formulation compared to the NN models proposed in Chapter 5. However, instead of using the KVAE defined in the original paper [71], we leverage the learned odometric model for guidance. Whereas, in the traditional KVAE, an LSTM was used to deduce the probabilities to combine the pseudo-observation and prediction models, we propose using the cluster probabilities obtained from the odometric model. We call our model Cluster Guided KVAE (CG-KVAE).

Additionally, testing is performed with anomaly detection methods similar to the ones seen in the previous chapters.

Consequently, the main contributions of this chapter are:

- the use of a low-dimensionality sensor (odometry) to guide the learning phase of a high-dimensional one (video), therefore obtaining a common clustering displaying better properties;
- the testing of the proposed model on data from real vehicles, in particular considering the camera and GPS data.

7.2 Proposed Approach

7.2.1 A multi-modal learning framework

Two types of data from a moving vehicle are provided, i.e., video acquired from an onboard sensor (First Person Viewpoint - FPV) and the corresponding odometry data. One base hypothesis of the method is a good synchronization between the odometry and the video data (see section 7.3.1 for further notes regarding the synchronization).

The proposed method is divided into two parts: The first phase is a *training* phase, in which a training dataset is used to learn models for future-value prediction of both the odometry and video data. The second phase is a *testing* phase, in which the learned models are tested on another dataset; predictions that diverge from the actual future values indicate the presence of an anomaly.

During the training phase, odometry data are leveraged to perform learning in the video case. In the testing phase, the odometry data are consequently passed again to the video module for performing future-frame prediction. Fig. 7.1 displays the training phase of the method, whereas Fig. 7.2 refers to the testing procedure.

$\{x_t^o\}_{t=1...\tau}$ defines a set of odometry data observations, where τ corresponds to the total number of considered time instants. Conversely, $\{x_t\}_{t=1...\tau}$ defines a set of video data observations.

7.2.2 Learning of the odometry model

The basic idea of our method is to leverage the model built on one sensor S_A to guide the learning of the model built on a sensor S_B. In the case considered, S_A corresponds to an odometric sensor (e.g., GPS), and S_B to the camera one. Consequently, the first step that needs to be performed is the learning of the odometry model. A DBN framework is built such that the odometry and the video model are mostly consistent. In particular, probabilistic linear state-space models are considered.

The learning on the odometry modality is based on a basic MJPF [17]. From each odometry sensor observation x_t^o, a GS [77] is obtained. The GSs contain the state and its first-order time derivative:

$$\tilde{z}_t^o = [z_t^o, \dot{z}_t^o] \tag{7.1}$$

As seen in Section 3.7.1, the states are correlated to the observations supposing a linear relationship of the following type:

$$\dot{z}_t^o = \frac{x_t^o - H z_{t-1}^o}{\Delta t} \tag{7.2}$$

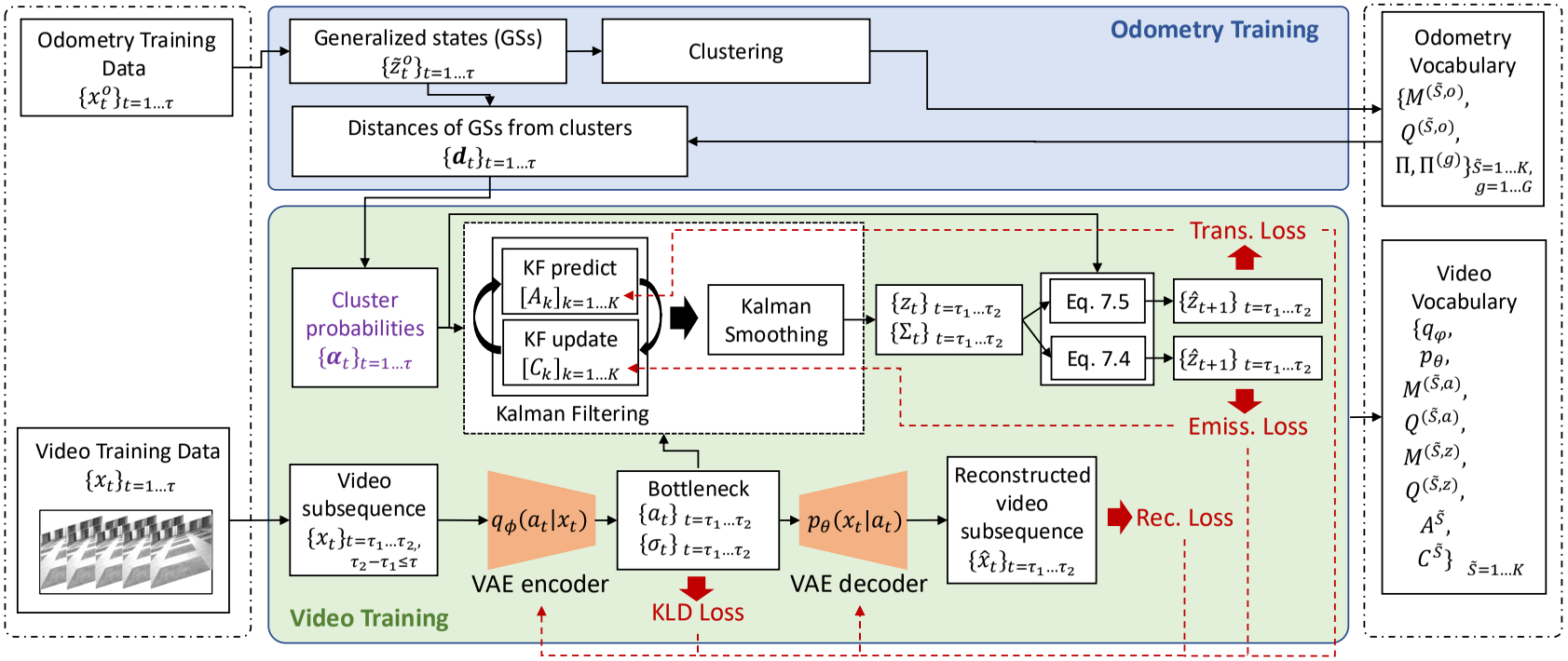


Fig. 7.1 Proposed training Structure. The upper block calculates the GSs of the odometry data, extracts the odometry clusters vocabulary, and finds the distance of each GS from each cluster. The odometry vocabulary is leveraged in the video block to train the CG-KVAE with the inverse distance (cluster probabilities) to focus the attention of the latent states a_t and z_t. © 2023 IEEE.

Fig. 7.2 Proposed testing Structure. The upper block extracts the GSs and applies anomaly detection on them through the MJPF. The lower block uses the trained CG-KVAE to perform filtering on the z_t state and extract multi-level anomalies. © 2023 IEEE.

where H is an identity matrix.

Clustering is performed on the GSs using the GNG algorithm [78].

For each cluster $\tilde{S}$, with $\tilde{S} = 1 \dots K$, where K is the total number of clusters, the following information is extracted, as in the previous chapters: *i)* the cluster centroid $M^{(\tilde{S},o)}$; *ii)* the cluster covariance $Q^{(\tilde{S},o)}$; *iii)* a transition matrix Π defining the probability of moving from one cluster to the other ones; *iv)* optionally, a set of TTMs $\Pi^{(g)}$.

Having extracted all the above-defined cluster features, the learning of the odometry model is complete.

Note, however, that also other DBN models could be used, extracting different features instead of the defined GS $\tilde{z}_t^o$ and adopting different prediction models. An example could be the model defined in Chapter 6, where the evolution of an agent's state is decomposed in its different motion-related parameters and a rotational prediction model is employed.

7.2.3 Extraction of clustering distance values in odometry

After the odometry model has been learned, the features to guide the learning of the video model are extracted. We propose to do this through a probabilistic distance (e.g., Bhattacharyya distance D_B) between each GS $\tilde{z}_t^o$ and each cluster:

$$d_t^{\tilde{S}} = D_B((\tilde{z}_t^o, \Sigma_t^o), (M^{(\tilde{S},o)}, Q^{(\tilde{S},o)})). \tag{7.3}$$

where Σ_t^o represents the covariance of $\tilde{z}_t^o$. We chose Bhattacharyya distance as a probabilistic distance coherent with one of the anomalies that will be introduced in Section 7.2.7 to perform anomaly detection. Other distances could be employed too.

In this way, a set of $\tau * K$ distances is obtained. $\boldsymbol{d}_t$ defines the set of K distances at each time step t. These distances are given as input to the video training block to guide the learning phase and to set the clustering division.

7.2.4 Learning of the video model

As a second step of the proposed method, training is performed on the video dataset employing the KVAE algorithm presented in [71] and summarized in Section 2.7.1.

We propose using the clustering obtained from the odometry data and the found distances $\{\boldsymbol{d}_t\}_{t=1\dots\tau}$ to set the same clustering and to guide the KVAE training phase. Note that the original KVAE paper [71] did not mention the concept of clustering but rather defined the presence of a set of K combinable dynamics. These K dynamics can be seen as a loose clustering assignment. As the original KVAE requires the user to set the number of clusters a

priori, the proposed method allows us to exploit the information from another modality (i.e., the odometry one) to fix this number. By using the distances $\{\boldsymbol{d}_t\}_{t=1...\tau}$, the two clusterings are fixed to overlap. During the online phase, this will allow a joint anomaly detection of the two modalities, which are now represented through similar information.

Also, note that this odometry-assisted assignment has differences compared to the case of dynamics learned from video. Firstly, zones with similar visual aspects and motion, but different positions, are assigned to different clusters. This behavior could either be desirable or not, depending on the case. In particular, if events are acceptable in some regions of a map but not in others, this is an advisable feature. Furthermore, in certain situations, video data could be different even if odometry data does not change, e.g., the car is not moving, and various events might happen in front of the agent, such as pedestrians and vehicles moving. However, these visual changes are unrelated to motion changes and are less relevant for interaction analysis.

Consequently, the same pseudo-observation and dynamics models defined in the original KVAE are kept, with the difference that the (optional) control matrices B_i are not included. These models were defined in Section 2.7.1, but we report them also here for better readability, without commenting on them further. However, it is worth reminding the reader that the KVAE learns two latent state: a_t and z_t. The z_t state must not be confused with the z_t^{IM} extracted in Chapter 5. In this chapter and in the next one, we define as Σ_t the covariance associated with the state z_t, and as Σ_t^a the covariance associated with the state a_t.

The KVAE pseudo-observation and dynamics models are defined, respectively, as follows:

$$a_t = \sum_{i=1}^{K} \alpha_t^{(i)} * C_i * z_t + v_t \tag{7.4}$$

$$z_{t+1} = \sum_{i=1}^{K} \alpha_t^{(i)} * A_i * z_t + \sum_{i=1}^{K} \alpha_t^{(i)} B_i * u_t + \omega_t \tag{7.5}$$

In both Eq. 7.4 and in Eq. 7.5, a probability vector $\boldsymbol{\alpha}_t$ is necessary to combine the different models and define which are more relevant. In the original KVAE, this was performed with an LSTM followed by a softmax. Instead, in our method, we propose to use the odometry model to guide the assignment of the matrices. The vector of distances $\boldsymbol{d}_t$ is adopted to calculate the probability vector $\boldsymbol{\alpha}_t$ ($\alpha_t^{(\tilde{S})}$ corresponds to one element in $\boldsymbol{\alpha}_t$):

$$\alpha_t^{(\tilde{S})} = softmax\left(\frac{1}{d_t^{(\tilde{S})} + \varepsilon} * \frac{1}{n}\right), \tag{7.6}$$

where n is a temperature, and ε is a small positive value added to avoid the denominator being zero. A low value of n leads to a multinomial probability distribution peaked around the nearest cluster, whereas a high value of n leads to a flatter multinomial probability distribution. This expression corresponds, therefore, to a softmax with temperature. Across epochs of the video training, the value n can be annealed.

Optimization is performed on the parameters ϕ and θ of the VAE, and on the matrices $[A_k, C_k]_{k=1...K}$.

Note how the odometry sensor data are used as a focusing attention mechanism. This clustered odometry data can be considered as a switch mechanism guiding the network to weigh more its sensitivity towards the cluster to which the current input will be assigned. It can be observed how, in this way, selected features are driven by the system's action. Following Gibson [81], the system not only stops at the passive detection (given by the images only) but also considers the stimulus from odometry, making it active.

Alg. 6 summarizes the video training algorithm - which follows the original KVAE [71] - and clarifies where the cluster probabilities $\boldsymbol{\alpha}_t$ should be inserted.

We summarize and further explain the video training choices in this paragraph. As discussed, the vector of probabilities $\boldsymbol{\alpha}_t$ originally learned through an LSTM [71] is here obtained using the distance of the odometry state $\tilde{z}_t^o$ from the found clusters. The clustering is built using the concept of GSs, which include both the positional and the motion information of the agent [77]. Consequently, this first unsupervised learning step allows not only to define the number of clusters but also their type, e.g., clusters where the agent moves linearly following a certain motion type. Then, the probabilities vector $\boldsymbol{\alpha}_t$ guides when each video cluster must be activated. The matrices A and C are learned mixing models with these odometry-obtained probabilities (with the mixing becoming more peaked around the dominant cluster with the decrease of the temperature n, which makes the assignment tighter). The motion-related z_t state, forced to be smooth, evolves locally according to the A matrices model, and learns to predict not the entire information, but what is common to the motion/evolution inside the cluster. Note how this concept resembles the idea proposed by Yann LeCun (Section 3.8) to concentrate the prediction effort on a subpart of the information in the scene, and not on all observation details. Conversely to what was proposed by LeCun, however, we also decode the prediction back to the observation level, even if it can be blurry.

7.2.5 Extraction of the video cluster features

Once the training is finished, as done for the odometry case, for each cluster $\tilde{S}$, the proposed method extracts the cluster centroids ($M^{(\tilde{S},z)}$, $M^{(\tilde{S},a)}$) over the values of the latent states (z_t, a_t), and over the cluster covariances ($Q^{(\tilde{S},z)}$, $Q^{(\tilde{S},a)}$).

Algorithm 6 Training over the video data.

Input: Video training data $\{x_t\}_{t=1\dots\tau}$, divided in sequences, as $\{x_t\}_{t=\tau_1\dots\tau_2}$. Distances from clusters $\{\boldsymbol{d}_t\}_{t=\tau_1\dots\tau_2}$.

Initialize VAE, matrices A and C.

for i = *1* ***to*** $E_p \leftarrow$ *number of epochs* **do**

Calculate cluster probabilities $\{\boldsymbol{\alpha}_t\}_{t=\tau_1\dots\tau_2}$, from odometry clusters distances $\{\boldsymbol{d}_t\}_{t=\tau_1\dots\tau_2}$, using Eq. 7.6.

Get latent state of images using VAE's encoder, for $t = \tau_1 \dots \tau_2$:

$a_t, {\sigma^2}_t = q_\theta(x_t) \leftarrow$ Image latent state

$\Sigma_t^a \sim I_L \sigma_t^2 \leftarrow$ Latent state covariance

Decode the latent state back to image level, for $t = \tau_1 \dots \tau_2$:

$\hat{x}_t = p_\phi(a_t) \leftarrow$ Reconstructed images

KALMAN FILTERING:

for t = τ_1 ***to*** $\tau_2 \leftarrow$ *time evolution* **do**

Perform Kalman Prediction, using Eq. 7.5 for the prediction model, combining matrices A with the calculated $\{\boldsymbol{\alpha}_t\}_{t=\tau_1\dots\tau_2}$:

$z_{t+1|t}, \Sigma_{t+1|t} = KF_{pred}(z_{t|t}, \Sigma_{t|t}, [A_k]_{k=1\dots K}, \boldsymbol{\alpha}_t)$

Perform Kalman Update, using Eq. 7.4 for the observation model, combining matrices C with the calculated $\{\boldsymbol{\alpha}_t\}_{t=\tau_1\dots\tau_2}$. The pseudo-observation is given by a_t and R^v (i.e., covariance matrix of the measurement noise).

$z_{t|t}, \Sigma_{t|t} = KF_{update}(z_{t|t-1}, \Sigma_{t|t-1}, [C_k]_{k=1\dots K}, \boldsymbol{\alpha}_t, a_t, R^v)$

end

KALMAN SMOOTHING:

Perform the Kalman Smoothing procedure over state z_t, obtaining smoothed values of z_t, Σ_t, with $t = \tau_1 \dots \tau_2$.

CALCULATION OF LOSSES:

A) VAE LOSSES:

Reconstruction loss as MSE between x_t and $\hat{x}_t$, and KLD loss over a_t, Σ_t, for $t = \tau_1 \dots \tau_2$.

B) KF LOSSES:

for t = τ_1 ***to*** $\tau_2 \leftarrow$ *time evolution* **do**

Perform prediction using Eq. 7.5, from z_t, obtaining $\hat{z}_{t+1}$. Transition loss is calculated comparing (z_{t+1}, Σ_{t+1}) from smoothing and $\hat{z}_{t+1}$.

Go back to latent state a_t using Eq. 7.4, from z_t, obtaining $\hat{a}_t$. Emission loss is calculated comparing (a_t, Σ_t^a) from the VAE and $\hat{a}_t$.

The other KVAE losses are calculated (i.e., entropy loss and initial covariance loss).

end

Perform optimization over VAE parameters ϕ, θ and /or matrices A, C, using the calculated losses or a subset of them.

end

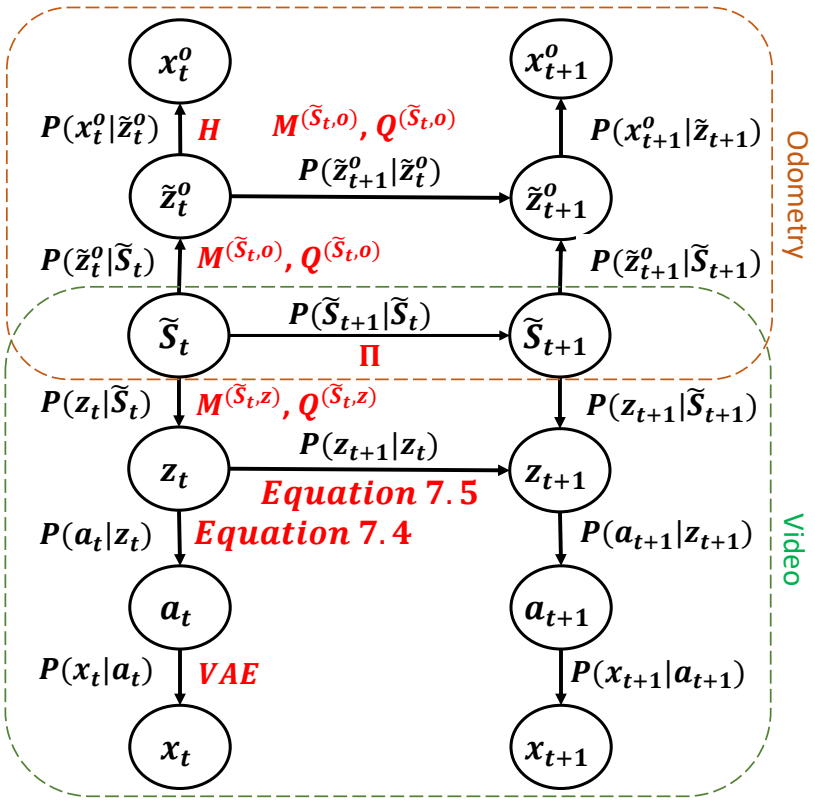


Fig. 7.3 Proposed DBN structure. © 2023 IEEE.

7.2.6 Learned Dynamic Bayesian Network

Whereas the DBN on the odometry remains the same as displayed in Fig. 3.3, the DBN on the video and how the two connect is shown in Fig. 7.3, with the learned elements displayed in red corresponding to each link.

$P(x_t|a_t)$ and $P(a_t|x_t)$ are defined by the VAE's decoder and encoder, i.e., the KVAE acts as an observation model.

The standard pseudo-observation (Eq. 7.4) and state prediction (Eq. 7.5) models of the VAE correspond to $P(a_t|z_t)$ and $P(z_{t+1}|z_t)$, respectively.

So far, the video subpart of the DBN corresponds to the original KVAE one displayed in 2.4. Additionally, it can be observed how $P(\tilde{S}_{t+1}|\tilde{S}_t)$, $P(z_t|\tilde{S}_t)$ and $P(\tilde{z}_t^o|\tilde{S}_t)$ are defined by the transition matrix Π (and, potentially, $\Pi^{(g)}$), and by mean $M^{(\tilde{S},z)}$ and covariance $Q^{(\tilde{S},z)}$, and by mean $M^{(\tilde{S},o)}$ and covariance $Q^{(\tilde{S},o)}$, respectively.

7.2.7 Anomaly detection

The second phase of the proposed method consists in the detection of anomalies on a testing set compared to the models learned in the training phase.

At each time-instant t, the MJPF can be used to extract odometry anomalies.

An MJPF is an SLDS comprised of K KFs at the state level and of a PF at the cluster level. The MJPF is employed as in [17] to extract anomalies by comparing predictions and updates at the state and cluster levels of the hierarchy.

While performing filtering and anomaly detection on the odometry data, the distance vector $\boldsymbol{d}_t$ is calculated (Eq. 7.3) and used to derive $\boldsymbol{\alpha}_t$ (Eq. 7.6), keeping the same n value used at the end of the training phase. The obtained $\boldsymbol{\alpha}_t$ is passed to the video block.

In parallel, each video frame x_t is given as input to the VAE's encoder, and a Kalman filtering step is performed on it, where the observation model is defined as in Eq. 7.4 and dynamics model as in Eq. 7.5, using the $\boldsymbol{\alpha}_t$ just calculated by the odometry block. Predicted latent state values $a_{t+1|t}$ can be decoded back to the image level.

The testing process for video prediction and anomaly detection is summarized in Alg. 7.

The learning of the DBN allows the detection of anomalies at different hierarchical levels. Therefore, similar anomalies to the ones introduced in Chapter 5 are applied and adapted to the proposed approach.

At the image-level x_t, the MSE between predicted images $x_{t+1|t}$ and real images x_{t+1} can be calculated. Even direct error reconstruction can be used, calculating the MSE between the directly reconstructed image $\hat{x}_t$ and the real image x_t.

At the latent states level, the error values at the a_t and z_t levels are:

$$err_{a_t} = \frac{1}{D_a} \sum_{n=1}^{D_a} (a_{n,t+1|t} - a_{n,t+1}), \tag{7.7}$$

$$err_{z_t} = \frac{1}{D_z} \sum_{m=1}^{D_z} (z_{m,t+1|t} - z_{m,t+1|t+1}), \tag{7.8}$$

where D_a and D_z are the dimensions of the two latent states, respectively.

Finally, the KLDA used in Chapter 5 can be adapted to the KVAE approach. First, the Bhattacharyya distance $D_B(\lambda(z_{t|t}), P(z_t|\tilde{S}_t) = \tilde{S}))$ is calculated between the current state z_t and each cluster, where:

$$\lambda(z_{t|t}) \sim \mathcal{N}(z_{t|t}, \Sigma_{t|t}) \tag{7.9}$$

$$P(z_t|\tilde{S}_t) \sim \mathcal{N}(M^{(\tilde{S},v)}, Q^{(\tilde{S},v)}) \tag{7.10}$$

Algorithm 7 Testing over the video data.

Input: Video testing data $\{x_t\}_{t=1...\tau}$. Distances from the clusters $\{\boldsymbol{d}_t\}_{t=1...\tau}$.
for *t = 1 to* $\tau \leftarrow$ *time evolution* **do**
Calculate cluster probabilities $\boldsymbol{\alpha}_t$, from odometry clusters distances $\boldsymbol{d}_t$, using Eq. 7.6.
Get the latent state of the images using the VAE's encoder:
$a_t, {\sigma^2}_t = q_\theta(x_t) \leftarrow$ Image latent state
$\Sigma_t^a \sim I_L \sigma_t^2 \leftarrow$ Latent state covariance
Decode the latent state back to image level:
$\hat{x}_t = p_\phi(a_t) \leftarrow$ Reconstructed images
KALMAN FILTERING:
Perform Kalman Prediction, using Eq. 7.5 for the prediction model, combining matrices A with the calculated $\boldsymbol{\alpha}_t$:
$z_{t+1|t}, \Sigma_{t+1|t} = KF_{pred}(z_{t|t}, \Sigma_{t|t}, [A_k]_{k=1...K}, \boldsymbol{\alpha}_t)$
Perform Kalman Update, using Eq. 7.4 for the observation model, combining matrices C with the calculated $\boldsymbol{\alpha}_t$. The pseudo-observation is given by a_t and R^v.
$z_{t|t}, \Sigma_{t|t} = KF_{update}(z_{t|t-1}, \Sigma_{t|t-1}, [C_k]_{k=1...K}, \boldsymbol{\alpha}_t, a_t, R^v)$
PROPAGATE PRED. DOWN THE HIERARCHY:
Go back to latent state a_t using Eq. 7.4 from $z_{t+1|t}$, $\Sigma_{t+1|t}$, obtaining $a_{t+1|t}$, $\Sigma^a_{t+1|t}$.
Decode the predicted latent state back to image level, as:
$\hat{x}_{t+1|t} = p_\phi(a_{t+1|t})$.
CALCULATE ANOMALIES:
Image level abns.:
1) direct reconstruction anomaly as:
$err_{\hat{x}_t} = MSE(x_t, \hat{x}_t)$;
2) prediction anomaly as:
$err_{\hat{x}_{t+1|t}} = MSE(x_{t+1}, \hat{x}_{t+1|t})$.
Latent states anomalies: calculate err_{a_t} and err_{z_t} using Eqs. 7.7 and 7.8.
Cluster level anomaly: find $KLDA_t$ using Eq. 7.11.
end
Output: $err_{\hat{x}_t}$, $err_{\hat{x}_{t+1|t}}$, err_{a_t}, err_{z_t}, $KLDA_t$.

being $\Sigma_{t|t}$ the updated covariance of z_t. We call $\lambda(\tilde{S}_t)$ the inverse of the set of D_B distances over all the clusters, similarly calculated as $\boldsymbol{\alpha}_t$ in Eq. 7.6; it describes the probability of being in each video cluster, given the updated latent state value, and corresponds to a diagnostic message of the DBN (i.e., a message from the observations). The corresponding predictive message $\pi(\tilde{S}_t)$ is the sum of the rows of Π weighted by $\boldsymbol{\alpha}_t$. The symmetric Kullback-Leibler Divergence between these two probabilities is calculated, i.e.:

$$KLDA = D_{\mathrm{KL}}(\pi(\tilde{\mathrm{S}}_\mathrm{t}) || \lambda(\tilde{\mathrm{S}}_\mathrm{t})) + D_{\mathrm{KL}}(\lambda(\tilde{\mathrm{S}}_\mathrm{t}) || \pi(\tilde{\mathrm{S}}_\mathrm{t})) \tag{7.11}$$

As in the previous chapters, the KLDA measurement allows us to define an anomaly at the highest level of the proposed hierarchy. This measurement identifies anomalies in the flow of events instead of identifying pixel-wise abnormalities.

7.3 Experimental results

7.3.1 Employed Evaluation Datasets

We evaluate our method on the iCab (see Section 4.2.1) and UAH-DriveSet datasets (see Section 4.2.2).

The iCab dataset [157] uses a Perimeter Monitoring (PM) scenario for training and Pedestrian Avoidance (PA), U-turn, and Emergency Stop (ES) scenarios for testing. Two sets of experiments taken at different times of day and season are present for the PM case and for the ES case. A set of PM is taken at the same season (but at a different time of day) as a set of ES. In this case, a shadow area is present on one side of the training courtyard for the training, whereas it is absent in the testing case. Experiments in this chapter are performed of this PM (PM_1) against his associated ES (ES_1) and of the combined PMs (plus data augmentation) on the remaining test cases.

The UAH-DriveSet dataset [190] is a dataset by the University of Alcalá for driver behavior analysis composed of various car sensory data from six drivers performing two routes (motorway and secondary road) with three types of behaviors (normal, drowsy, and aggressive). In this chapter, testing is performed on the secondary road part of the dataset. For this case, two groups of trajectories were provided for the normal case, a group in one direction of the route, and a group in the opposite direction. Instead, the drowsy trajectories were all in one direction, and the aggressive trajectories were in the opposite direction. We chose to train on the normal trajectories moving in the same direction as the drowsy ones, and we tested against the drowsy case. The dataset provided GPS, accelerometer, and video data. Due to GPS having a sampling rate of 1 Hz only, a KF is employed to combine GPS and accelerometer data (10 Hz), to obtain a 10 Hz estimation of trajectory data.

As observed in Section 7.2, one hypothesis of the method is a good synchronization between the odometry and the video samples. In both used datasets, the timestamps of the odometry and of the video are known. For the method to work well, the difference in timestamps between a video sample and its aligned odometry sample should be as small as possible (and it should be the smallest, the fastest the vehicle is moving).

Table I Roughness, car speed correlation, and Entropy of Π for the state and features obtained with each method. © 2023 IEEE.

	Roughness (r)	Vel. correlation (ρ)	Π Entropy (e)
VAE	0.1288	0.172	0.319
KVAE	0.0819	0.200	0.484
CG-KVAE	0.0861	0.334	0.188

7.3.2 Results on the iCab dataset

Training of the model on the PM data

First, the model is trained to learn the normal pattern. Two separate training procedures are performed, one using PM_1, and the other one combining it with PM_2 and performing data augmentation on it. We denominate the model learned on PM_1 as MD_1 and the model learned on PM_1 and PM_2 as MD_2.

From the odometry data training, the GNG outputs a total of 23 clusters. The method proposed in [17] is used, i.e., using position and first-time derivatives as features on which to cluster, as explained in Section 7.2.2. The trained KVAE has dimensions of a_t equal to 24, and of z_t equal to 4. Sequences of lengths $L = 20$ are used. The images were reduced to a dimensionality $64x64$. The VAE is composed of filters of dimension 32, 64, and 128, with a kernel of size 3. The value of n is set to 1 and increased of 0.2 at each epoch in which the matrices A and C are trained.

The model was trained in Python, using the PyTorch library.

Table I reports different discriminators on three cases of latent states: *i)* the one from a VAE performing direct reconstruction; *ii)* a_t from the original KVAE with LSTM training; *iii)* a_t from CG-KVAE. Roughness describes how smooth the state is and is calculated as $r = \frac{1}{4}(\dot{a}_{t,std} - \dot{a}_{t-1,std})^2$, where $\dot{a}_{t,std}$ is the standardized value of $a_t - a_{t-1}$. Another considered indicator is the mean absolute value of Pearson's Linear Correlation coefficients ρ between a_t and $\dot{x}_t^o$. Finally, the last column of the Table displays the entropy e of the transition matrix Π. In general $r, e, \rho \in [0,1]$ and we desire r and e to be low, and ρ to be high. The simple VAE gives the worst results over r and ρ. CG-KVAE displays the best entropy value, as it keeps the transition matrix of the odometry data.

To better understand the values of transition matrix entropy, see Fig. 7.4. In Fig. 7.4a the clusters obtained with the original KVAE (fixing the number to 23) are displayed. Notice how a cluster covers zones on opposite sides of the courtyard, and how the uncertainty in the passage between clusters is high (high entropy). Fig. 7.4b shows the CG-KVAE clusters, having lower entropy. Fig 7.4c shows, for each cluster in CG-KVAE, the other one that has

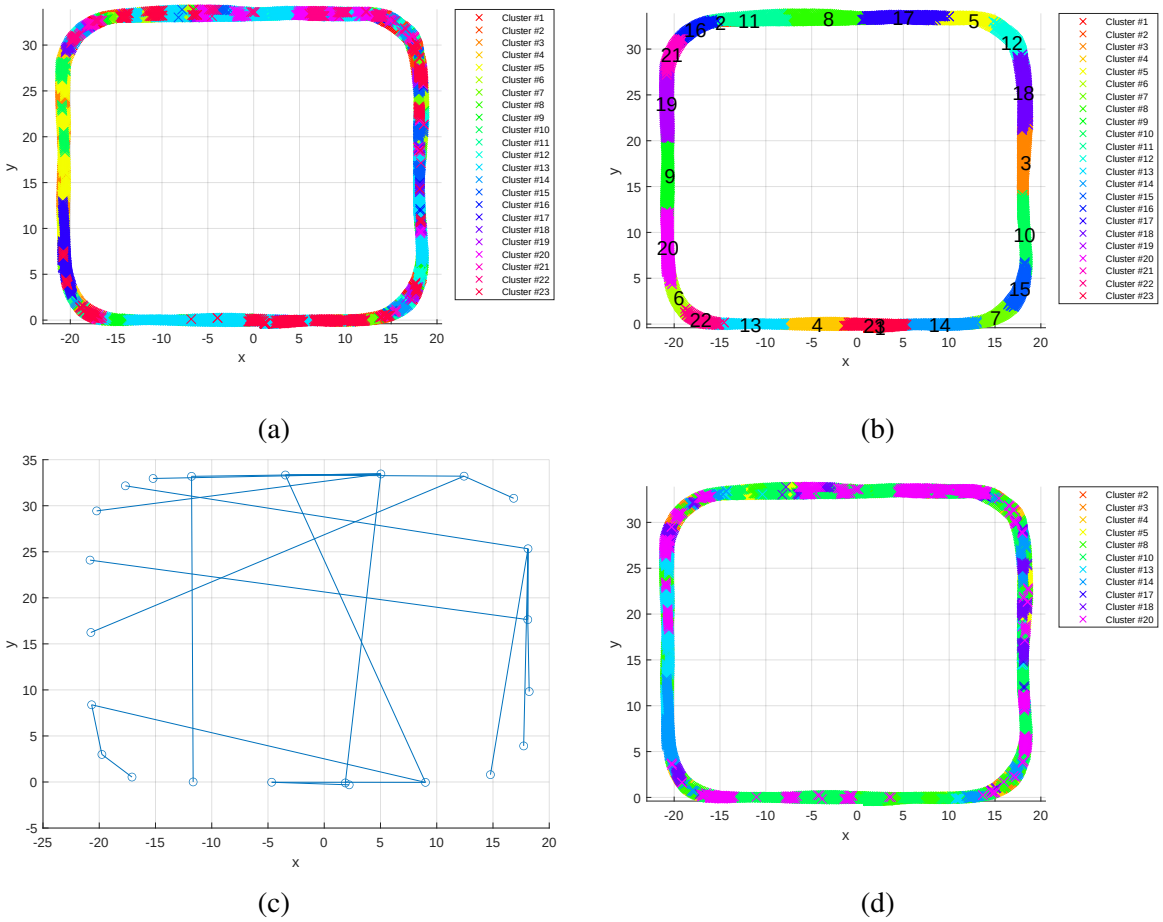


Fig. 7.4 (a): clusters of the KVAE, (b): clusters of the CG-KVAE, (c): for each cluster, the 2 clusters with closest A matrices are connected, (d): closest clusters of CG-KVAE in KVAE. © 2023 IEEE.

most similar A matrices. Notice how often zones on corresponding sides of the courtyard are connected, as they show very similar dynamic behavior. Finally, Fig. 7.4d displays the clusters of KVAE again, colored as the clusters of CG-KVAE having the most similar A matrices. The features have been reordered so that the ones displaying higher correlation are at the same index. However, as the learned embedding space is different, the correspondence is limited and the figure is not especially meaningful in zones with high changes such as curving zones. However, some A connected to straight areas and displaying higher values along their diagonal can be observed to correspond to zones in the same area in the original KVAE. For example, notice cluster 8 of KVAE (green). Observing Fig. 7.4c again, it can be seen how cluster 8 is very similar to cluster 14, in a symmetric courtyard area. Consequently, one can deduce that the CG-KVAE learns similar A matrices for zones in symmetrical visual areas but associates the better transition matrix of odometry.

Testing of the model on the ES data

First, we show the results obtained when testing the model on the ES data, using MD_1.

Fig. 7.5 uses colors to represent the events happening along the time axis. Yellow zones are curves that should not generate a high anomaly, as they are also found in the training data. Violet zones represent the part of the courtyard that was in the shadow in training but is in the light during testing; some considerations regarding these areas are given further on. Red and green areas are clear pedestrian-related abnormal areas, i.e., the pedestrian appears inducing the car to stop (red), and the car restarts after the pedestrian passes (green). Finally, in cyan zones, the car abnormally steers to the left.

Fig. 7.6 to Fig. 7.9 present all results related to this data case.

Fig. 7.6 shows the state-level anomaly obtained on the odometry data using the MJPF proposed in [17], with the trained odometry model.

Fig. 7.7, instead, displays the state-level anomaly that would be obtained if an MJPF were built directly on the VAE's encodings. It can be observed how curving zones all result in very high anomalies, whereas actual abnormal areas are mostly not detected as such. This behavior is due to the high non-linearity present in curving areas compared to the rest of the dataset, resulting in the model not learning to predict in those parts with sufficient accuracy. To solve this problem, a method such as a KVAE or the AMJPF or the OF-AMJPF in Chapter 5 should be used instead.

Fig. 7.8 shows the state-level anomalies (above) and the KLDA (below) when using the standard KVAE with LSTM, whereas Fig. 7.9 displays the same anomalies when a CG-KVAE is employed. It can be observed that, in both cases, prediction in curving zones is performing much better, not giving much higher anomalies than the rest of the sequence, differently to Fig. 7.7. The CG-KVAE displays fewer noisy anomalies than the KVAE; a peak is present in all pedestrian-related areas (red and green) and in the steering zone (cyan). Additionally, the violet areas display a strong abnormality, since odometry is used to select the model to guide prediction. During training, a shadow area was present in this zone; it is absent during testing. An anomaly is thus detected. On the other hand, the LSTM does not display strong anomalies in these areas, since one of the other areas in the light is used for prediction (as the courtyard has high symmetry). Whereas learning a shadow model and fixing it to a zone is of no interest some other types of more relevant behavior could be learned on a particular side of the courtyard. Consequently, a KVAE with LSTM could be used when similar rules can apply to similarly-looking zones; on the contrary, the CG-KVAE allows us to bind a behavior to its odometry area.

Fig. 7.10a displays the ROC (Receiver Operating Characteristic) curves for the anomalies using the original KVAE and the CG-KVAE, showing the better performance of the latter.

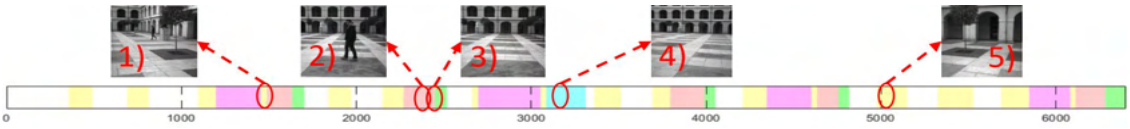


Fig. 7.5 Color-coded events in the ES dataset, and image examples. © 2023 IEEE.

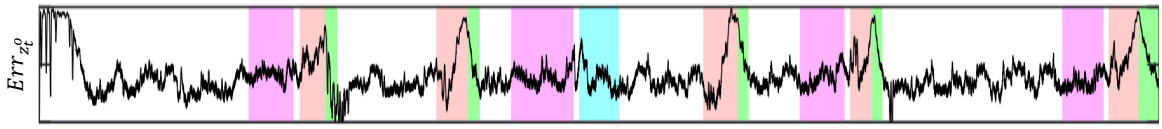


Fig. 7.6 State-level anomaly across time instants (on the x-axis) obtained with the MJPF [17] used on odometry data. © 2023 IEEE.

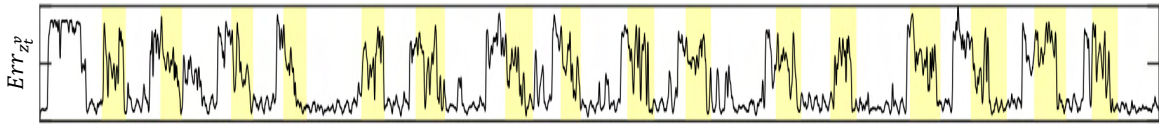


Fig. 7.7 State-level anomaly on ES across time instants (on x-axis) obtained with the direct MJPF [17] used directly on the VAE encodings. © 2023 IEEE.

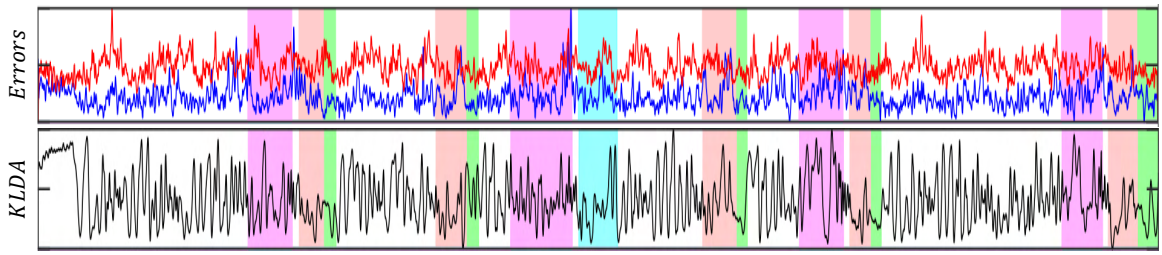


Fig. 7.8 Anomalies at the different levels on ES using the KVAE with LSTM. In blue: on a_t; in red: on z_t; in black: on $\tilde{S}_t$. © 2023 IEEE.

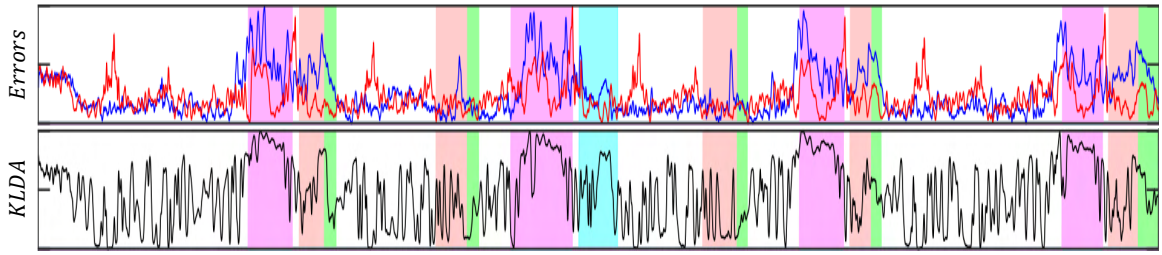
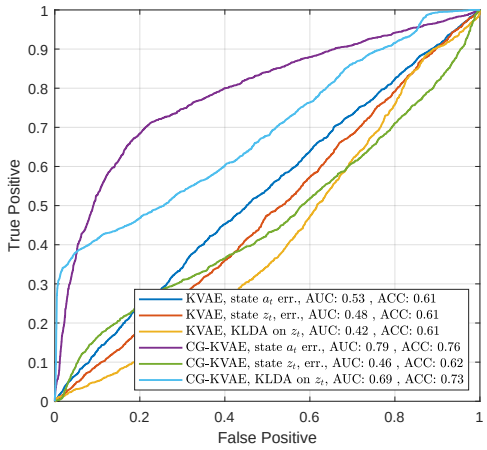
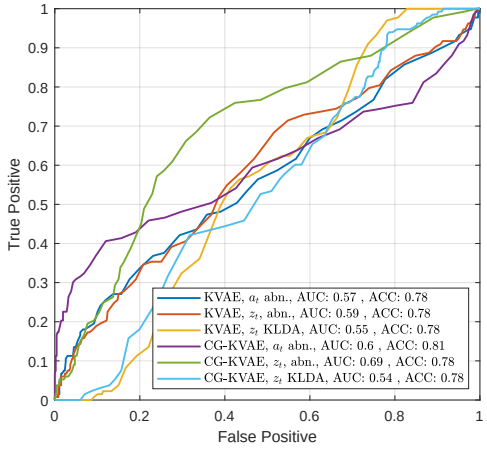


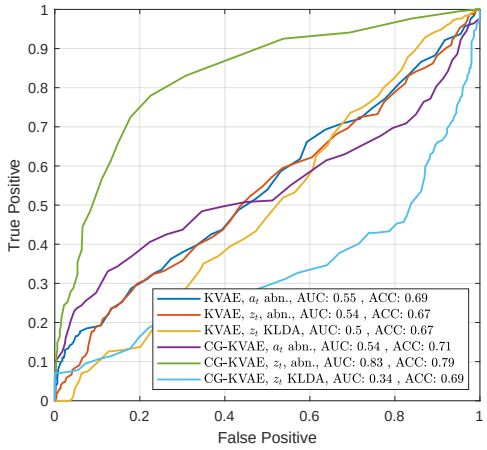
Fig. 7.9 Anomalies at the different levels on ES using the KVAE with the Odometry Clustering assignment. © 2023 IEEE.



(a)



(b)



(c)

Fig. 7.10 ROC curves for the following cases: (a) ES, (b) PA, (c) U-turn. For (b) and (c), the best cluster level as seen in Fig. 7.13 was chosen. No B_i matrices were used for either the CG-KVAE or the traditional KVAE. © 2023 IEEE.

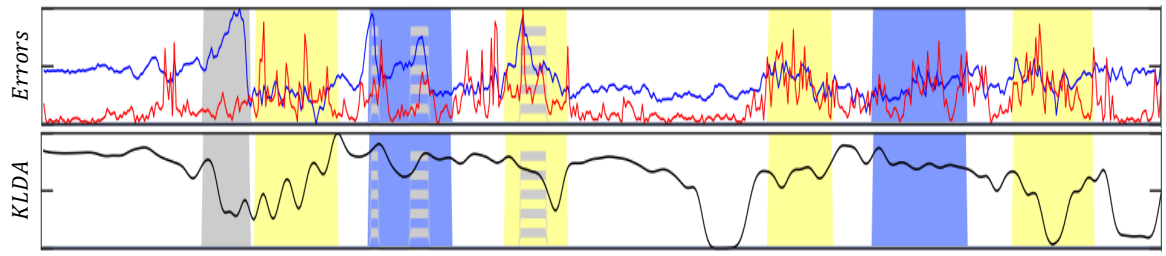


Fig. 7.11 Anomalies on PA using CG-KVAE. Blue interval: main anomaly (pedestrian+avoidance); grey: shadows; yellow: curve. The blue plot is the anomaly on a_t and the red plot is the anomaly on z_t. © 2023 IEEE.

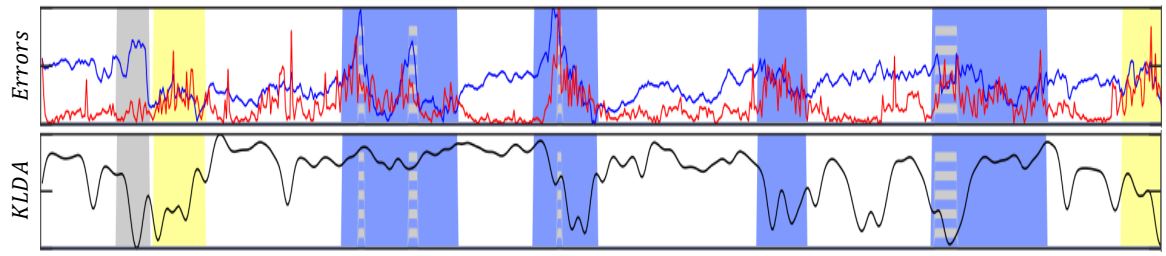


Fig. 7.12 Anomalies on U-turn using CG-KVAE. Blue interval: main anomaly (pedestrian+U-turn); grey: shadows; yellow: curve. The blue plot is the anomaly on a_t and the red plot is the anomaly on z_t. © 2023 IEEE.

The performance of the Emergency Stop dataset cannot be compared with the one in Chapter 5 because different parts of the PM and ES datasets were used in the two chapters. In this chapter, PM_1 was tested against ES_1, whereas in Chapter 5 the full PM dataset (further augmented) was tested against ES_2.

Testing of the model on the PA and U-turn data

The model is also tested on the PA and U-turn data, using MD_2. Fig. 7.11 and Fig. 7.12 display the anomalies on PA and U-turn, respectively. Blue zones represent the main anomaly: pedestrian presence and avoidance maneuver for PA; pedestrian presence, U-turn, and curves in the opposite direction to that of training for U-turn. Grey areas correspond to shadows not present in training. Yellow zones are curves (normal motion).

Note that some shadows are present in the PA and U-turn data that are not present in training; these shadows are the cause of the two peaks in Fig. 7.11 (above), one before the avoidance zone and one after it.

Fig. 7.10b and Fig. 7.10c show the ROC curves for the anomalies using the original KVAE and the CG-KVAE, for the PA and U-turn case, respectively. For these two datasets, the performance with the CG-KVAE is slightly better, but not significantly so.

We can observe that the method performs worse than the ones proposed in Chapter 5. Note, however, that the model is more computationally complex, as backpropagation is performed on batches of images across multiple time instants and not on a single time instant. Therefore, the images were reduced to a lower dimensionality (64x64 vs the 120x240 used in Chapter 5) and the VAE is smaller (it has three convolutional layers with $32, 64, 128$ channels against five convolutional layers with $16, 32, 64, 128, 128$ channels). Consequently, the comparison is not fair, being the two models so different.

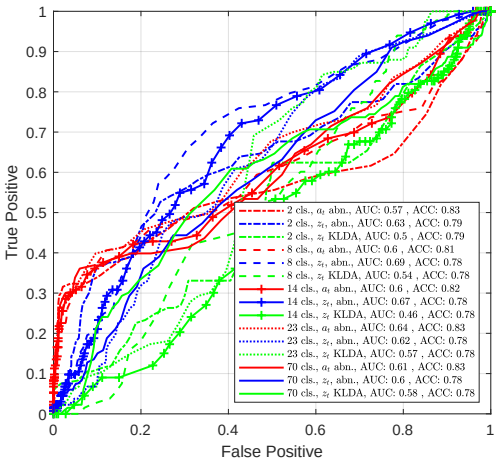
Comparison using several cluster experiments

The clustering over the odometry data can be optimized using a procedure similar to the ones proposed in [103] and in [106], i.e., several clustering experiments can be performed, and the one that performs better according to a proposed functional can be extracted. In [103] and [106], clusters were sought to have a high covariance across position and a low covariance across velocity (i.e., a precise prediction model with the highest coding capability). A trade-off between these two requirements can be found, which corresponds to a trade-off between the number of clusters and the precision of their prediction model. In [106], the optimal trade-off number of clusters on the PM odometry task over several clustering methods was found to be between 7 and 14 clusters, despite 23 giving a higher precision.

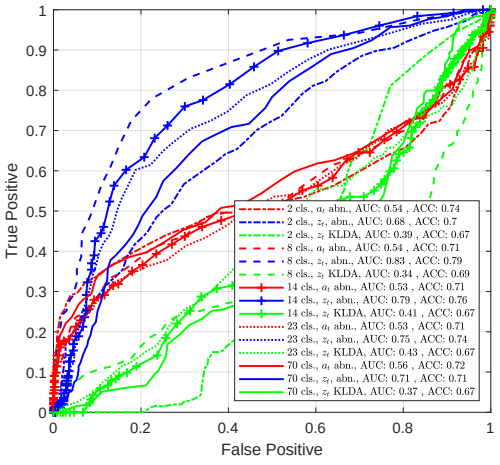
Experiments are conducted varying the GNG parameters, in particular the utility threshold κ defined in [103]. The clustering algorithm consequently stops at a different number of clusters. Training of the CG-KVAE and testing on the PA and U-turn datasets is performed with five different cases: $2, 8, 14, 23$ and 70 clusters. Fig. 7.13 shows the obtained ROC curves using the states and cluster anomalies. It can be observed that the same anomaly type displays a similar behavior across experiments, with the z_t anomaly having the best performance. The worst results correspond to $K = 2$ and $K = 70$: in the first case, the models are less precise because they merge different behaviors; in the second case, a too high number of clusters leads to over-fitting. The best results are obtained with 8 and 14 clusters. Fig. 7.13c displays the centroids for the case with 8 clusters. Notice how the division is between four straight zones and four curving zones, which is indeed the most natural division of the courtyard area by human interpretation, too.

Testing with different CG-KVAE models

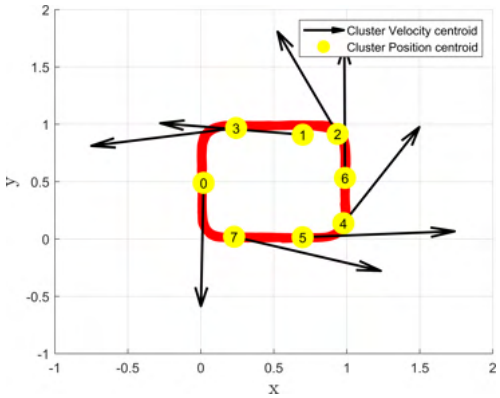
In this section, results are analyzed when varying the CG-KVAE model. As observed in section 7.2.2, the odometry model can be built in different ways. The DBN with linear



(a)

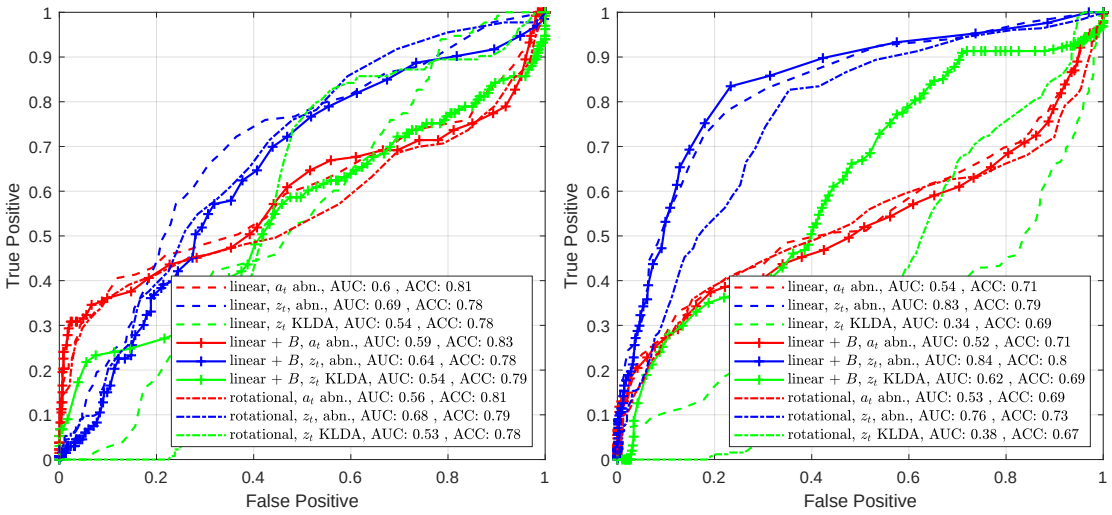


(b)



(c)

Fig. 7.13 ROC curves with varying cluster parameters: (a) PA, (b) U-turn. 'cls' = clusters, (c) PM data (red) with cluster centroids (in yellow for position and in black for velocity) for the clustering giving best results (8 clusters). © 2023 IEEE.



(a) (b)

Fig. 7.14 ROC curves with varying models: (a) PA, (b) U-turn. 'linear' refers to the CG-KVAE with linear models in the odometry clusters [17]; 'linear + B' has the same odometry models as 'linear', but adds the optional matrices B_i in the video model; 'rotational' uses rotational models in the odometry clusters as in Chapter 6. © 2023 IEEE.

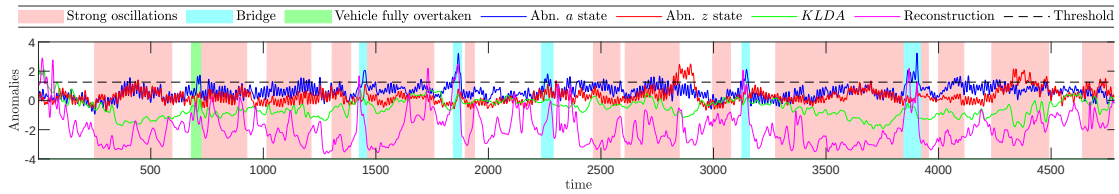
predictions introduced in [17] ('linear') is compared with the method in Chapter 6, which clusters also leveraging the rotation angle and acceleration and uses rotational models ('rotational').

Another experiment is performed using the linear odometry case again and considering the addition in the video model of the optional control matrices B_i mentioned in section III-D ('linear + B'). As control vector u_t, the velocity part of the $\tilde{z}_t^o$ state is used, i.e., $\dot{z}_t^o$. This version, therefore, supposes that a complete knowledge of the odometry velocity data is passed to the video model instead of the cluster probabilities only.

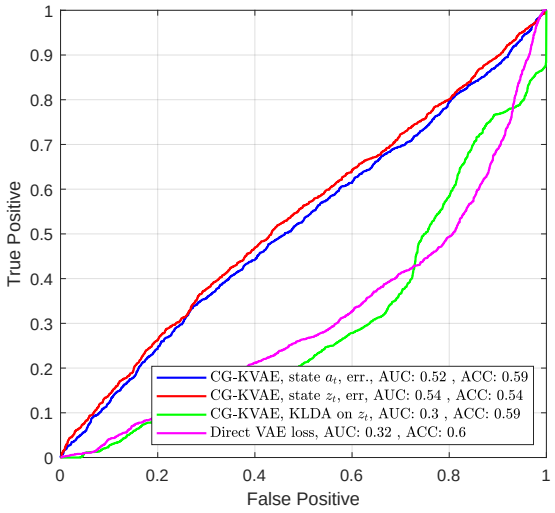
Fig. 7.14 reports the ROC curves of these experiments on the PA and U-turn cases. All cases use 8 clusters. The three models display comparable results, with the 'rotational' one performing slightly worse (possibly due to the use of the more noisy acceleration information) and with the 'linear + B' performing better on U-turn for what concerns the KLDA anomaly (due to the added direct connection between states z_t and z_t^o).

7.3.3 Results on the UAH-DriveSet dataset

First, the model is trained using the sequences of the UAH-DriveSet dataset on the normal trajectories moving in the same direction as the drowsy ones, keeping one trajectory out for validation. Then, testing is performed on the training and validation trajectories, extracting



(a)



(b)

Fig. 7.15 (a) Anomalies on the third drowsy trajectory; (b) Corresponding ROC curves. © 2023 IEEE.

the anomalies, and, for each one of them, calculating the mean abn_{mean} and the variance abn_{var}. This allows us to define an anomaly threshold to be used in testing:

$$thresh = abn_{mean} + \beta * abn_{var}, \tag{7.12}$$

We set $\beta = 1.5$. Such a threshold defines as abnormal those points with anomaly value differing from the mean of more than β times the variance, e.g., 68.3% of training/validation data if $\beta = 1$, 86.64% if $\beta = 1.5$. We take $\beta = 1.5$ to leave a margin for observed training and validation anomaly false positives.

Finally, testing is performed on the drowsy trajectories, extracting the anomalies. Note that the UAH-DriveSet dataset is not an anomaly detection dataset; moreover, there is no ground truth defining which parts of the drowsy trajectories actually fall into said behavior and which instead are normal. Consequently, we selected one of the drowsy trajectories and manually marked events that we observed in the first 4500 time instants. The chosen trajectory was the third one; the corresponding driver performed clear oscillating behaviors,

(a)

(b)

(c)

Fig. 7.16 Examples of zones from the third drowsy trajectory giving high anomalies. (a): Overtaking ($t \sim 700$); (b): Bridge ($t \sim 1450$); (c): Oscillations ($t \sim 4400$). © 2023 IEEE.

while others appeared more normal. Fig. 7.15a uses colors to define the zones. Red zones correspond to strong oscillations; bridges - which are normal observations - appeared in the cyan areas; a vehicle fully overtakes the driver's one in the green region.

Fig. 7.15a shows the anomalies at the different levels: *i)* the direct reconstruction anomaly in violet; *ii)* the state anomalies, in blue on a_t, and in red on z_t; *iii)* the KLDA in green. All the displayed anomalies are normalized relative to the corresponding abn_{mean} and abn_{var} and the threshold at 1.5 is shown.

First, it can be observed that zones where a bridge is present always result in high anomalies, despite them being normal. As these zones are rare and are composed of darker parts and higher contrast, as can be seen in Fig. 7.16b, they easily result in higher anomaly values. However, it can be deduced that this type of anomaly is derived from a reconstruction problem since it also causes the highest direct reconstruction errors.

Another zone giving high abnormality is the one where the vehicle is fully overtaken, displayed in Fig. 7.16a.

Finally, it can be observed how oscillating maneuvers - typical of drowsy behavior - give higher abnormalities than other ones. An example of this behavior, causing the peak around time $t \sim 4400$, is shown in Fig. 7.16c.

Fig. 7.15b shows the ROC curves using CG-KVAE and direct VAE reconstruction. The performance of the latter method is very poor because it only considers one image at a time and, therefore, is not able to catch anomalies at the dynamic level (which are intrinsic in oscillating cases). The perfomance is worse than the random chance classifier (ACC = 0.5). This results also derives from the inability of the used VAE to well reconstruct normal but rare images such as the bridges. In contrast, many anomalies are motion anomalies, which cannot be detected through reconstruction. Using the CG-KVAE, the performance is still very low, but significantly improved.

7.4 Conclusions

This chapter proposes a method to learn a predictive model for high-dimensional data (video) using a CG-KVAE assisted by models learned on low-dimensional data (odometry). An HDBN framework is combined with a DL method (a VAE). The model learns an object's latent representation a_t and its latent state z_t, describing its dynamics from raw video and odometry data. The odometry cluster probabilities are leveraged to combine the prediction and pseudo-observation matrices composing part of the video model. During testing, predictions from the algorithms are compared with updates to extract multi-level anomalies. Experiments showed that the model could predict future frames and detect anomalies using real-world video sequences. Several testing experiments were performed, varying the model and its parameters.

In the next chapter, the method is extended to perform localization in a known environment by modifying the training model and the MJPF algorithm. Supposing not to be provided at testing with the odometry anymore, we can estimate it by leveraging the learned vocabulary and a probability vector estimated from the images only.

Chapter 8

Vehicle Localization and anomaly detection for Video Surveillance in a Dynamic Bayesian Network Framework

In the previous chapters, we focused on anomaly detection. In this chapter, instead, Visual-Based Localization (VBL) is tackled by extending the approach proposed in Chapter 7. Two alternative methods are introduced and evaluated.

The rest of the chapter is structured as follows. After introducing the problem in Section 8.1, we present the two versions of the approach in Sections 8.2 and 8.3, respectively. Section 8.4 compares the two versions from the theoretical point of view. We then describe a method to optimize the clustering in Section 8.5. Section 8.6 discusses extensive experimental evaluation on both terrestrial and drone datasets. Finally, Section 8.7 draws some conclusions.

8.1 Introduction

Human beings possess the ability to localize themselves in a familiar environment and to detect surprising events happening in it compared to previous experiences [61, 135].

In less complex animals, such as insects like honeybees and ants, localization relies on calculating movement direction and speed, combined with compass cues like the position of the sun [95]. This process is called path integration. However, no map of the environment seems to be built using external landmarks. In contrast, more complex animals like humans and other mammals combine path integration with landmark navigation and integrate information from their senses to build a map of the environment that they have explored [92, 61].

When novelties appear in the learned environment, they can detect the change and take a response.

Similar to how humans use their different senses to localize and detect changes, surveillance vehicles are equipped with a diverse set of sensors. We have seen in Section 3.1 that sensors belong to two classes: proprioceptive and exteroceptive. Proprioceptive sensors allow gaining awareness of one's own state (self-awareness); exteroceptive sensors allow gaining awareness of the surrounding environment (situational awareness). Examples of these two classes are odometric sensors (e.g., wheel odometry) and cameras, respectively. Proprioceptive sensors are directly connected to path integration. On the other hand, cameras offer information not only about the vehicle's ego-motion but also about external landmarks.

The abilities to localize in a known environment and to detect new situations find their corresponding Artificial Intelligence fields in localization and anomaly detection. When localization is performed at test time using solely visual information, we talk about VBL (see Section 3.6). In VBL, during training, camera data and the corresponding odometry data are used to learn a model for localizing the agent using the images as input. During testing, the agent's positions are not provided anymore, and the learned model - e.g., the learned map - is used to localize from the images.

Simultaneously localizing oneself and detecting anomalies is important for tasks such as video surveillance and fault detection with mobile robots. Consider a robot patrolling a pre-defined route. To perform the correct actions to follow the route, it constantly needs to know its location. Moreover, it must detect mobile or stationary anomalies, such as intruders or suspicious objects [32]. Another relevant example is that of drones supervising ships to detect faults such as broken cables or leakages.

This chapter tackles the problem of simultaneous VBL and anomaly detection for surveillance vehicles. The following hypotheses are given:

- A: we suppose that training and testing are performed in the same environment, with varying conditions. This setting is coherent with a video surveillance application in which a robot always moves in the same building or ship;
- B: we suppose to have an estimation of the agent's position during the training phase to learn the correlations between video and odometry data.

We extend our work introduced in Chapter 7. Consequently, our objective is not solely to perform vehicle localization, but to do it coherently with our self-awareness framework for autonomous systems [183]. We combine DL-based methods, such as VAEs, [122] with traditional Signal Processing methods, such as filtering based on DBNs [80]. Our objective is to leverage the performance of CNNs in learning a suitable low-dimensional latent space

of the video data without resulting in a black-box approach. We obtain a method that is hierarchical, probabilistic, and explainable. DBNs also allow learning correlations between multiple sensory modalities, i.e., the odometry and the video, making the approach multi-sensorial. Additionally, the proposed method is data-driven: only the video and odometry data of the sensors are provided, and the models are built using them without any prior information, model, or hand-crafted features. The camera calibration parameters are also not necessary. The five properties described in Sections 3.2.3 and 3.3 are, therefore, satisfied.

In the proposed method, during training, we learn a DBN model for predicting the evolution of the odometry data, as in [17]. The odometry model is passed to the video model learning block, which trains the CG-KVAE introduced in the previous chapter. However, we propose some modifications in the learning of the CG-KVAE, directed to the VBL application. Two alternative versions of the method are presented. In both versions, two matrices are inserted in the model to predict the state of the odometry from the state of the video. In the first version (FV), the rest of the training remains the same as in chapter 7. In the second version (SV), the clustering procedure is modified: instead of clustering only on the odometry state, we cluster on both the odometry and video latent state. This allows us to improve the localization of the vehicle. In both versions, a Coupled DBN (CDBN) is obtained from the odometry and video models. Only video data are provided during testing, and filtering is performed using a Coupled Markov Jump Particle Filter (CMJPF). The CMJPF is a modification of the MJPF [17]. The CMJPF allows us to perform simultaneous localization and anomaly detection. The anomalies are both a final output (signaling whether something unusual has happened) and an intermediate result leveraged to guide the localization. Anomalies explain possible model failures, increasing the explainability of the method.

Furthermore, we examine how to choose an appropriate clustering, to avoid the problems of over-clustering (i.e., when there are more clusters than necessary) and under-clustering (i.e., when there are not enough clusters, and the method's accuracy decreases). We propose an approach based on the covariance of the video and odometry states in each cluster, following the method proposed in [103], where clustering optimization was performed only on the odometry data.

To summarize, this chapter's main contributions are:

- the learning of a DBN-based model in a self-awareness framework to couple a low-dimensional and a high-dimensional sensory modality and to learn how to predict the state of the low-dimensional modality from the state of the high-dimensional one;

- the introduction of the CMJPF: whereas the MJPF allowed us to perform anomaly detection on a single sensor modality, the CMJPF combines the learned models over video and odometry to execute VBL and anomaly detection;
- the use of the extracted anomalies to improve the localization process in a surveillance application, to predict where it is failing, and to explain why;
- the introduction of a method for both VBL and anomaly detection that is data-driven, hierarchical, probabilistic, explainable, and does not require prior models (e.g., camera calibration parameters);
- the comparison of two alternative versions of the method;
- the proposal of an approach to choose an approximate clustering level.

8.2 First Version of the Proposed Approach

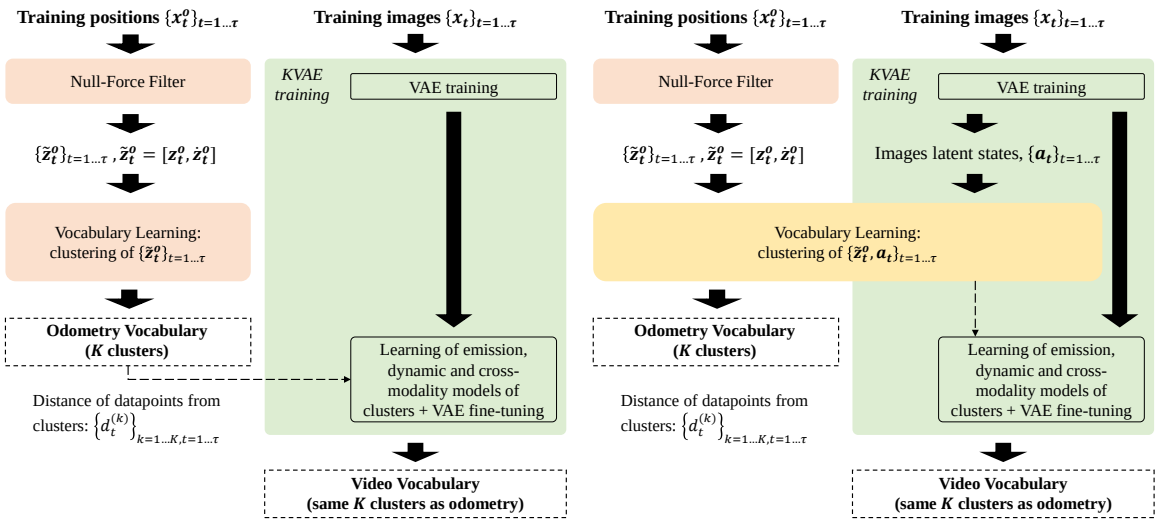
We start by describing in detail the first version of the approach.

The proposed method is divided into two parts: an offline training phase and an online testing phase. During the training phase, the DBN models for odometry and video data (First Person Viewpoint) are extracted. In the testing phase, the video data are provided, and the learned models are used to predict the values of the odometry data and to extract anomalies. This division mirrors the one seen in the methods proposed in the previous chapters. Compared to all previous chapters, however, there are differences in the inputs and outputs of the testing phase. The training phase takes as input both odometry and video data. In contrast, the testing phase only takes as input the video data. Furthermore, not only the anomalies are predicted, but also the localization.

Fig. 8.1a shows the training procedure from a very high-level point of view. Fig. 8.2 displays both training and testing, and provides further details.

8.2.1 Training of the odometry model

The training of the odometry model is conducted in the same way as seen in Section 7.2.2. To summarize, GSs $\tilde{z}_t^o$ are extracted from the positional data points x_t^o and are clustered with the GNG algorithm, obtaining K clusters and an odometry vocabulary. The odometry vocabulary includes cluster centroids $M^{(\tilde{S},o)}$, cluster covariances $Q^{(\tilde{S},o)}$, an overall transition matrix Π, and TTMs $\Pi^{(g)}$. Finally, the distance $d_t^{\tilde{S}}$ of each data point from each cluster is calculated.



(a) First version. (b) Second Version.

Fig. 8.1 Training phase in the first (a) and second (b) versions of the proposed anomaly detection and localization approach. In the first version, the odometry state is extracted by filtering the training positions, and is then used to perform clustering. The odometry vocabulary guides the learning of the KVAE, which leads to the extraction of the video vocabulary. In the second version, the clustering is performed using both the odometry states and the video latent states obtained from the VAE bottleneck.

8.2.2 Training of the video model

The training procedure of the video model is similar to the one presented in Section 7.2.4. Therefore, we summarize the common parts and focus on describing what is modified.

First, from the distances $d_t^{\tilde{S}}$ obtained from the odometry training, it is possible to estimate the probability $\alpha_t^{(\tilde{S})}$ that the odometry point at time t belongs to cluster $\tilde{S}$:

$$\alpha_t^{(\tilde{S})} = softmax\left(\frac{1}{d_t^{(\tilde{S})} + \varepsilon} * \frac{1}{m}\right), \tag{8.1}$$

where m is a temperature, and ε is a small positive value added to avoid the denominator being zero.

At each time instant t, an image x_t is passed through the encoder q_ϕ of a VAE, to obtain a first latent state a_t. A second latent state, z_t, is also found, which has a smaller dimension compared to a_t. Whereas a_t captures the content information of the images, z_t focuses on the dynamics between images. The two latent states are correlated through a pseudo-observation

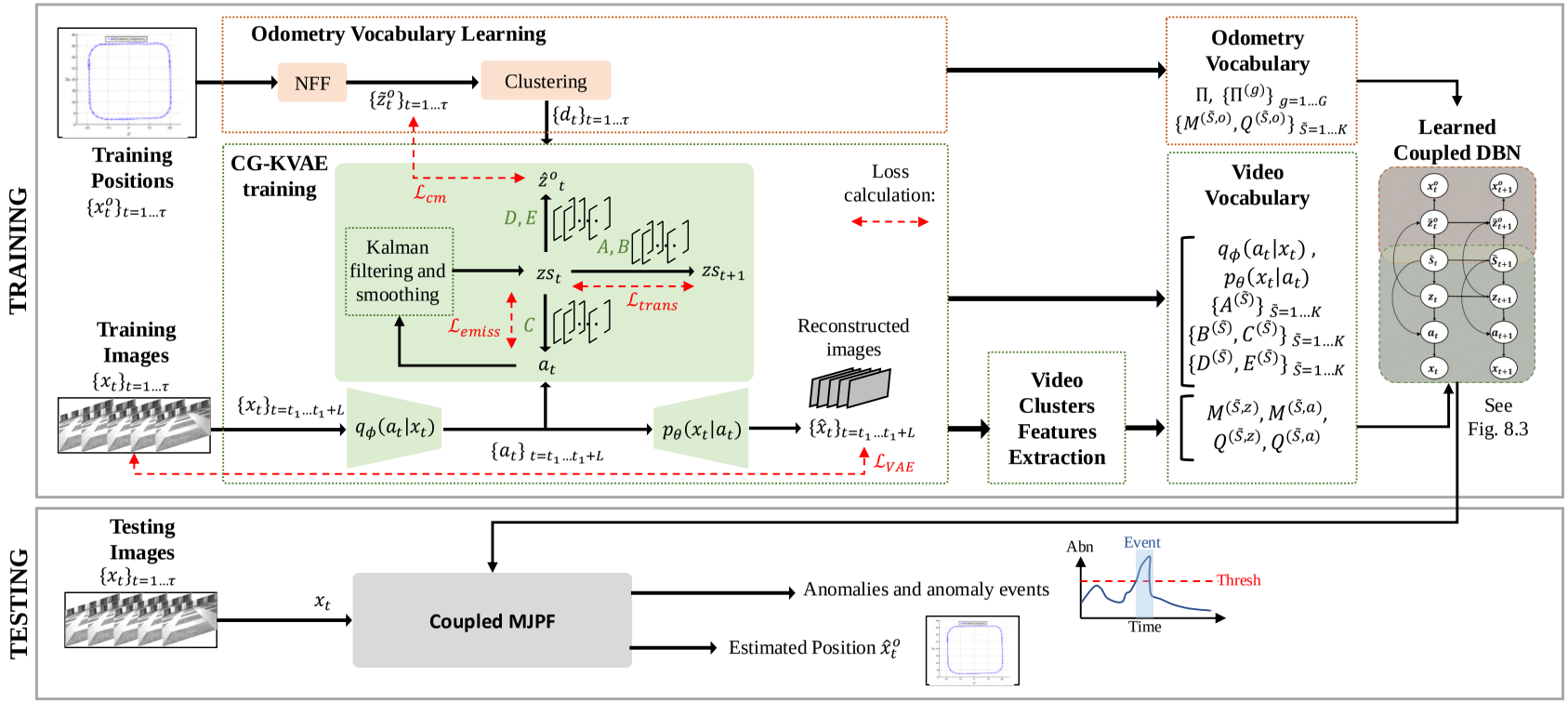


Fig. 8.2 Training phase (Section 8.2.1, Section 8.2.2 and Section 8.2.3) and testing phase (Section 8.2.4) of the method (FV). During the training phase the odometry vocabulary is first built (Section 8.2.1). Then, a modified KVAE is learned, leveraging also the information from the odometry vocabulary (Section 8.2.2). The latent states obtained from the training images are used to extract the features of the video clusters. A Coupled DBN is built from the odometry and video vocabulary (Section 8.2.3). During the testing phase (Section 8.2.4), a Coupled MJPF, constructed considering the Coupled DBN, extracts anomalies and performs localization.

model defined as:

$$a_t = \sum_{i=1}^{K} \alpha_t^{(i)} C^{(i)} z_t + v_t \tag{8.2}$$

The dynamical model connects z_t values at consequent times:

$$z_{t+1} = \sum_{i=1}^{K} \alpha_t^{(i)} A^{(i)} z_t + \sum_{i=1}^{K} \alpha_t^{(i)} B^{(i)} + \omega_t \tag{8.3}$$

The matrices $\{C^{(i)}\}_{i=1\ldots K}$, $\{A^{(i)}\}_{i=1\ldots K}$, represent a set of pseudo-observation models and of transition models, respectively. They are combined through the vector of probabilities $\boldsymbol{\alpha}_t$. The original KVAE method considered a set of matrices $\{B^{(i)}\}_{i=1\ldots K}$, which multiplied an optional control vector. In contrast, in this chapter, we consider a set of matrices B providing an addition term. Through these models, we can loosely associate with an odometry cluster a different way of selecting features from the VAE bottleneck (through the $C^{(i)}$ matrices) and a different image variation across time ($A^{(i)}$ matrices). This choice will enable us during testing to distinguish whether something abnormal has happened in the content of the images or in their motion, increasing the explainability of the method, as we can identify to what model the anomaly is related.

For training the KVAE, at each loop of each epoch, we consider a batch of image sequences of length L, i.e., $\{x_t\}_{t_1\ldots t_1+L}$. The batch is passed through the encoder $q_\phi(a_t|x_t)$ of the KVAE to obtain the latent states $\{a_t\}_{t_1\ldots t_1+L}$. Kalman filtering and smoothing are then performed on these states, using the observation model in Eq. 8.2 and the prediction model in Eq. 8.3. From this point onwards, we define with z_t the low-dimensional latent state extracted using filtering only, and with zs_t the one after applying the smoothing process too.

Since for localization we also desire to relate a value of the latent state z_t inside a cluster to its corresponding odometry state $\tilde{z}_t^o$, we additionally train a set of matrices $\{D^{(i)}\}_{i=1\ldots K}$ and $\{E^{(i)}\}_{i=1\ldots K}$, such that:

$$\hat{z}_t^o = D^{(\tilde{S}_t)} zs_t + E^{(\tilde{S}_t)} + M^{(\tilde{S}_t,o)} + m_t, \tag{8.4}$$

where $\hat{z}_t^o$ is the estimated odometry state at time instant t and m_t is the residual error. To ease the training, we make the estimation of $\tilde{z}_t^o$ start from the odometry cluster centers $M^{(\tilde{S}_t,o)}$.

Consequently, we add a *cross-modality loss* $\mathscr{L}_{cm}$:

$$\mathscr{L}_{cm} = \sum_{\tilde{S}=1}^{K} \alpha_t^{(i)} \left\| \hat{z}_t^o - \tilde{z}_t^o \right\|^2 \tag{8.5}$$

Fig 8.2 indicates the loss calculations with red dotted lines.

At the end of the KVAE training, for each cluster $\tilde{S}$, we extract the cluster centroids $(M^{(\tilde{S},a)}, M^{(\tilde{S},z)})$ and the cluster covariances $(Q^{(\tilde{S},a)}, Q^{(\tilde{S},z)})$ over the values of both latent states a_t and z_t.

8.2.3 Learned Coupled Dynamic Bayesian Network

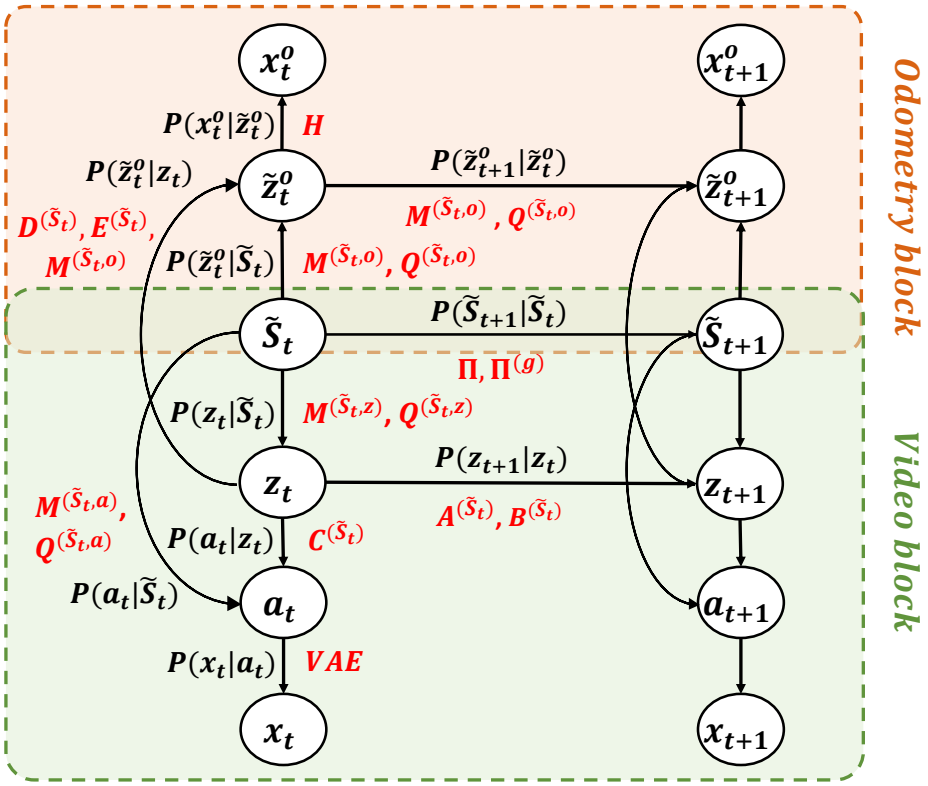


Fig. 8.3 Learned Coupled Dynamic Bayesian Network (CDBN). © 2022 IEEE.

During training, two DBNs are learned, as shown in Fig. 8.3. The features representing the links are shown in red.

The DBN is very similar to the one learned in the previous chapter, i.e., see Fig. 7.3. A link is added between the video state z_t and the odometry GS $\tilde{z}_t^o$, represented by Eq. 8.4. This link enables to predict the odometry state from the video state. Furthermore, we add in the image the link between a_t and $\tilde{S}_t$, represented by $M^{(\tilde{S}_t,a)}$ and $Q^{(\tilde{S}_t,a)}$, which will be adopted to estimate the current cluster, given the value of a_t. This link was already learned in Chapter 7, but was not employed in the method, so we chose not to display it in the figure, for simplicity and improved readability. In this chapter, instead, this link is exploited during the tracking.

8.2.4 Combined MJPFs for estimating odometry from video.

During the online phase, at each time instant, only the video data x_t is given, and we derive the vehicle's localization and the anomalies using the information of the CDBN in Fig. 8.3. The traditional MJPF in [17] was built on the model of a single sensor. In this chapter, a CMJPF is proposed, coupling the models of two sensors, where only one is observed.

We have discussed the MJPF in Chapter 2. We summarize here the most relevant information to understand the CMJPF. The MJPF uses of a set of KFs at the state level and a PF at the cluster level. A PF maintains a set of N particles during filtering, with each particle representing a hypothesis and being associated to a mean, a covariance, and a cluster at each time step. At each instant, a prediction and update phase are executed. Prediction and update at the state level adopt the state prediction and update models related to the cluster of the particle. The cluster-level prediction employs the transition matrices Π and $\Pi^{(g)}$. During the update phase, particles undergo resampling to eliminate the least probable hypotheses.

The proposed CMJPF is based on the original MJPF but includes significant changes. Firstly, each particle is associated with two means (one for the odometry GS and one for the video state), two covariances, and a cluster. Secondly, the prediction and the update at the state level are split into two steps, as they are performed both for the odometry GS and for the video state. Conversely to a classic odometry MJPF, in the proposed CMJPF, we don't have the odometry observation. Consequently, we note that the odometry 'update' is not an actual update, as the positions are not observed. This update is in fact a prediction performed through the vocabulary of the video model and the D and E matrices. Further details regarding this point will be provided on the next page.

Similarly to the original MJPF, the prediction is performed at the cluster level leveraging the matrices Π and $\Pi^{(g)}$.

Additionally, in the video case, note that pseudo-observation and state prediction are not performed through the sum over the $\boldsymbol{\alpha}_t$ values (see Eq. 8.2 and Eq. 8.3). Instead, the PF of the CMJPF is used to keep several cluster hypotheses in parallel, on a set of N particles, as in the original MJPF.

In the following paragraphs, we describe the steps executed by the algorithm at each time instant t (see also Fig. 8.4 and Alg. 8).

Particle-independent calculations

At each instant t, we perform a set of operations that do not depend on the N particles. The image x_t is passed through the VAE's encoder q_ϕ to extract the latent state a_t. The Bhattacharyya distance D_B is calculated between each video state and each video cluster:

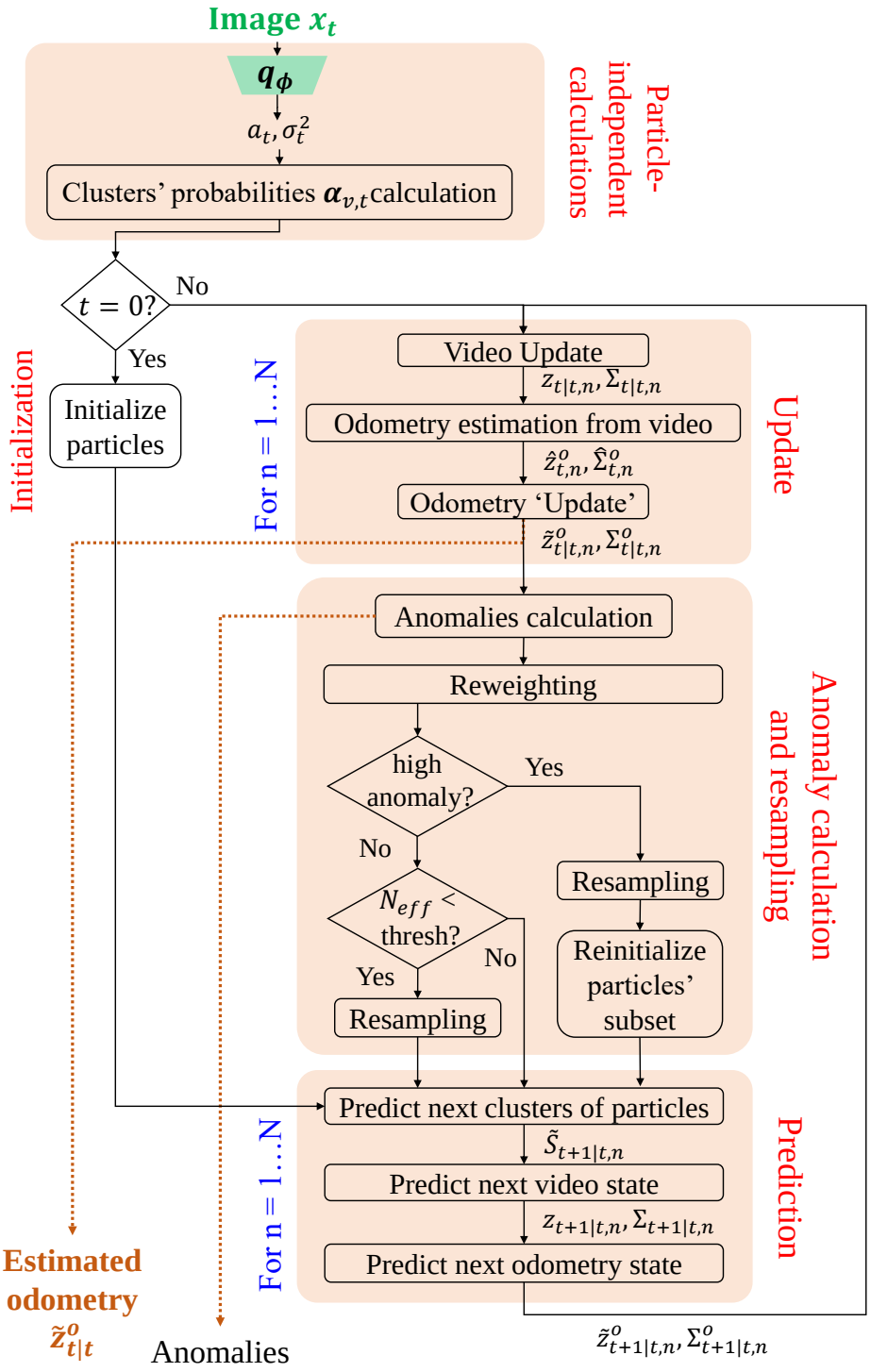


Fig. 8.4 Scheme of the testing procedure.

Algorithm 8 Method for obtaining odometry from video.

Input: Learned DBN vocabulary
$first\_resampling\_performed = False$
for t = *1* ***to*** $\tau \leftarrow$ *Time evolution* **do**
  PARTICLE-INDEPENDENT CALCULATIONS:
  $x_t \leftarrow$ Image at t
  Get the latent state of images using the VAE's encoder:
  $a_t = q_\theta(x_t) \leftarrow$ Image latent state
  Find the distances of video encodings from video clusters:
  $d_{v,t}^{(\tilde{S})} = D_B((a_t, \Sigma_t^a), (M^{(\tilde{S},a)}, Q^{(\tilde{S},a)}))$, for $\tilde{S} = 1 \dots K$
  Find the video cluster probabilities $\alpha_{v,t}^{(\tilde{S})}$ from the distances $d_{v,t}^{(\tilde{S})}$, for $\tilde{S} = 1 \dots K$, using Eq. 7.6.
  if t == *1* $\leftarrow$ *Initial iteration* **then**
    INITIALIZATION OF THE PARTICLES:
    for n = *1* ***to*** $N \leftarrow$ *Particles* **do**
      Initialize cluster assignment $\tilde{S}_{t=1,n}$ of particle n using cluster probabilities $\alpha_{v,t}^{\tilde{S}}$, with $\tilde{S} = 1 \dots K$.
      Initialize n-th particle video state as:
      $z_{t=1,n} \sim \mathcal{N}(M^{(\tilde{S}_{1,n},z)}, Q^{(S_{1,n},z)})$.
      Initialize n-th particle odometry state as:
      $\tilde{z}^o_{t=1,n} \sim \mathcal{N}(M^{(\tilde{S}_{1,n},o)}, Q^{(\tilde{S}_{1,n},o)})$.
    end
  end
  else
    UPDATE:
    for n = *1* ***to*** $N \leftarrow$ *Particles* **do**
      Perform the video update: $z_{t|t,n}, \Sigma_{t|t,n} = KF\_update(z_{t|t-1,n}, \Sigma_{t|t-1,n}, C^{(\tilde{S}_{t,n})}, a_t, R^v)$
      Get the odometry estimation from video: $\hat{z}^o_{t,n} = D^{(\tilde{S}_{t,n})} z_{t|t,n} + E^{(\tilde{S}_{t,n})} + M^{(\tilde{S}_{t,n},o)}$
      Perform the odometry 'update': $\tilde{z}^o_{t|t,n}, \Sigma^o_{t|t,n} = KF\_update(\tilde{z}^o_{t|t-1,n}, \Sigma^o_{t|t-1,n}, \hat{z}^o_{t,n}, R^o)$
    end
    Calculate the anomalies. Based on them, assess if restarting particles is necessary.
    Reweight the particles.
    if *Particles Restarting necessary* **then**
      Resample, restart $n * P_{par}$ particles and set $first\_resampling\_performed$ == True.
    end
    else if $first\_resampling\_performed$ == *False and* $N_{eff} < N_{th,l}$ **then**
      Resample particles and set $first\_resampling\_performed$ == True.
    end
    else if $N_{eff} < N_{th,h}$ **then**
      Resample particles.
    end
  end
  PREDICTION:
  for n = *1* ***to*** $N \leftarrow$ *Particles* **do**
    Predict the next cluster $\tilde{S}_{t+1,n}$ from $\tilde{S}_{t,n}$ using Π and $\Pi^{(g_{t,n})}$.
    Predict the video state: $z_{t+1|t,n}, \Sigma_{t+1|t,n} = KF\_v\_pred(z_{t|t,n}, \Sigma_{t|t,n}, A^{(\tilde{S}_{t,n})}, Q^v)$.
    Predict the odometry state: $\tilde{z}^o_{t+1|t,n}, \Sigma^o_{t+1|t,n} = KF\_pred(z^o_{t|t,n}, \Sigma^o_{t|t,n}, M^{(\tilde{S}_{t,n},o)}, Q^{(\tilde{S}_{t,n},o)})$.
  end
end
Output: Anomalies and $x^o_{t|t,n}$ with $n = 1 \dots N$

$d_{v,t}^{(\tilde{S})} = D_B((a_t, \Sigma_t^a), (M^{(\tilde{S},a)}, Q^{(\tilde{S},a)}))$. Finally, a vector $\boldsymbol{\alpha}_{v,t}$ of probabilities is obtained from the set of K distances $d_{v,t}^{(\tilde{S})}$ using Eq. 8.1. It defines how probable it is to be in each cluster, based on the video observation. Note that the best temperature m_v to use in this case might not be the final one used in training, i.e., m, as we are now working on the video state and not on the odometry state anymore.

Initialization of the particles

At the first time instant, the particles of the CMJPF are initialized. This means defining, for each particle, the cluster, the mean, and covariance of both the odometric and video state. The cluster $\tilde{S}_{t=1,n}$, of particle n, is sampled using the multinomial distribution described by $\boldsymbol{\alpha}_{v,t=1}$. Mean and covariance of video and odometry states of the particle can then be defined sampling from the Gaussian distributions $\mathcal{N}(M^{(\tilde{S}_{1,n},z)}, Q^{(\tilde{S}_{1,n},z)})$ and $\mathcal{N}(M^{(\tilde{S}_{1,n},o)}, Q^{(\tilde{S}_{1,n},o)})$, respectively.

Prediction phase

At each instant, the next cluster, the next odometry state, and the next video state are predicted for each particle through the learned prediction models. At the cluster level, the prediction is performed by combining the transition matrix Π and the TTM $\Pi^{(g)}$, where g is the time spent in the considered cluster.

For the video state prediction, we use for each particle the A matrix of the associated cluster, i.e., $A^{(\tilde{S}_{t,n})}$, in a standard KF prediction step. We define this step in Alg. 8 with the notation $KF\_v\_pred(z_{t|t,n}, \Sigma_{t|t,n}, A^{(\tilde{S}_{t,n})}, Q^v)$, because the standard KF prediction equation is used on the video state, starting from the updated mean $z_{t|t,n}$ and covariance $\Sigma_{t|t,n}$ at time instant t, and using the prediction matrix $A^{(\tilde{S}_{t,n})}$ with the prediction uncertainty Q^v (i.e., the process noise covariance).

For the odometry part, no modification is necessary compared to the original model proposed in [17], which used a standard KF step predicting that the object will move with the mean velocity of the cluster (i.e., the second half of the vector $M^{(\tilde{S}_{t,n},o)}$).

$$\tilde{z}_{t+1/t,n}^o = A^o \tilde{z}_{t/t,n}^o + B^o U^{(\tilde{S}_{t,n})} + w_t^o \tag{8.6}$$

where $A^o \tilde{z}_{t/t,n}^o$ takes the GS information from $\tilde{z}_{t/t,n}^o$ and makes null its time derivatives. $B^o U^{(\tilde{S}_{t,n})}$ encodes the time derivative information at time t. $U^{(\tilde{S}_{t,n})}$ contains the derivative-related part of the variable $M^{(\tilde{S}_{t,n},o)}$. The variable ω_t^o is a zero-mean Gaussian distribution representing the noise of the model.

Update phase

At each time instant, except for the first one, we perform the update of video and odometry states using the current image. First, the video update is done. For each particle, the C matrix of the associated cluster is used, i.e., $C^{(\tilde{S}_{t,n})}$. We define in Alg. 8 this step with the notation $KF\_update(z_{t|t-1,n}, \Sigma_{t|t-1,n}, C^{(\tilde{S}_{t,n})}, a_t, R^v)$ because the traditional KF update equation is used on video, starting from the predicted mean $z_{t|t-1,n}$ and covariance $\Sigma_{t|t-1,n}$ at time instant $t-1$, and using the pseudo-observation matrix $C^{(\tilde{S}_{t,n})}$; the pseudo-observation is the state a_t, with uncertainty R^v (i.e., the measurement noise covariance).

To execute the odometry state update, an observation of the vehicle's position would be required, but is unavailable. To address this limitation, in the course of model training, the supplemental matrices D and E were introduced. These matrices derive an approximate estimate $\hat{z}^o_{t,n}$ for the odometry state based on the latent state $z_{t,n}$ (see Eq. 8.4). An odometry "update", denoted as $\tilde{z}^o_{t/t,n}$, is obtained following the conventional Kalman Filter equations. The update process incorporates $\hat{z}^o_{t,n}$ as the odometry observation. Note that this operation is not an actual update but rather a combination of two predictions, one using the odometry prediction model defined by $(M^{(\tilde{S}_t,o)}, Q^{(\tilde{S}_t,o)})$ and the other employing the cross-modality model defined by matrices $(D^{(\tilde{S}_t)}, E^{(\tilde{S}_t)})$. We adopt the name "update" because we combine the two predictions through the standard KF update equations. $\hat{x}^o_{t,n}$ and $x^o_{t/t,n}$, corresponding to $\hat{z}^o_{t,n}$ and $\tilde{z}^o_{t/t,n}$ are then obtained through the observation model in Eq. 7.2.

Anomaly calculation

Several anomalies are extracted after the update phase. In the experiments presented in Section 8.6, we calculate five types of anomalies:

- the image reconstruction error of the VAE, $Abn_{rec} = MSE(x_t, \hat{x}_t)$, where MSE is the Mean Squared Error;
- the MSE between the predicted and updated state a_t, averaged over the N particles i.e., $diff_a = \frac{\sum_{n=1}^{N}(a_{t|t-1,n} - a_{t|t,n})}{N} = \frac{\sum_{n=1}^{N}(diff_{a,n})}{N}$;
- the MSE between the predicted and updated state z_t, averaged over the N particles i.e., $diff_z = \frac{\sum_{n=1}^{N}(z_{t|t-1,n} - z_{t|t,n})}{N} = \frac{\sum_{n=1}^{N}(diff_{z,n})}{N}$;
- the KLDA, which signals whether a previously unseen sequence of clusters is now being followed;
- MSE between the two odometry predictions $diff_{\hat{z}^o} = \frac{\sum_{n=1}^{N}(\hat{z}^o_{t,n} - \tilde{z}^o_{t,n})}{N} = \frac{\sum_{n=1}^{N}(diff_{\hat{z}^o,n})}{N}$.

Note that each of these anomalies is related to a different level of the DBN: the image level, the a_t state level, the z_t state level, the $\tilde{S}_t$ cluster level, and the cross-modal interaction, respectively. They are used to discover what part of the model should be updated. In addition to the anomalies defined above, the minimum value in the vector $\boldsymbol{\alpha}_{v,t}$ is used as an anomaly indicator: when it is higher than normal, it signals that the model is uncertain regarding the clusters' assignment. In figures, we identify this anomaly as the "minimum $\tilde{S}$ likelihood abnormality".

Particles resampling

We reweight the particles: we give higher weight to particles belonging to clusters with a higher likelihood (accordingly to Section 3.7.3) and with a lower anomaly. As a reweighting anomaly, we use $diff_{a,n}$. Particles are resampled when the filter's effective size N_{eff} (see Section 2.4.4) is below a defined threshold [58]. We propose using two separate thresholds. The first one, $N_{th,l}$, is used at the start of the trajectory. After the first resampling, it is substituted by a second threshold, $N_{th,h}$, which is employed for the rest of the tracking. At the start of a trajectory, the algorithm needs to decide in which location of the overall space the agent is situated and might need more time - and so a lower threshold - to correctly do it. Conversely, a higher threshold can be used for the rest of the tracking. This reasoning does not apply if the vehicle always starts from the same position: in this case, there is no need to use a lower threshold at the beginning of the trajectory, because we approximately know where the vehicle is located.

Particles restarts

The algorithm can detect whether its localization prediction is diverging from the actual path that the vehicle is following. If this is the case, it needs to correct its localization. The extracted anomalies are used to estimate how certain the algorithm is of its localization and if it needs to perform re-localization. We defined a set of anomalies $\{abn_j\}_{j=1...6}$. For all elements of this set - except Abn_{rec} -, a percentage P_{par} of the particles is re-initialized when abn_j exceeds a threshold Th_j for more than a percentage P_w of a time window W_h. In the case of the reconstruction anomaly Abn_{rec}, a different approach is employed. Before re-initializing the particles, it is necessary to wait until the anomaly falls below the threshold for a designated time window W_l. This waiting period is crucial: if it were not used, the particles would restart exactly when they are less likely to be placed in the correct clusters because the high reconstruction anomaly indicates that those images were not part of the training data.

8.3 Second Version of the Proposed Approach

This section explains the second version of the proposed method. Fig. 8.1b shows the training procedure on a high level. The procedure of the SV can be easily compared with the one of the first version, on the left, in Fig. 8.1a. A more detailed description of both training and testing is offered in Fig. 8.5.

8.3.1 Training of the odometry and video models

The processing of odometry and video starts in the same way as in the FV: the odometry GSs $\tilde{z}_t^o$ are extracted through an NFF, and the video latent states a_t are obtained for all images after training a VAE. Differently from the FV, the clustering is then performed on the joint odometry-video state $j_t = [\tilde{z}_t^o, a_t]$. It is worth noting, however, that $\tilde{z}_t^o$ and a_t have different dimensionalities: $\tilde{z}_t^o$ can be either 4-dimensional or 6-dimensional, whereas a_t can have much greater dimensionality (in the results section, we chose to make it range from 16-dimensional to 128-dimensional, depending on the complexity of the dataset). If we want to give the same importance to the odometry and video subcomponents of j_t during clustering, we need to weigh them differently.

In the GNG algorithm [78], a distance is calculated between each j_t and the temporary clusters proposed by the method. Proposed cluster centers are moved closer to the input data, and the nearest cluster to each input data point is the one that gets moved with the greatest step. The used distance is a weighted MSE and is applied after performing standardization on the state. The weights of the different components are distributed so that the same importance is given to the odometry and video parts. For example, if the odometry GS is 4-dimensional, and the video latent state is 16-dimensional, each odometry component is given a weight of $\gamma = \frac{1}{4}$, and each video component is given a weight of $\delta = \frac{1}{16}$.

After executing the clustering, the same odometry vocabulary components are extracted, as in the FV case (i.e., $M^{(\tilde{S},o)}$, $Q^{(\tilde{S},o)}$, Π, and $\Pi^{(g)}$). Additionally, a first version of a subpart of the video model is extracted too, i.e., $M^{(\tilde{S},a)}$, and $Q^{(\tilde{S},a)}$.

The distance $d_t^{\tilde{S}}$ of each j_t point from each cluster is calculated. It is the summation of two distances:

$$d_t^{(\tilde{S})} = \gamma d_t^{(\tilde{S},o)} + \delta d_t^{(\tilde{S},a)}, \tag{8.7}$$

where $d_t^{\tilde{S},o}$ is a probabilistic distance (e.g., Bhattacharya D_B) between each odometry GS $\tilde{z}_t^o$ and each odometry cluster:

$$d_t^{(\tilde{S},o)} = D_B((\tilde{z}_t^o, \Sigma_t^o), (M^{(\tilde{S},o)}, Q^{(\tilde{S},o)})) \tag{8.8}$$

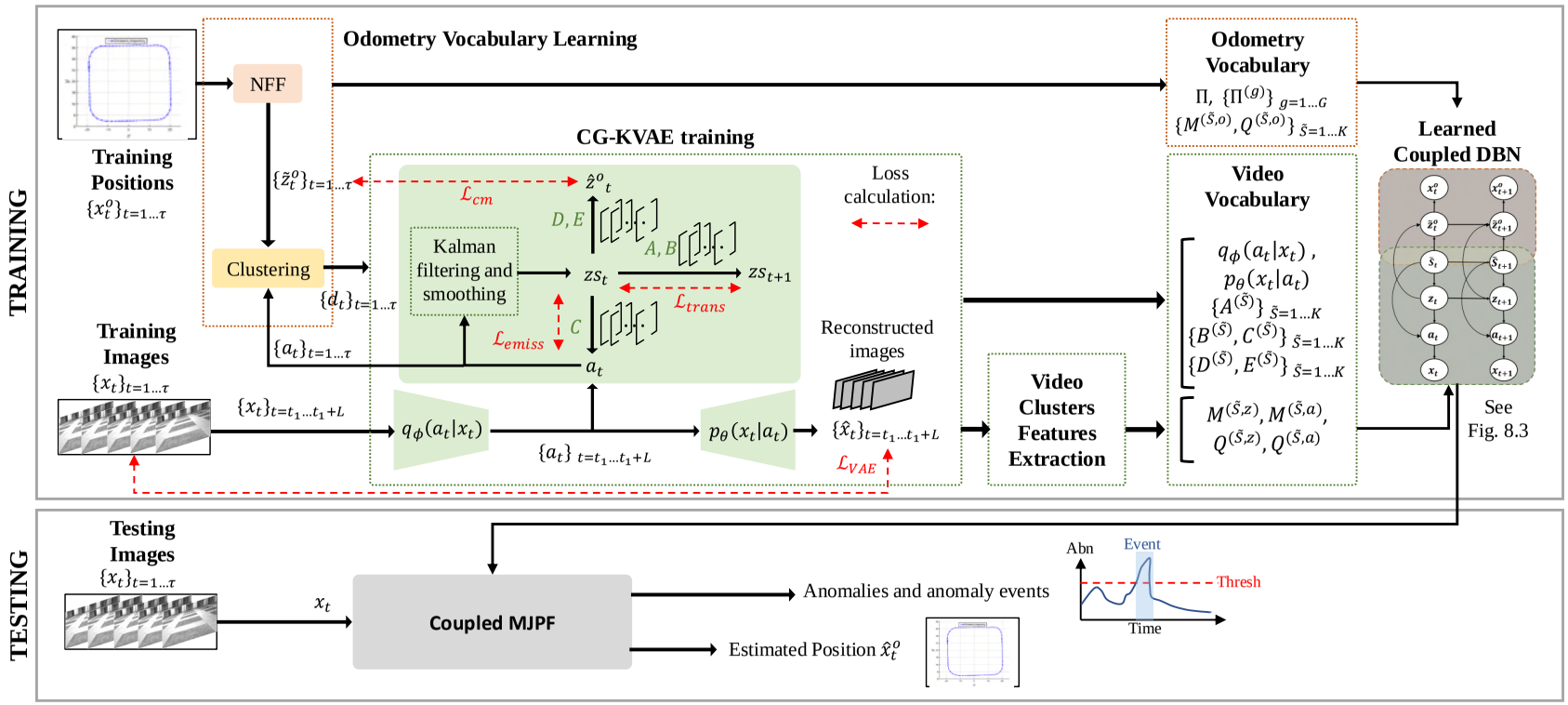


Fig. 8.5 Training phase (Section 8.3.1, and Section 8.3.2) and testing phase (Section 8.3.3) of the method (SV). During the training phase the odometry is first filtered and a VAE is built on the video data. Then, clustering is performed using both odometry and video states. A modified KVAE is learned, leveraging the information from the joint odometry-video clustering as well. The latent states obtained from the training images are used to extract the features of the video clusters (Section 8.3.1). A Coupled DBN is built from the odometry and video vocabulary (Section 8.3.2). During the testing phase (Section 8.3.3), a Coupled MJPF, constructed from the Coupled DBN, extracts anomalies and performs localization.

where Σ_t^o represents the covariance of $\tilde{z}_t^o$.

In contrast, $d_t^{\tilde{S},a}$ is the probabilistic distance between each video state $\tilde{a}_t$ and each video cluster:

$$d_t^{(\tilde{S},a)} = D_B((a_t, \Sigma_t), (M^{(\tilde{S},a)}, Q^{(\tilde{S},a)})) \tag{8.9}$$

where Σ_t^a is the covariance associated with a_t.

The training of the video model in the SV is performed in the same way as in the FV (see Section 8.2.2).

8.3.2 Learned DBN

The learned DBN can be expressed by the same structure as the one in the FV (see Fig. 8.3).

8.3.3 Combined MJPFs for estimating odometry from video.

The CMJPF in the SV of the method remains unaltered compared to the FV (see Section 8.2.4).

8.4 Comparison between the two versions of the approach

The difference between the FV and SV of the approach is the information adopted as input to the GNG clustering. The clusters of the FV are built using only odometry data, whereas the clusters of the SV are obtained employing both odometry and video data. This difference leads to some advantages and disadvantages in each version.

Firstly, the SV has a performance advantage compared to the FV. As can be easily noticed in Fig. 8.4, the first actions performed at each time instant are the particle-independent operations. These include two steps: encoding the image with the VAE and calculating the probability of being in each cluster ($\boldsymbol{\alpha}_{v,t}$), based on the VAE bottleneck. Therefore, a first estimate of the probability of being in each cluster is obtained based solely on the content of the images. The clustering in the SV of the method is built using the image information too. As a result, the first estimate of the cluster at time t will tend to be more precise in the SV compared to the FV. Final localization results are shown to be better in the SV.

In contrast, the FV has the advantage of being more modular. The odometry vocabulary is learned, without using the video data, and, then, the video block is learned. In contrast, in the SV, the learning of the odometry vocabulary depends on the video data too. This has two consequences. First, when performing continual learning, video anomalies would lead to the need to relearn the odometry vocabulary as well. Second, the CG-KVAE learning phase can

become more restricted too: instead of learning the VAE and matrices together from the start, the VAE must be learned first. To ensure comparable results, we explore the approach of initially training the VAE independently in both the FV and SV (which does not align with the original training suggestions of the KVAE paper [71]).

Another minor advantage of the first version is that the transition matrix tends to be less entropic, which potentially leads to the need of less particles to track all the hypotheses.

The comparison of the accuracy and transition matrix entropy in the two versions of the approach is conducted in detail in the result section.

8.5 Clustering Optimization

One important aspect of the proposed approach is the clustering. The input given to the clustering method (i.e., only the odometry or both the video and odometry) is not the only relevant issue. Another major problem is when to stop the GNG clustering.

GNG is a type of clustering algorithm in which the number of clusters does not need to be defined a priori. Clusters are added as the process progresses. A condition to stop the clustering can be defined: for example, the algorithm can be stopped when the mean distance between most input data points and their closest cluster center is below a certain value [99].

In this section, we propose to analyze the characteristics of the obtained clustering graphs as the algorithm progresses and to decide at the end which graph to choose. $\mathscr{L}$ levels of clustering are obtained (i.e., $\mathscr{L}$ graphs). In the SV of the approach, we desire clusters with the following main characteristics:

- the velocity prediction inside the cluster should be precise. In this way, the odometry prediction performed by the CDBN link $P(\tilde{z}^o_{t+1}|\tilde{z}^o_t)$ will tend to be more accurate;
- the image content in the cluster should be homogeneous. This choice allows for a better first estimate of the cluster probabilities from the video latent state;
- fewer clusters are desirable, so to have a simpler model requiring less memory to be stored. Therefore, clusters should extend on an area that is as big as possible while guarantee an accurate velocity prediction and image content homogeneity.

In the FV of the approach, only the first and third characteristics in the list apply. These same two considerations were made in the papers [103] and [106]. In these two works, a method for optimizing odometry clustering was proposed. The $\mathscr{L}$ clustering levels are analyzed. The average covariance over position and velocity in each level is calculated. Based on these two values (or on derived quality metrics, in [106]), the clustering level that

guarantees a tradeoff between accurate velocity and spatial extension is found. In [103], a loss function is computed based on the position and velocity mean covariances, and it is minimized. In [106], two quality metrics are extracted: one is inversely proportional to the positional covariance, increasing when the number of clusters increases, while the other is directly proportional to the velocity covariance, decreasing when the number of clusters increases; a tradeoff is selected as the clustering level closer to the intersection of the two lines.

In this thesis, we propose a similar method, extending it to the case in which also video information is present.

Let us consider the optimization procedure on the SV. First, $\mathscr{L}$ clustering levels are computed. In each clustering level, the covariance over the positional, velocity, and video latent state information is calculated. We express these three covariances at clustering level l with the following symbols: $Q^{(\tilde{S},p,l)}$, $Q^{(\tilde{S},v,l)}$, and $Q^{(\tilde{S},a,l)}$. The eigenvalues of each of the three covariances are calculated, i.e., $eig(Q^{(\tilde{S},p,l)})$, $eig(Q^{(\tilde{S},v,l)})$, and $eig(Q^{(\tilde{S},a,l)})$. This operation is repeated for each cluster in the graph. In [106], quality metrics were calculated on the position and velocity covariances by averaging the values in the covariance. In this chapter, instead, we calculate the eigenvalues of the covariances, as they are more meaningful.

The mean value of the eigenvalues, over the covariance dimension, and over the different clusters, is found for all three cases. We perform this calculation for each of the $\mathscr{L}$ graphs. Therefore, at the end, three quality metrics are obtained for each of the $\mathscr{L}$ levels, one related to the positional covariance (q^p), one to the velocity covariance (q^v), and one to the image content covariance (q^a). The definition of these three quality metrics, for the l-th graph, is summarized by the following formulas:

$$q_l^p = \frac{\sum_{k=1}^{K}\sum_{d=1}^{D^p} eig^{(d)}(Q^{(k,p,l)})}{K * D^p} \tag{8.10}$$

$$q_l^v = \frac{\sum_{k=1}^{K}\sum_{d=1}^{D^v} eig^{(d)}(Q^{(k,v,l)})}{K * D^v} \tag{8.11}$$

$$q_l^a = \frac{\sum_{k=1}^{K}\sum_{d=1}^{D^a} eig^{(d)}(Q^{(k,a,l)})}{K * D^a} \tag{8.12}$$

In the first equation, $eig^{(d)}$ refers to the d-th eigenvalue, and D^p is the dimensionality of the positional covariance (i.e., either 2 or 3). The same notation rules apply for the second and third formulas but refered to the velocity and video latent state covariances.

From the three desired characteristics listed on the previous page, we can perform some considerations. First, to obtain a precise velocity prediction inside clusters, q_l^v should be low;

to obtain homogenous video content information inside clusters, q_l^a should also be low; to obtain a sparse positional extension inside clusters, q_l^p should be high.

We suppose that the $\mathscr{L}$ clustering levels are such that, at the l-th level, the q^p, q^a, and q^v have reached the regime and they do not significantly decrease any further. We normalize q^p, q^a, and q^v and set a threshold from the regime for each of them. For example, the same threshold Th_r can be set for all three cases. The clustering level at which each of the three quality metrics goes below the threshold is found. The number of clusters corresponding to these three levels is averaged. We select as the optimum level the closest one to this average, among the $\mathscr{L}$ available levels.

A remaining problem is choosing the threshold Th_r. However, the method that we propose aims to obtain an approximate clustering level, one that does not lead to under-clustering or excessive over-clustering. A range of threshold values should, therefore, lead to acceptable results. We analyze these values in the results section.

To further explain, let us suppose to fix the threshold to 0.25. This means that the distance between the regime and the threshold is 25% of the distance between the highest value of the quality metric (typically the value for the first clustering level). Therefore, the threshold is set so to favor a high velocity and video latent state covariance, as desired.

8.6 Experimental Results

8.6.1 Employed evaluation datasets

We evaluate our method on the following datasets: iCab ES_1 compared with PM_1 (see Section 4.2.1), Egocart (see Section 4.2.3), FM and LAM drone scenarios (see Section 4.1.4), and CarlaABN (see Section 4.3). In the iCab, Egocart, and CarlaABN datasets, a terrestrial vehicle moves on a 2D space. In contrast, in the FM and LAM scenarios, a drone flies in a 3D space.

In the iCab dataset, we employ the PM_1 perimeter monitoring scenario for training and the ES_1 emergency stop scenario for testing. These two scenarios take place in the same courtyard, during the same season but at different times of day, which leads to a zone being in the shadows during training, but not during testing. A difficulty of the dataset for performing the localization is the symmetry of the courtyard on the four sides (see the first four images in Fig. 8.6). The training data are divided into $4,500$ for actual training and $1,598$ for validation.

In the Egocart dataset, a shopping cart moves in an empty retail store. We divide the $13,360$ training images into $10,259$ for training and $3,101$ for validation. Also in this dataset,

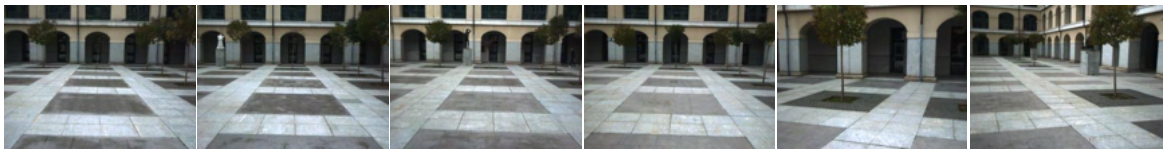


Fig. 8.6 Example of images in the iCab dataset. Observe the similarity between the first four images. The vehicle in those four cases is located on the four opposite sides of the courtyard.

positions far away from each other can display high visual similarity, due to the aisles of the supermarket having similar structure; however, the symmetry is not as high as in the iCab dataset. In contrast, the Egocart dataset has other difficulties compared to the iCab, FM, and LAM datasets. Firstly, its visual complexity is higher. Secondly, whereas the agent in the iCab, FM, and LAM datasets moves along a single path, the cart in the Egocart dataset turns at many intersections between the aisles and takes many alternative routes in the store.

We provide the reader with some summary information regarding the CarlaABN, FM, and LAM datasets as well. Further information can be retrieved in the respective dataset sections.

CarlaABN is a simulated dataset obtained with the Carla simulator. In the normal scenario, a vehicle moves around an empty city. Three abnormal scenarios are captured: a sudden deceleration (SD), an Out-Of-Street motion (OOS), and an Abnormal Turn at an Intersection (ATI).

The FM and LAM scenarios belong to a drone indoor dataset extracted in the context of this thesis. In the FM scenario, the drone performs a frontal motion. In the abnormal case, pedestrians move either on the side of the drone or in front of it. When pedestrians appear in front of the drone, the drone stops to let them pass and then continues its motion, similarly to the iCab ES_1 scenario. In the LAM scenario, the drone performs a lateral motion. In the abnormal case, some static abnormalities appear in the scene, constituted by squared boards of black or grey color. The FM scenario does not present significant problems of visual repetitions. Conversely, in the LAM case, certain instances of visual repetition occur between frames captured at considerable distances from each other.

In this chapter, compared to the previous ones, we add the simultaneous localization and use anomalies to improve it. As this is the main innovation, results are mainly related to these two capabilities and focus less on mere anomaly detection. All real-world datasets (iCab, Egocart, FM, and LAM) are adopted to test the localization capability of the method. The CarlaABN, instead, is employed to analyze the explainability of the anomalies. Notably, the Carla simulator allowed us to extract precise scenarios displaying anomalies that are

simple, known, and spatially/temporally contained, all characteristics that facilitate the study of explainability.

The localization results of the proposed method over the Egocart dataset and the memory requirement of the trained model are compared with the methods evaluated in [207]. Some of the methods in [207] are tested for comparisons on the Icab and drone datasets too.

An analysis of how the method's results change depending on the tracking parameters is provided.

Furthermore, we attempt to answer some important questions, such as: "how does the difference between the FV and SV affect the training and testing?", "how does the precision of the clusters change in a 2D vs. a 3D dataset?", "how do visual repetitions in the dataset influence the performance of the method?". While answering these questions, several characteristics of the datasets are explored, such as visual complexity, covered area, variety of motion types and viewpoints, and visual repetitions.

8.6.2 Implementation details

In this section, the implementation details of the method are provided: firstly, the optimization steps in the training procedure are described, then the details of the NN models for the three datasets are reported.

Training optimization

The training proceeds in different separately optimizable steps. For the clustering optimization, see Section 8.5. To stop the VAE training, a traditional division of the data between training and validation is used.

We additionally have to optimize the set of parameters $[N_{th,l}, N_{th,h}, m_v, P_w, W_h, W_l]$ and the anomaly thresholds of the CMJPF. We proceed in two phases.

First, we test on the validation data - without particles restarting - using a grid search over $N_{th,l}$, $N_{th,h}$, and m_v; we select the combination providing the best results. If we decide not to use the restarting of particles based on the anomaly levels, we can stop at this point. Otherwise, we use the combination providing the best results to perform testing on the training data.

Then, we calculate the mean and standard deviation of the anomalies over the training data and define a set of thresholds of the following type:

$$T^{(i,j)} = abn_m^{(j)} + \beta^{(i)} abn_{std}^{(j)}, \tag{8.13}$$

Table I Details about the models trained for each dataset.

	Icab	Egocart	Carla	Drone (FM)	Drone (LAM)
$[x_t]$	64x64	192x108	108x72	152x84	152x84
kernels	$(3,3)$	$(3,3)$	$(3,3)$	$(3,3)$	$(3,3)$
channels	$32,64,128$	$16,32,64,64,128$	$16,32,32,64,64$	$16,32,32,64,64$	$16,32,32,64,64$
$[a_t]$	32	128	16	16	16
$[z_t]$	8	20	8	8	8
K	45	249	87	79	75

where $T^{(i,j)}$ is a threshold i for anomaly j; $abn_m^{(j)}$ and $abn_{std}^{(j)}$ are the mean and standard deviation of anomaly j, and $\beta^{(i)}$ is a scalar associated to threshold i. Consequently, we test on the validation data - this time adopting particles restarting - and perform a grid search over P_w, W_h, W_l, and $\beta^{(i)}$. The combination providing the best result is used at testing. The grid search can be performed in two ways: either by allowing a different β for the different anomalies or by forcing all anomalies to use the same β. In the first case, compared to the second one, the search time must be multiplied by the number of anomaly types. In the following sections, when performing the grid search for the anomaly thresholds, we always employ the same β for all anomalies, to spare computational time. The other solution, however, could potentially lead to better results.

In the following sections, we perform both tracking parameters optimization and anomaly thresholds optimization for the Egocart and CarlaABN datasets. Instead, for the other datasets, unless explicitly specified, we execute only the parameters optimization. No particles restarting is performed. The reason behind this choice is that, in the Egocart and CarlaABN datasets, the vehicle turns at several intersections and follows different paths. In contrast, in the other datasets, the vehicle follows the same path, even if it travels at diverse speeds, and stops at different points. Therefore, in the former case, it is more likely that the localization might deviate on the wrong path, and the method needs to correct it.

In Section 8.6.7, an analysis of the impact of the choice of parameters is provided.

Models details

Table I shows the details of the models used for the three datasets. For the Icab case, images are reduced to a resolution of 64x64. We use a VAE with 3 convolutional layers with kernel size equal to 3 and channel dimensions $(32,64,128)$, and one Fully Connected (FC) layer. Dimension of a_t is 32, and of z_t is 8. The number of clusters K is 45. Batch size is 16 with a step decay rate of 0.99 and decay steps equal to 20. Gradient cropping is used. Weight decay is set to 0.001. Training is performed for 35 epochs on the VAE alone, 300 on the matrices

Table II Comparison between the FV and SV of the proposed method. The table shows the localization error in meters.

Dataset	FV		SV	
	Mean Err. (m)	Median Err. (m)	Mean Err. (m)	Median Err. (m)
Egocart ($\hat{x}^o_{t,n}$)	2.34	0.86	1.83	1.07
Egocart ($x^o_{t/t,n}$, final)	2.08	0.76	1.65	0.96
iCab ($\hat{x}^o_{t,n}$)	1.15	0.79	1.17	0.84
iCab ($x^o_{t/t,n}$, final)	1.06	0.78	0.98	0.75
Drone FM ($\hat{x}^o_{t,n}$)	0.21	0.13	0.23	0.14
Drone FM ($x^o_{t/t,n}$, final)	0.21	0.13	0.23	0.14
Drone LAM ($\hat{x}^o_{t,n}$)	1.45	0.57	0.88	0.39
Drone LAM ($x^o_{t/t,n}$, final)	1.45	0.56	0.87	0.38

alone, and 700 on all parameters. For the Egocart and Carla cases, see Table I again. All absent parameters are the same that are used in the Icab case.

The model was trained in Python, using the PyTorch library.

8.6.3 Localization results

First, we study the localization results: we evaluate the FV and SV of the method, we compare our approach with other methods from the state of the art, and we consider one dataset to analyze the clustering optimization and how the number of clusters affects the localization performance.

Comparison between the FV and SV of the proposed approach

Table II compares the localization results of the FV and SV of the approach on the iCab, Egocart, FM, and LAM datasets. The mean and median localization errors in meters are shown. Results refer to particles surviving resampling until the end of the trajectory.

The first row related to each dataset reports the localization error from the $\hat{x}^o_{t,n}$ estimations (i.e., directly from the video DBN prediction) and from the $x^o_{t/t,n}$ estimations (i.e., after performing the odometry 'update' phase). Therefore, the second row involves the use of the link $P(\tilde{z}^o_{t+1}|\tilde{z}^o_t)$ as well, whereas the first row doesn't. The second row is the final result of the method (and, as can be observed, displays better performance in all four cases).

The two columns on the left of the table refer to the FV, and the two columns on the right refer to the SV. The SV tends to display better results, especially from the point of view of

the mean error. A significant improvement can be observed on the Egocart (43 cm on the mean error, whereas the median error increases by 20 cm) and drone LAM cases (58 cm on the mean error and 18 cm on the median error). A minor improvement is present for the iCab dataset (14 cm on the mean error, whereas the median error increases by 6 cm). In the FM case, instead, the SV provides better results, but the difference is minimal (2 cm for the mean error and 1 cm for the median error).

Comparisons between the proposed approach and other methods

In this section, we compare the proposed method against state-of-the-art ones. Tables III, IV, V, and VI show the localization errors obtained on the Egocart, Icab, FM, and LAM datasets, respectively. We perform these comparisons to evaluate if the method can reach a localization accuracy around the state of the art. However, we must also note that the proposed method does not solely perform localization. Instead, it is inserted in a wider theoretical framework for self-aware autonomous systems, which includes anomaly detection and explainability. Conversely, the methods used for comparison were developed only for localization.

The last two rows of each table (marked in blue) report the localization error of our method, for the FV and SV, respectively. In the next paragraphs, when referencing our method, unless expressly stated otherwise, we refer to the SV.

Table III shows the results obtained in [207] on the Egocart dataset through several methods, alongside three methods that we added. The first section of the table refers to Image Retrieval (IR) methods, and the second one to Regression (REG) methods. We omit the classification-based method reported in [207] as we assume to be working on scenarios where the environment zones are not labeled. In contrast, we have added three methods: IR-VAE, IR-TC-VAE, and REG-ENC. In IR-VAE, first, a VAE is trained to reconstruct the training images. A database of the latent states obtained with the VAE for the training data is built. During testing, each image x_t is given as input to the VAE, extracting the latent state a_t. The closest latent state in the training database is found; x_t is finally assigned the position of this latent state. This method is, therefore, similar to methods such as IR-VGG16, and IR-IV3. The difference is that the latent state is extracted from a VAE built solely on the training dataset, instead of using a model pre-trained on ImageNet, a classification dataset with more than 1.28 million images. The VAE is exactly the same that is employed in our approach. The IR-TC-VAE case adds a temporal constraint, such that retrieval can be performed only on images of the training set that do not have a position further away of 4 meters from the position estimated for the previous testing time instant. This case is similar to IR-TC-VGG16, but, again, with the VAE instead of the pre-trained model. Finally, REG-ENC is a regression method that trains a CNN to directly predict the position from the

Table III Localization error on Egocart (in meters). The final result of the proposed method (either with FV or SV) is highlighted in blue.

Method	Mean Err. (m)	Median Err. (m)
IR-FISHER	1.63	0.31
IR-VGG16	0.72	0.28
IR-IV3	0.73	0.28
IR-TR-VGG16	0.55	0.28
IR-TR-IV3	0.69	0.32
IR-PNET-VGG16	2.17	1.38
IR-PNET-IV3	0.70	0.39
IR-TC-VGG16	0.52	0.28
IR-TR-TC-VGG16	0.44	0.29
IR-VAE	1.60	0.32
IR-TC-VAE	3.61	0.39
REG-PNET-RGB-VGG16	1.62	1.23
REG-PNET-RGB-IV3	0.57	0.39
REG-PNET-TR-RGB-IV3	0.56	0.42
REG-PNET-TR-TS-RGB-IV3	0.55	0.36
REG-PNET-RGB-POS-IV3	0.42	0.29
REG-SVR-PNET-RGB-VGG16	1.96	1.54
REG-SVR-TR-RGB-VGG16	1.46	0.90
REG-ENC	8.59	7.66
Proposed (D_B) - $x^o_{t/t,n}$ (FV)	2.08	0.76
Proposed (D_B) - $x^o_{t/t,n}$ (SV)	1.65	0.96

image. The used CNN has a structure similar to the VAE encoder, with the difference that a ReLU activation, a dropout layer ($p = 0.5$), and an FC layer are added after the bottleneck feature a_t (the variance is, instead, not learned). The FC layer gives the position as output. For this reason, we denominate this method REG-ENC, as the used CNN is the same as the encoder of the VAE, with the addition of three layers. Furthermore, the same training choices (e.g., learning rate, optimizer) are employed as for the VAE. This method is similar to REG-PNET-RGB-POS-IV3. The reason for adding these three methods is to have a fairer comparison. All the methods employed in [207] start from models pre-trained on ImageNet, whereas our method does not. Potentially, also our approach could start from a pre-trained model. However, this is not completely coherent with our framework: pre-trained models are trained through classification, which is supervised. Instead, in our framework, we desire to learn concepts in an unsupervised and continual way, leveraging anomalies.

Table IV Localization error on iCab (in meters). Results not in parenthesis refer to building the database using all 6,558 frames; results in parenthesis refer to using only the 4,500 training ones. The final result of the proposed method is highlighted in blue.

Method	Mean Err. (m)	Median Err. (m)
IR-IV3	1.28 (1.48)	0.61 (0.65)
IR-TC-IV3	0.72 (0.77)	0.60 (0.64)
IR-VAE	23.00 (23.00)	23.00 (23.00)
IR-TC-VAE	23.00 (23.00)	23.00 (23.00)
REG-PNET-RGB-POS-IV3	1.52	1.15
REG-ENC	23.88	22.78
Proposed - $x^o_{t/t,n}$ (FV)	1.06	0.78
Proposed - $x^o_{t/t,n}$ (SV)	0.98	0.75

Table V Localization error on Drone FM (in meters). Results not in parenthesis refer to building the database using all 20,771 frames; results in parenthesis refer to using only the 18,253 training ones. The final result of the proposed method is highlighted in blue.

Method	Mean Err. (m)	Median Err. (m)
IR-IV3	0.18 (0.18)	0.16 (0.16)
IR-TC-IV3	0.18 (0.18)	0.16 (0.16)
IR-VAE	0.20 (0.20)	0.16 (0.16)
IR-TC-VAE	0.20 (0.19)	0.16 (0.16)
REG-PNET-RGB-POS-IV3	0.24	0.20
REG-ENC	0.89	0.76
Proposed - $x^o_{t/t,n}$ (FV)	0.21	0.13
Proposed - $x^o_{t/t,n}$ (SV)	0.23	0.14

For comparison on the Icab, FM, and LAM datasets, we selected three of the methods in [207] to have one IR method (IR-IV3), one IR method that considers the sequencing of images (IR-TC-IV3), and one REG method (REG-PNET-RGB-POS-IV3). IR-IV3 executes IR based on features from Inception-v3 [212] pre-trained on ImageNet; IR-TC-IV3 adds the aforementioned temporal constraint. The temporal constraint is fixed to 4 meters for the iCab dataset, and to 0.5 meters for the FM and LAM datasets, which cover a smaller area. REG-PNET-RGB-POS-IV3 performs regression using a PoseNet [119] trained starting from an Inception-v3 backbone pre-trained on ImageNet. An FC layer of 2048 units is added to the backbone, followed by a ReLU activation, a dropout layer ($p = 0.5$), and another FC

Table VI Localization error on Drone LAM (in meters). Results not in parenthesis refer to building the database using all 19,104 frames; results in parenthesis refer to using only the 17,368 training ones. The final result of the proposed method is highlighted in blue.

Method	Mean Err. (m)	Median Err. (m)
IR-IV3	0.32 (0.32)	0.20 (0.20)
IR-TC-IV3	0.33 (0.34)	0.20 (0.20)
IR-VAE	0.47 (0.49)	0.25 (0.28)
IR-TC-VAE	0.86 (0.89)	0.33 (0.38)
REG-PNET-RGB-POS-IV3	0.74	0.71
REG-ENC	1.83	1.14
Proposed - $x^o_{t/t,n}$ (FV)	1.45	0.57
Proposed - $x^o_{t/t,n}$ (SV)	0.87	0.38

layer giving the position as output. The training was conducted with the Adam optimizer for 50 epochs, with batch size 16, and a learning rate of 10^{-5}. The number of epochs, instead of being fixed, is chosen based on when the loss on the validation data starts increasing. Note that, instead, in [207] no division between training and validation was used.

Table III shows that, on the Egocart dataset, the proposed method performs better than IR-PNET-VGG16, REG-SVR-PNET-RGB-VGG16 for the mean and median error, and better than REG-PNET-RGB-VGG16 for the median error. However, it achieves worse results than the other compared methods that use pre-trained models, as it does not leverage the pre-training on ImageNet. On the other hand, the model performs significantly better than REG-ENC, better than IR-TC-VAE, and worse than IR-VAE.

In the iCab case (Table IV), due to the symmetry of the environment, better results are obtained with those methods that also enforce temporal constraints/reasoning (in particular, IR-TC-IV3 and the proposed method). The proposed method performs much better than the three methods (IR-VAE, IR-TC-VAE, and REC-ENC) that do not employ a pre-trained model and use the same VAE (or encoder). The reason behind the failure of these models resides in the high visual symmetry of the dataset: the VAE features corresponding to opposite sides of the courtyard are very similar. However, through the Particle Filter, it is possible to consider diverse hypotheses in parallel for several time instants, and then select which ones to keep.

In the FM scenario (Table V), similar results are obtained with the different considered methods. The only exception is the REG-ENC approach, which localizes worse than all the other ones. Our method is coherent with the other ones, performing slightly worse on the mean error and slightly better on the median error.

Finally, Table VI displays the results in the LAM scenario. In this case, our approach performs worse than the methods with pre-trained models and IR-VAE, better than REG-ENC, and very similarly to IR-TC-VAE.

Therefore, we can observe that the proposed method achieves results that are sometimes worse, sometimes better, and sometimes equivalent to other state-of-the-art methods. Out of the six methods presented in all tables, the IR-TC pre-trained methods (i.e., IR-TC-IV3 and IR-TC-VGG16) always outperform the proposed approach, except for the FM scenario, where the results are similar. The REG-ENC consistently performs worse than our approach. It is, however, worth noting that IR-TC-IV3 and IR-TC-VGG16 employ pre-trained models, whereas REG-ENC, like our approach, does not.

In summary, the proposed method performs on a similar range with other state-of-the-art VBL methods. It provides, however, further features compared to them:

- a structure coherent with a self-awareness framework;
- the definition of coupled odometry and visual generative maps allowing the extraction of anomalies for explainability and potential continual learning.

Localization was also attempted using ORB-SLAM3 [30] on the iCab and Egocart datasets. We remind that a single image is used, no depth data, and no Inertial Measurement Unit. On both datasets, however, the localization lost track in correspondence with curves or sudden motions. It is relevant to note that the frame rate of the Egocart dataset is quite low (3 fps). We also remind that, compared to our method, SLAM supposes that the environment is not known.

Comparisons with different clustering levels

In this section, we analyze the localization results obtained with different clustering levels. Table VII shows the mean and median localization error obtained with 8, 27, 45, 116, and 150 clusters. These cases correspond to a threshold Th_r (see Section 8.5) of around 0.75, 0.45, 0.35, 0.10, and 0.075, respectively. The threshold range 0.10-0.35 is a good trade-off between accurate velocity prediction, visual content homogeneity, and high spatial extension, favoring the first two elements. We can observe that, in this range, the localization error maintains similar values.

Using a much higher threshold leads to under-clustering and to a severe degradation of performance, as can be observed in the case with a 0.75 threshold. This is a bad choice from the localization performance point of view, as it corresponds to giving much more priority to having few big clusters than to having accurate velocity predictions and visual content homogeneity inside each cluster.

Table VII Localization error on iCab (in meters), with the SV, varying the number of clusters. The last column defines the threshold Th_r corresponding to the clustering level.

Number of clusters	Mean Err. (m)	Median Err. (m)	Th_r
8	4.98	2.09	~ 0.75
27	1.13	0.83	~ 0.45
45	0.98	0.75	~ 0.35
116	1.10	0.74	~ 0.10
150	1.05	0.68	~ 0.075

Using a lower threshold, performance remains stable, with the disadvantage of an increase in memory requirements and computational complexity, caused by the need to store a larger vocabulary and to perform comparisons against a greater amount of clusters during tracking.

It is also worth noting that the number of clusters that allow to obtain good performance change depending on the used method and on the objective. In Chapter 5 and 7, we observed that 6 and 8 clusters, respectively, were sufficient to obtain good anomaly detection results. However, for localization with the proposed piece-wise linear method, 8 clusters are not enough to reach a satisfactory performance. Moreover, it is also worth noting that in Chapter 5, where 6 clusters were enough, neural networks were employed instead of linear models.

8.6.4 Quantitative and qualitative anomaly analysis

In this section, the anomalies are analyzed from a quantitative and qualitative point of view, on the Egocart, iCab, and CarlaABN datasets. We focus especially on the explainability of the anomalies. The results in this section are all extracted with the FV of the method.

Egocart

Fig. 8.7a presents the final errors associated with the first Egocart testing trajectory under two cases: without activation of the restart mechanism for a subset of particles, and with activation ($P_{par} = 25\%$). The restarts improve the results from a mean error of 4.40 m to one of 1.63 m. Green dotted bars show the times in which restarts are performed. Fig 8.7b displays some of the detected anomalies, explaining what caused the restarting choice. We display some examples reconnecting to Fig. 8.8. At the point labeled as “1” in the figures, the restart is due to a cluster-level anomaly (i.e., different sequencing compared to training). For point “2te”, the restart is prompted by a high reconstruction error. In Fig. 8.8, the pixel-wise image reconstruction error for point “2te” can be observed and compared with the same

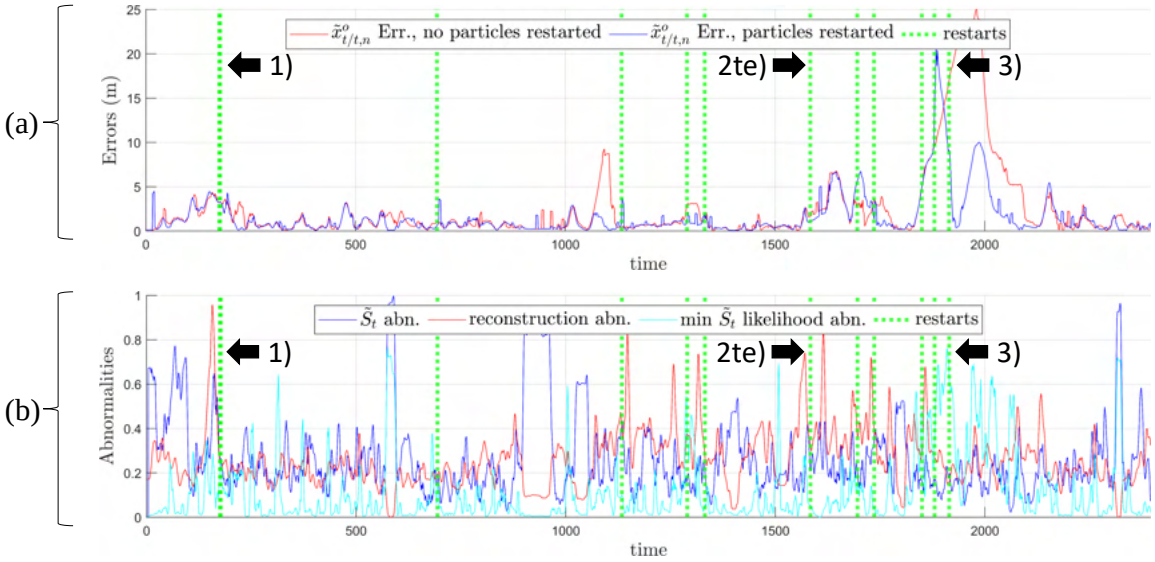


Fig. 8.7 (a): errors for the first Egocart trajectory, when particles re-initialization isn't adopted (red) and when it is (blue). (b): examples of anomalies. Restart points 1), 2te) and 3) are the ones displayed in Fig. 8.8 too. The $\tilde{S}_t$ abnormality corresponds to the KLDA. © 2022 IEEE.

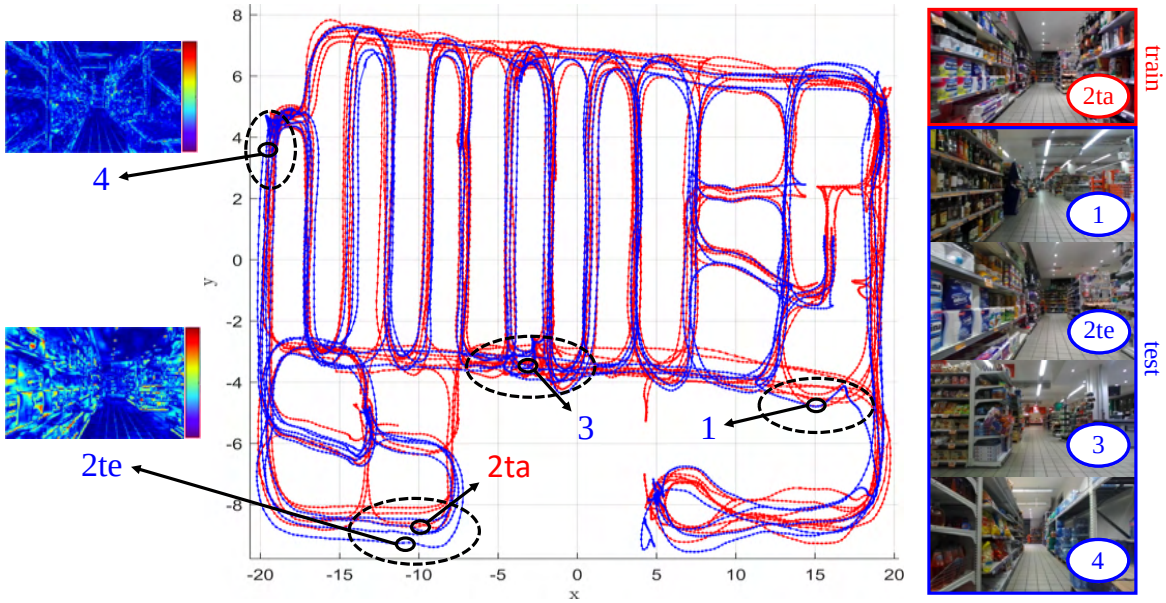


Fig. 8.8 Egocart positions for training (red) and testing (blue) data. One training point ("2ta") and four testing points ("1", "2", "2te", and "4") are highlighted, and the corresponding camera frames are shown on the right. The direct image reconstruction error of points "2te" and "4" is displayed through a heatmap. The red color signifies a high error, whereas the blue color represents a low error. © 2022 IEEE.

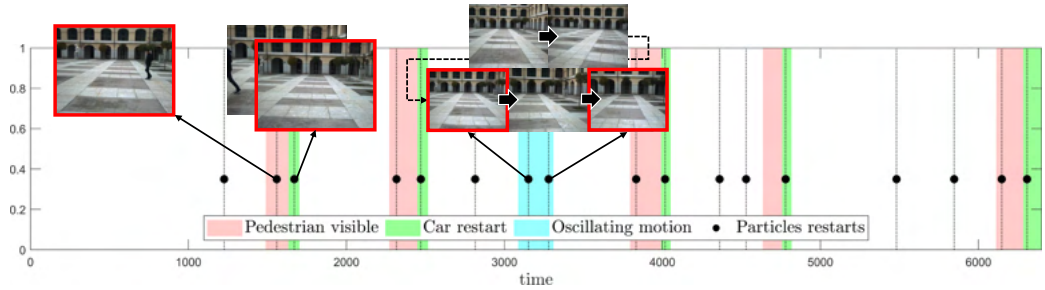


Fig. 8.9 Anomaly events and particles restarts in the Icab case. The colored zones display the anomaly regions and their type. The black dots show the time instants at which particles of the PF were restarted. © 2022 IEEE.

type of error in a normal testing point ("4"). Indeed, we can note how the test point "2te" is outside of the traversed zone in training, with point "2ta" being the closest one from the train set. At point "3", the odometry points display significant entropy in the probability of belonging to the distribution of the clusters: see in Fig. 8.8 that this is a dense zone. Due to this uncertainty, some particles are reinitialized.

Icab

A ground truth was defined for the Icab dataset. We analyze how accurate the anomalies are and, consequently, how often particle restarts are fired for a good reason. In this example, therefore, we restart particles also in the iCab case. The restarting of particles corresponds to the detection of an abnormal event. Fig. 8.9 displays the color-coded anomalies across time: the visible pedestrian forcing the car to stop (red), the car restarting after the emergency stop (green), and an oscillating motion after a curve (cyan). Abnormal regions account for 21.69% of the trajectory. The black dots indicate instances when abnormal events surpassed the thresholds, resulting in particle restarts. Out of the 17 restarts, 11 corresponded to zones in the anomaly GT ($precision = 0.647$). The remaining restarts were mainly due to oscillating motions of a smaller amplitude than the one plotted in cyan. Out of the 11 abnormal events displayed in Fig. 8.9, 10 correspond to at least one anomaly over the threshold ($recall = 0.909$). The only false negative is the third "pedestrian visible" anomaly.

CarlaABN

We use qualitative results on the Carla dataset to display how each anomaly indicator can capture distinct types of abnormal situations.

Fig. 8.10 shows some of the anomalies presented in Section 8.2.4 on three abnormal scenarios manually extracted with the simulator. Fig. 8.11 displays the trajectories and some

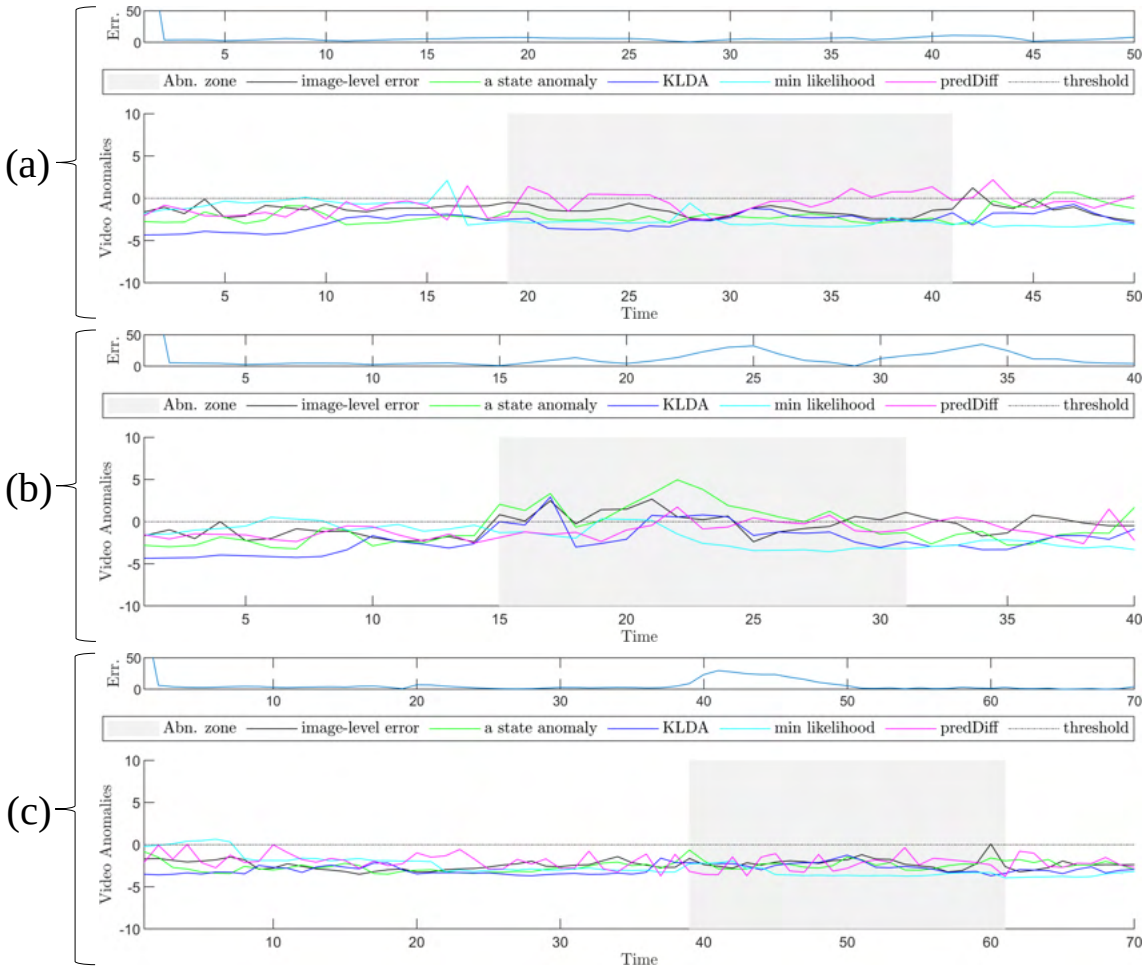


Fig. 8.10 Anomalies extracted on the three CarlaABN testing trajectories. The grey areas are the abnormal zones: (a) sudden deceleration (SD); (b) Out-Of-Street Motion (OOD); (c) Abnormal Turn at Intersection (ATI). One frame for each scenario is shown in Fig. 8.11.

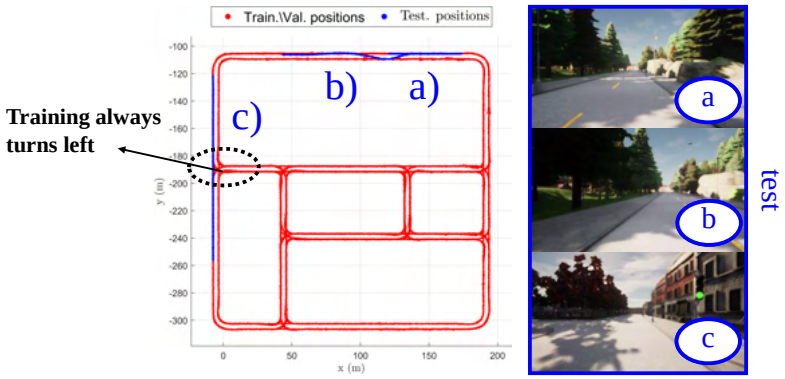


Fig. 8.11 CarlaABN positions for training (red) and testing (blue) data. On the left, one camera frame is displayed for each abnormal scenario.

Table VIII Memory requirements for each dataset.

	Icab	Egocart	Carla	Drone (FM)	Drone (LAM)
Memory (MB)	3.24(3.13)	20.7(6.5)	0.85(0.79)	1(0.94)	1(0.97)

image examples for the three anomalies. The (a), (b), and (c) letters refer to the same testing trajectories in the two figures. The simulator enables us to obtain abnormal data in which a single or a limited number of causes are isolated. In Fig. 8.10a, the vehicle abruptly stops and immediately restarts (see the area in gray). Note the increase in the prediction difference indicator (corresponding to $diff_{\hat{z}^o}$), which shows that the odometry and video models are performing two different predictions. The odometry relies more on its prediction, which supposes that the car continues to move forward at high speed. In Fig. 8.10b, the vehicle moves into the wrong lane and obtains a lateral view of a forest that was not observed during training. As expected, the a_t state anomaly (i.e., $diff_a$) and the z_t state anomaly (i.e., $diff_z$) exceed the threshold due to the newly observed images. However, the prediction difference indicator (i.e., $diff_{\hat{z}^o}$) is high, too, due to the abnormal motion. Finally, in Fig. 8.10c, the vehicle proceeds straight at an intersection, where, in training, it always turned left (see Fig. 8.11). The algorithm fails to detect the anomaly. However, it is worth noting that the gray area exhibits higher values of the KLDA compared to the rest. The failure can be attributed to setting all thresholds with the same β value, whereas a lower threshold for KLDA may have been more beneficial.

8.6.5 Algorithm memory requirements and speed

We analyze the algorithm's memory requirements and computational time, considering the Icab and Egocart models. The algorithm is implemented in Python using the Pytorch library. The experiments were performed using an Intel® CoreTM i7K 8700K Processor and a dual NVIDIA® GeForce® GTX 1080 Ti with 11 GB RAM GDDR5X each.

Table VIII shows the required memory to store the overall model. In the Icab case, 3.24 MB are necessary, of which 3.10 MB are for the VAE parameters and for the matrices A, B, C, D, and E.

The distance $d_{v,t}^{(\tilde{S})}$ (see Subsection 8.2.4) can be calculated in different ways. In the text, we proposed to use the Bhattacharya distance D_B between $M^{(\tilde{S},a)}$ and $Q^{(\tilde{S},a)}$. If, instead, the MSE between $M^{(\tilde{S},a)}$ and a_t is employed, it is not necessary to store the matrix $Q^{(\tilde{S},a)}$. The total memory requirement is reduced to 3.13 MB. The minor gain is due to having only 22 clusters and to the small dimension of the latent state z_t.

Table IX The last column on the right displays the algorithm's computational cost with varying settings. The changed settings are displayed from the second to the fourth column and are: the number of particles (N), the method to calculate the distance $d_{v,t}^{(\tilde{S})}$ between video states and video clusters, the adoption of the restart mechanism ("Restarts").

Dataset	N	$d_{v,t}^{(\tilde{S})}$	**Restarts**	**time/img (ms)**
Icab	5	D_B	Yes	25
	25	D_B	Yes	65
	50	D_B	Yes	115
	50	MSE	Yes	109
	50	MSE	No	108
	50	D_B	No	114
Egocart	150	D_B	Yes	640
	125	D_B	Yes	550
	125	MSE	Yes	504
	125	MSE	No	503
	125	D_B	No	549

In the Egocart case, the memory required for storing the overall model is 20.7 MB, and can be reduced to 6.5 MB, resulting in a substantial gain, as 249 clusters are used. The localization results are similar in the two cases: a mean error of 2.08 and median error of 0.76 with D_b, a mean error of 1.98, and median error of 0.97 for the MSE case. The comparison is performed with the FV of the approach. The model is smaller than the ones based on Inception V3 [212] (93 MB, 129 MB, 258 MB) and VGG-16 [202] (512 MB) used in [207]. No additional training data information needs to be stored.

Table IX shows the computational cost of the algorithm, for both the Icab and Egocart model, varying the number of particles, using either D_B or MSE for calculating $d_{v,t}^{(\tilde{S})}$, and adopting particles restarts or not adopting it. First, we observe that using MSE instead of D_B reduces the computational cost of 6 ms in the Icab case with 50 particles (N) and of 46 ms in the Egocart case with 125 particles. The computational cost of particle resampling, instead, is negligible: approximately $1ms$ in both cases. In the Icab case, there is a fixed cost of $15ms$, plus $2ms$ for each added particle; in the Egocart case, these two costs are $100ms$ and $3.6ms$, respectively.

8.6.6 Attention analysis

We analyze the image information on which the network concentrates during the processes of localization and prediction. In this way, we investigate the meaning of the z_t state. There are

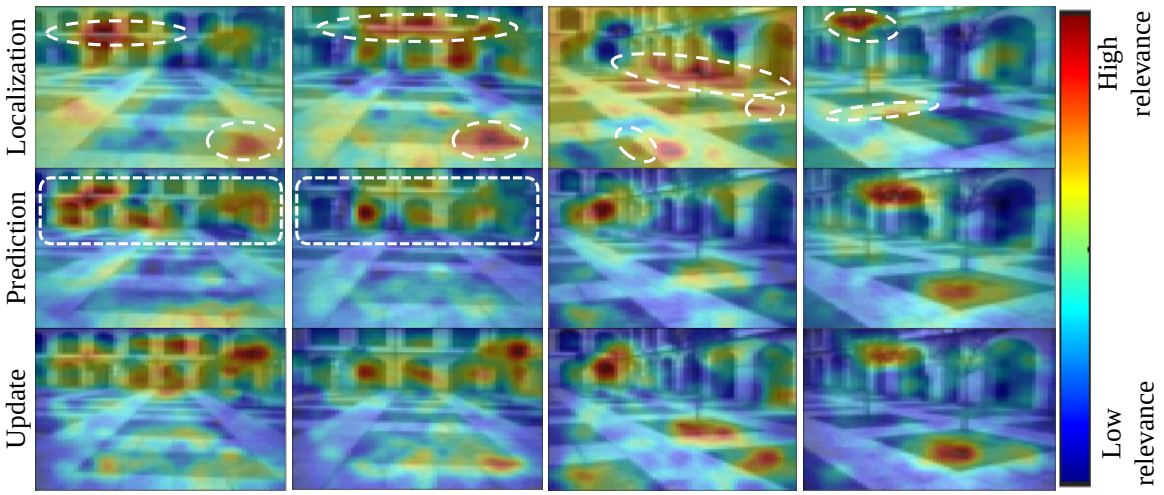


Fig. 8.12 Attention analysis over the Icab dataset: on which image features does the network concentrate its attention when performing localization, prediction, and update?

two ways to visualize where NNs are concentrating their attention: *A)* methods that change the model input to find what areas, when perturbed, generate the most significant changes in the output [240, 68] ; *B)* methods that visualize how the network internally responds to an input [199, 10]. Most of these methods are thought for image classification.

As in our model the convolutional layers are followed by FC ones and by a set of matrices, we do not have a 2D feature space as in most cases of the second type. Consequently, we chose a simple method of the first type. We loop over the training data points; for each data point x_t, filtering is performed over $[x_{t-\eta}, x_{t-1}]$, with η being a chosen time window. Then, the update at t is performed in two ways: *i)* using the actual observation x_t; *ii)* sliding a set of occlusion windows over x_t. For each occlusion window, we calculate the difference in the update $z_{t|t}$ and in the localization prediction obtained with i) and ii). The predictions $z_{t+1|t}$ calculated through the A matrices are compared too. The obtained differences are used to build three attention images for each data point.

Fig. 8.12 shows the obtained explanation maps on parts of the Icab dataset. A color closer to red means that a higher difference was obtained between the two cases, i.e., i) and ii). The rows are related to localization, prediction, and update. For localization, the network seems to keep information regarding some elements such as corners on the floor pattern, horizontal lines of the colonnade, and trees (circled in white on the first row of Fig. 8.12). For the prediction, the model seemed to concentrate on the colonnade. This could be explained by the fact that this element of the image is more easily encoded in a linear way (the colonnade gets bigger as the car approaches) compared to more complicated patterns,

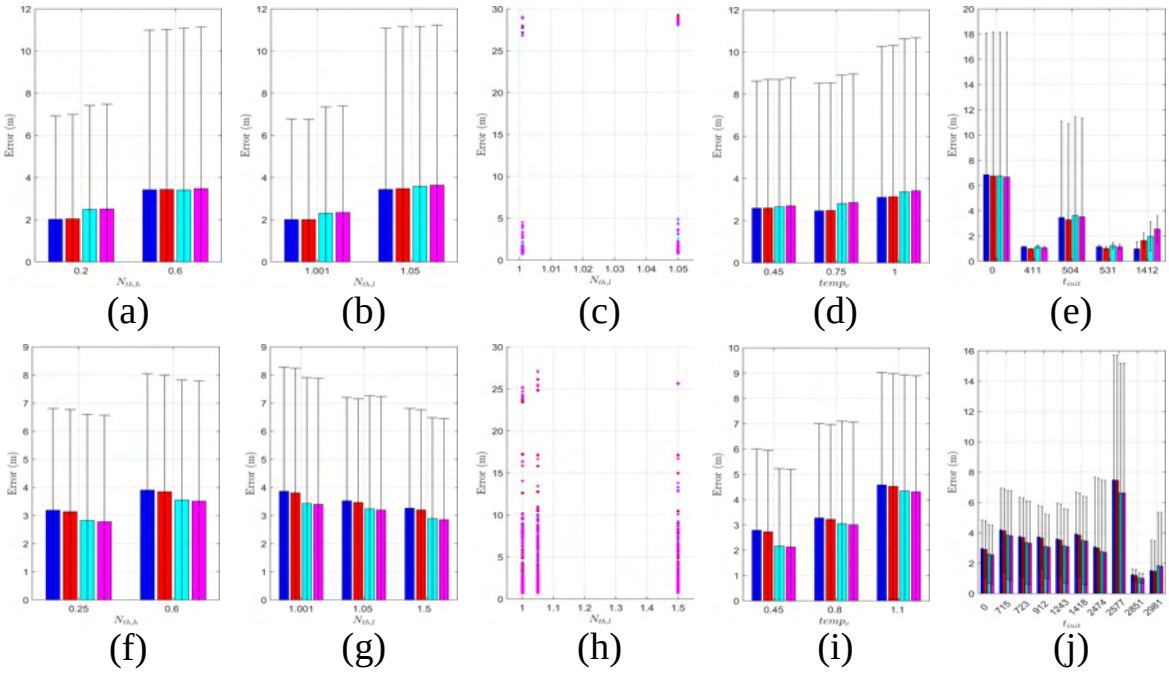


Fig. 8.13 Localization error on validation dataset choosing different values of the parameters, for Icab (a-e) and Egocart (f-j) cases. (a,f): $N_{th,h}$, (b,g) and (c,h): $N_{th,l}$, (d,i): m_v. (e,j) refer to different starting points of tracking.

such as the horizontal lines of the floor. A combination of the localization and update spots is highlighted for the update.

8.6.7 Parameters choice analysis

In this section, we analyze how the parameters for the CMJPF influence the results of the validation set.

Fig. 8.13 displays the localization results obtained on the validation set of the iCab and Egocart datasets, with different parameters. Subfigures (a) to (d) and (f) to (i) consider variations of $N_{th,l}$, $N_{th,h}$, and m_v. The varied parameter is defined on the x-axis of the plots. The mean final distance with each parameter is displayed as the high of the colored columns and the standard deviation as an error bar. The error bar was thresholded to zero on the lower part when necessary. Each color represents a different method of calculating the localization error. Blue and red correspond to the localization from the weighted average of the particles using $\hat{x}^o_{t,n}$ and $x^o_{t|t,n}$, respectively; cyan and magenta, instead, refer to the use of particles surviving resampling, as in Table III and IV. The first row is for Icab, and the second is for Egocart.

Notice how, in the Icab case, where no intersections are present, using the particles with their weights gives the best result. Conversely, in a case with many intersections, such as in the Egocart dataset, considering the estimation of the particles remaining after full resampling provides better results. At an intersection, two paths could have equally high weights initially, but the less likely one will be cut at the end.

Fig. 8.13b shows that better results are obtained with a lower $N_{th,l}$ on the Icab dataset, which reflects how the environment is structured: in a highly symmetrical courtyard, waiting longer before the first particles resampling allows us to pick the right side with a higher probability. Fig. 8.13c shows this same case displaying all the points with their respective error. Notice how cases with errors of almost 30 meters are found on the very top of the figure: the vehicle is predicted to be on the opposite side of the courtyard. It happens 2 times for $N_{th,l} = 1.01$ and 5 times for $N_{th,l} = 1.05$. In contrast, on the Egocart dataset, a higher $N_{th,l}$ provides better results, as can be observed in Fig. 8.13g. This is attributable to two characteristics of the Egocart dataset: it has a low frame rate and it has many turns that lead to the environment changing quickly. Consequently, resampling after a longer time also at the beginning can be detrimental because wrong path estimations need to be cut promptly. In addition, the Egocart dataset does not have the strong symmetry of the Icab dataset, which made a lower $N_{th,l}$ useful.

A lower temperature parameter m_v (Fig. 8.13d and 8.13i) has a similar impact to that of the resampling thresholds $N_{th,l}$ and $N_{th,h}$, as it distributes the particles on the possible hypothesis, which is beneficial in a case with many intersections as in the Egocart dataset. Fig. 8.13e and 8.13j are not related to a trainable parameter but to the different instants that were used as starting points. When evaluating the best choice of parameters, tracking is started from several starting points across the trajectories. This provides a better choice of the parameter $N_{th,l}$ if the training and validation trajectories always start from the same area but the testing trajectories might not.

This first testing over several parameters was conducted on a grid of 120 combinations for Icab and 540 for Egocart. Similar testing is conducted to choose the parameters related to the anomaly thresholds on a grid of 90 and 810 combinations.

8.6.8 How does the difference between the FV and SV affect the training and testing?

In this section and in the following ones, we tackle some important questions related to the proposed approach. The first question relates to the choice between the FV and the SV. In Section 8.4, we have already seen the advantages and disadvantages of the two versions:

the FV is more modular but provides less accurate localization, whereas the SV gives better localization results but fragments the training of the model. Section 8.6.3 demonstrated the accuracy difference between the two methods by testing on four datasets. In this section, we analyze in greater detail the difference between the two versions. We can subdivide the main question into four sub-questions.

How do the clustering quality metrics change in the two cases?

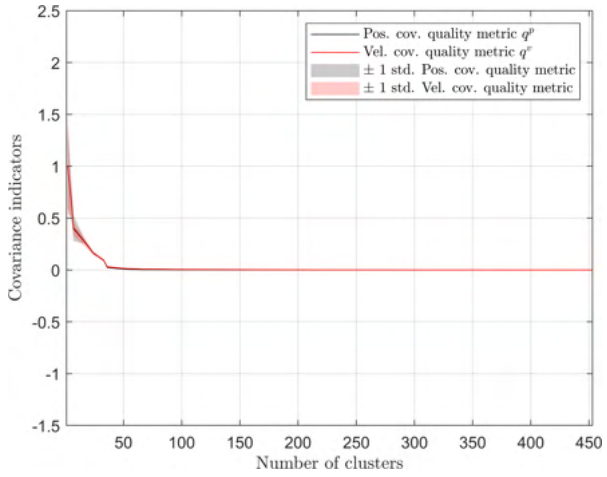
We have seen in Section 8.5 that the optimal clustering level can be selected using three quality metrics q^p, q^v, and q^a. Let us now examine how these quality metrics change when changing the number of clusters, taking the iCab and LAM scenarios as examples.

Fig. 8.14 is related to the iCab dataset: in (a) and (b), the evolution of the quality metrics depending on the number of clusters is displayed for the FV and SV, respectively. As the FV employs q^p and q^v for optimization, we only show them in Subfigure (a). In contrast, Subfigure (b), which is related to the SV, illustrates all three metrics. We also show the standard deviation of the metric across the different clusters of the clustering level. To facilitate comparison, Subfigure (c) joins the plots of the FV and SV (showing the q^v evolution over the FV too). The plots are normalized on the same scale. This normalization can be operated as the same odometry GSs and VAE latent states are used in the two versions, since the odometry information is the same, and the same VAE was employed. Observe how, in the SV, both the velocity and image covariance quality metrics tend to descend faster, meaning that fewer clusters are necessary to obtain similar levels of velocity accuracy and image content homogeneity. However, also note that, after a certain clustering level (~ 50 clusters), the velocity covariance metric of the SV always remains higher than the same metric for the FV, and decreases very slowly. The reason behind this behavior can be explained by the trade-off between the velocity accuracy and the image homogeneity. There might be very similar images with a certain level of difference in the velocity evolution that end up being clustered together. This concept will be discussed further in the next section.

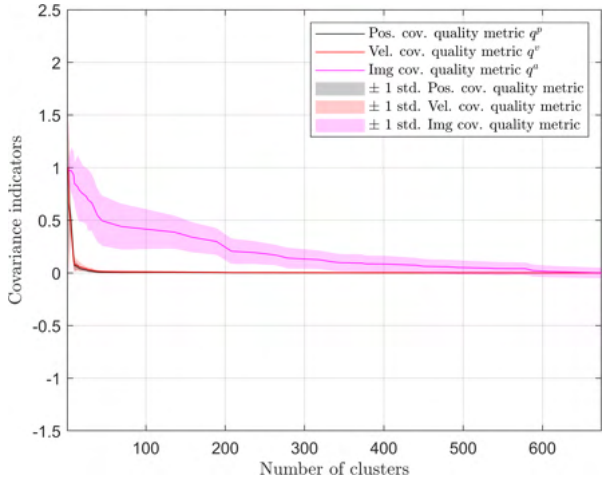
Fig. 8.15 displays the quality metrics for the drone LAM scenario. In this case too, the image covariance quality metric q^a descends much quicker in the SV of the approach. Conversely, the velocity covariance quality metric q^v decreases more slowly in the SV.

How do the clustering means and covariances change?

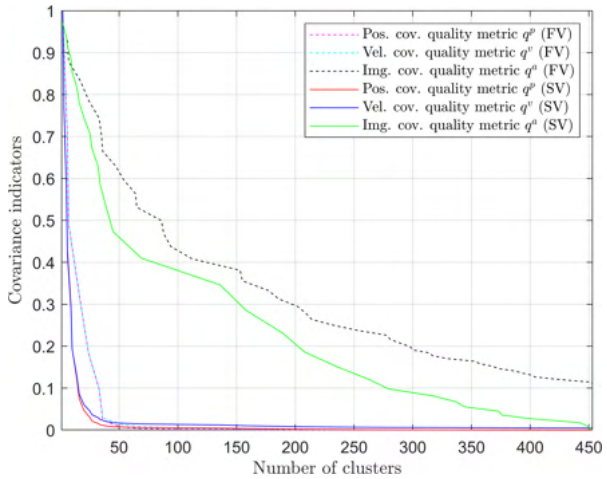
In this section, we take one example (the iCab) to show the variation of the image clustering on the odometry space. The performed comparison is only qualitative and is executed on the



(a)

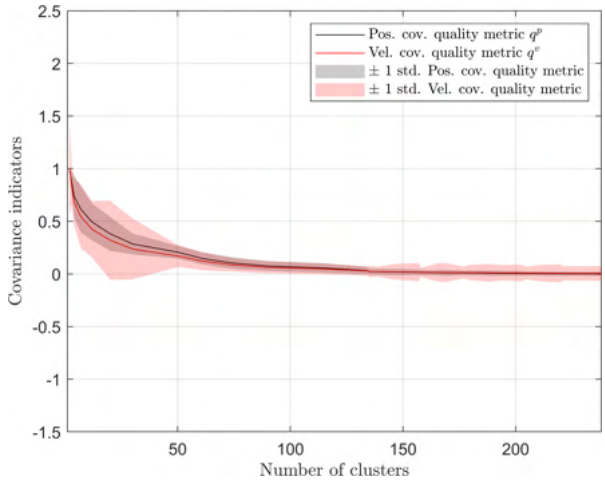


(b)

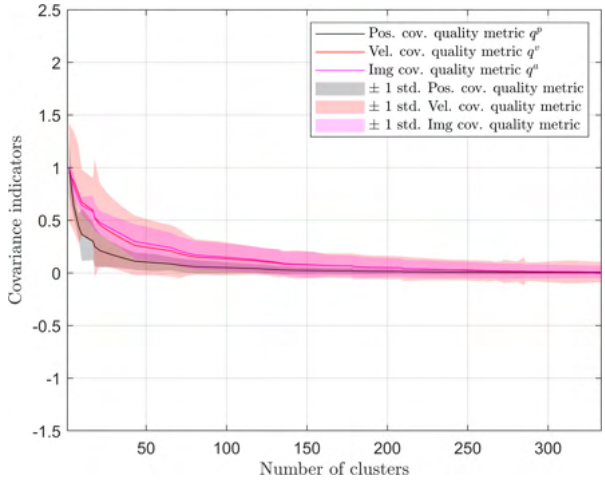


(c)

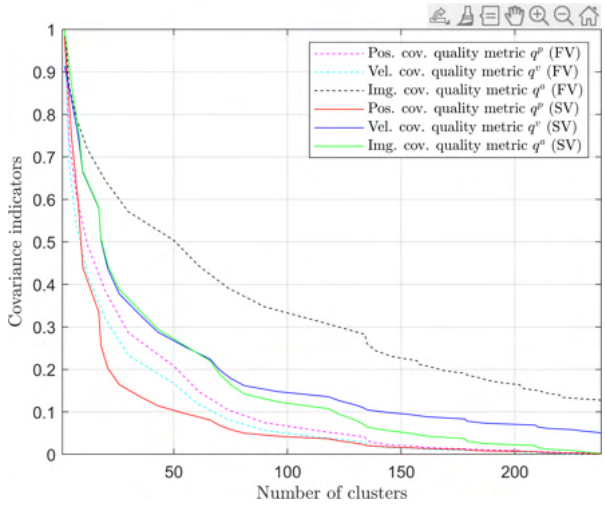
Fig. 8.14 Clustering quality metrics on the iCab dataset with a different number of clusters for the FV (a), the SV (b), and a comparison of the two cases (c).



(a)

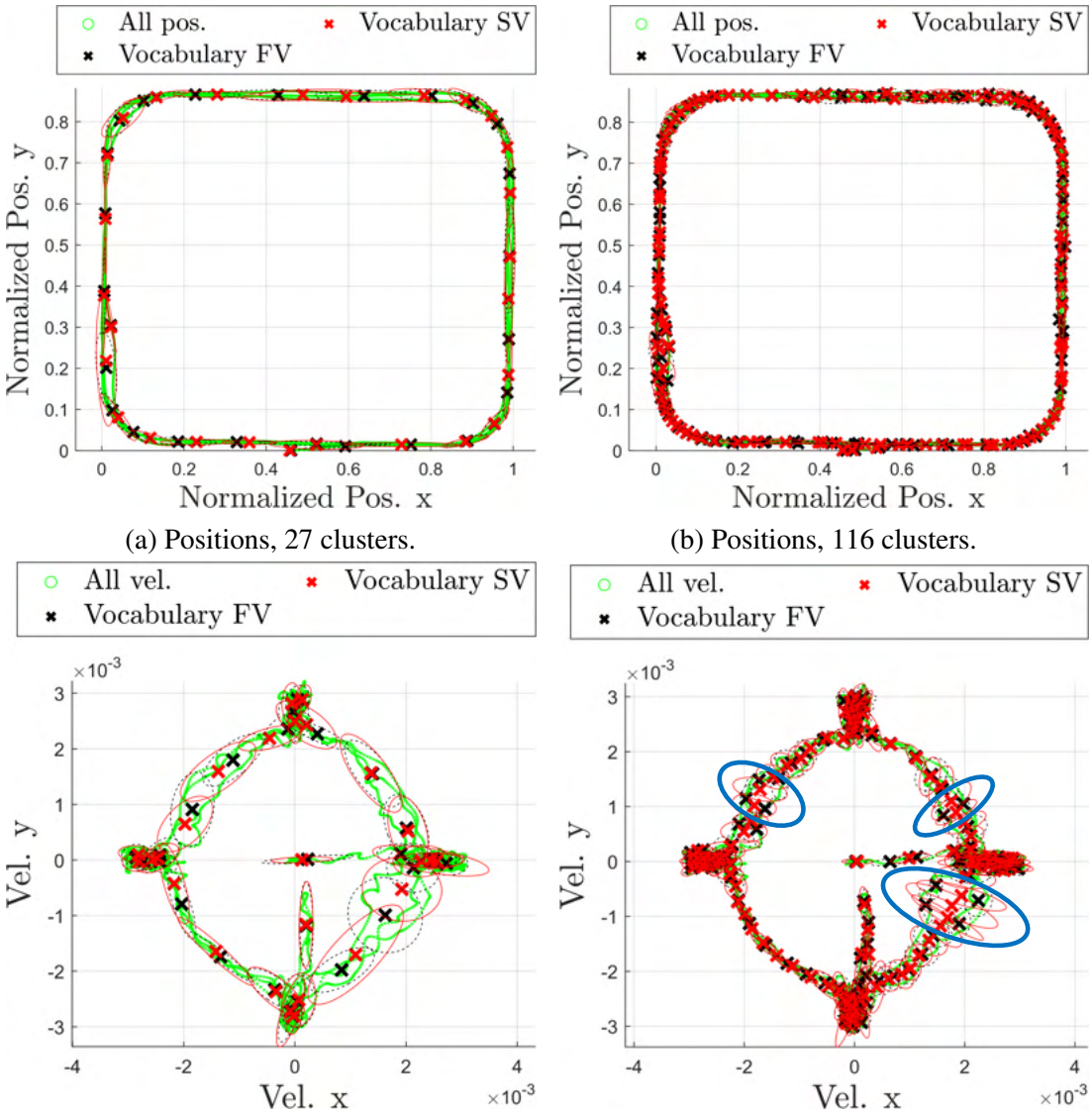


(b)



(c)

Fig. 8.15 Clustering quality metrics on the LAM drone scenario with a different number of clusters for the FV (a), the SV (b), and a comparison of the two cases (c).



(a) Positions, 27 clusters.

(b) Positions, 116 clusters.

(c) Velocities, 27 clusters.

(d) Velocities, 116 clusters.

Fig. 8.16 Cluster means and extensions for the iCab dataset, on the position space (upper part) and velocity space (lower part), with 27 clusters (part on the left) and 116 clusters (part on the right).

iCab dataset as it is the simplest one from the point of view of the odometry (having simple motions in a 2D space).

In Figs. 8.16a and 8.16b, we display the positional cluster centers for the FV (black) and SV (red) when using 27 and 116 clusters, respectively. The extension of the clusters is also represented by ellipses with axes proportional to the standard deviation of the clusters. Figs. 8.16c and 8.16d are related to the same two cases (27 and 116 clusters), but display velocity plots.

Table X Evaluation indices that highlight the method's capability to retrieve the original cluster when provided with a single data point. See the text for further explanation.

Dataset	Ratio correct cluster (better ↑)	MSE odom. dist. (m) (better ↓)	Bhatta odom. dist. (better ↓)
Icab, 27 clusters, FV	0.7113	**6.6973**	**114.1005**
Icab, 27 clusters, SV	**0.7368**	7.3099	142.4723
Icab, 45 clusters, FV	0.6918	**6.0767**	**194.3291**
Icab, 45 clusters, SV	**0.7813**	6.1775	259.6680
Icab, 116 clusters, FV	0.6804	4.9142	603.0445
Icab, 116 clusters, SV	**0.8697**	**3.3763**	**190.6402**
Egocart, FV	0.5959	3.0040	14.9611
Egocart, SV	**0.8089**	**1.3744**	**4.0040**
FM, FV	0.3990	0.2116	2.3175
FM, SV	**0.7637**	**0.0899**	**0.5567**
LAM, FV	0.5570	0.2013	2.0470
LAM, SV	**0.7904**	**0.0863**	**0.3401**

It is worth noting how, in the case of 27 clusters, many clusters have similar positional and velocity centers in the FV and SV. Some clusters in the SV, instead, are moved to provide a better image homogeneity. This is more evident in the case with 116 clusters: observe the areas circled in blue in Fig. 8.16d. In the FV, the clusters in this area are distributed to favor a more precise velocity prediction. Therefore, the two velocity "pathways" in these areas get assigned to different clusters. Conversely, in the SV, the clusters join points on the two "pathways". The reason behind this is the trade-off between the velocity accuracy and image homogeneity. This plot thus explains the behavior of the q^v quality metric in the SV compared to the FV, discussed in the previous section and illustrated in Fig. 8.14c.

In which case can I retrieve the original cluster better if I am provided with one image data point only?

As discussed in Sections 8.2.4 and 8.4, the first operations performed in the tracking are the particle-independent ones. A first estimate of the probability of being in each cluster is computed using a single image data point. We compare here the performance of the FV and SV in this operation.

Table X displays three metrics for this purpose. All metrics only relate to the particle-independent calculations and, therefore, are computed through the use of a single image data point, independently from all other points and without considering the rest of the tracking. Thus, they are not metrics on the final result, but on an intermediate computation.

The values in the first column ("ratio correct cluster") are calculated as follows. First, we find the probability $\boldsymbol{\alpha}_{v,t}$ of being in each cluster, given an image data point. The cluster displaying the highest probability is selected and compared with the "true cluster". With "true cluster" we identify the odometry cluster closest to the odometry point. Comparing every assignment with the "true clusters", we compute the ratio of correct cluster assignments displayed in the table.

The second and third columns of the table show the average distance between the center of the predicted cluster and of the "true cluster". They display the MSE and Bhattacharyya distances, respectively.

The table presents these performance indices for the iCab dataset (on three different clustering levels), and on the Egocart, FM, and LAM datasets, with the FV and SV. For each compared couple, the best index is highlighted in bold. The ratio in the first column is consistently better (i.e., higher) in the SV. The indices in the second and third columns tend to have better (i.e., lower) values in the SV as well. The SV of the approach, therefore, starts from an improved estimation of the cluster probabilities vector $\boldsymbol{\alpha}_{v,t}$.

How does the evolution between clusters change?

In the SV of the approach, the clusters are built considering both odometry and video. This can lead to a more entropic transition matrix. The reason is that a zone that could compose a single cluster in the FV (due to its motion homogeneity), could instead be split into several consequent clusters in the SV (due to its visual heterogeneity). Transition matrices that are more entropic might lead to the need for more particles because they generate more hypotheses of transition between clusters. This can be seen as a (small) disadvantage of the SV.

Table XI illustrates this concept on the different considered datasets. In the first and second columns, we show the transition matrix entropy and the mean entropy across the TTMs. The last column displays the mean number of time instants spent in a cluster before moving to the next one. The entropies of the matrices tend to be slightly higher in the SV. Coherently with this, in the SV, the agent tends to spend less time in each cluster, and, therefore, to jump more often between clusters. In the cases in which the inverse happens (a higher entropy and lower time in the FV), the margin of difference tends to be small. For each compared couple, we highlight in bold the case with lower entropy and higher time spent in a cluster.

Table XI Evaluation indices showing the entropy of the transition matrix and Temporal Transition Matrices, and how much time is spent on average in one cluster before moving to the next one. See the text for further explanation.

Dataset	Trans. Mat. Entropy	TTMs mean Entropy	Mean time instants spent in a cluster
Icab, 27 clusters, FV	**0.7118**	**0.1073**	**54.0854**
Icab, 27 clusters, SV	0.8589	0.1207	35.9837
Icab, 45 clusters, FV	**0.4803**	**0.0560**	**33.9773**
Icab, 45 clusters, SV	0.5696	0.0576	28.1887
Icab, 116 clusters, FV	**0.2511**	0.0152	**17.0038**
Icab, 116 clusters, SV	0.2755	**0.0145**	13.3482
Egocart, FV	0.1852	0.0094	52.7416
Egocart, SV	**0.1720**	**0.0070**	**60.8092**
FM, FV	0.4257	**0.0425**	36.3727
FM, SV	**0.3937**	0.0452	**42.8511**
LAM, FV	0.3971	**0.0279**	**9.1113**
LAM, SV	**0.3960**	0.0345	6.2284

8.6.9 How does the precision of the clusters change in 2D vs. 3D data?

In this section, we investigate the variation in the required number of clusters based on whether we are using 2D or 3D data. First, however, we must examine the datasets to identify those in which it makes sense to perform a comparison.

Figs. 8.17 and 8.18 compare several characteristics of the iCab, Egocart, FM, and LAM datasets.

In Fig. 8.17a, the number of training data points for each dataset is shown. The drone datasets are bigger than the iCab and Egocart datasets. However, the same zones are crossed multiple times, since many repetitions of the experiments are performed (as in the iCab case too).

Fig. 8.17b displays the covered area in cubic meters (m^3), obtained by multiplying the range along each dimension, in Fig. 8.17c. These two figures refer to positional information. Therefore, to remind the reader which are the trajectories followed by the vehicles in the four cases, we report them in Fig. 8.18. It is evident that, in a dataset such as iCab, the covered area is vast, but it is crossed in a very sparse way, as only its perimeter is treaded. Conversely, the space is covered in a much denser way in the Egocart and drone datasets. To account for this difference, we divide the covered space into zones of one cubic decimeter of volume. We then count the number of these zones that are crossed by the vehicle. The results are shown in Fig. 8.17d. It is worth noting that, as expected, the covered decimeters for the iCab are

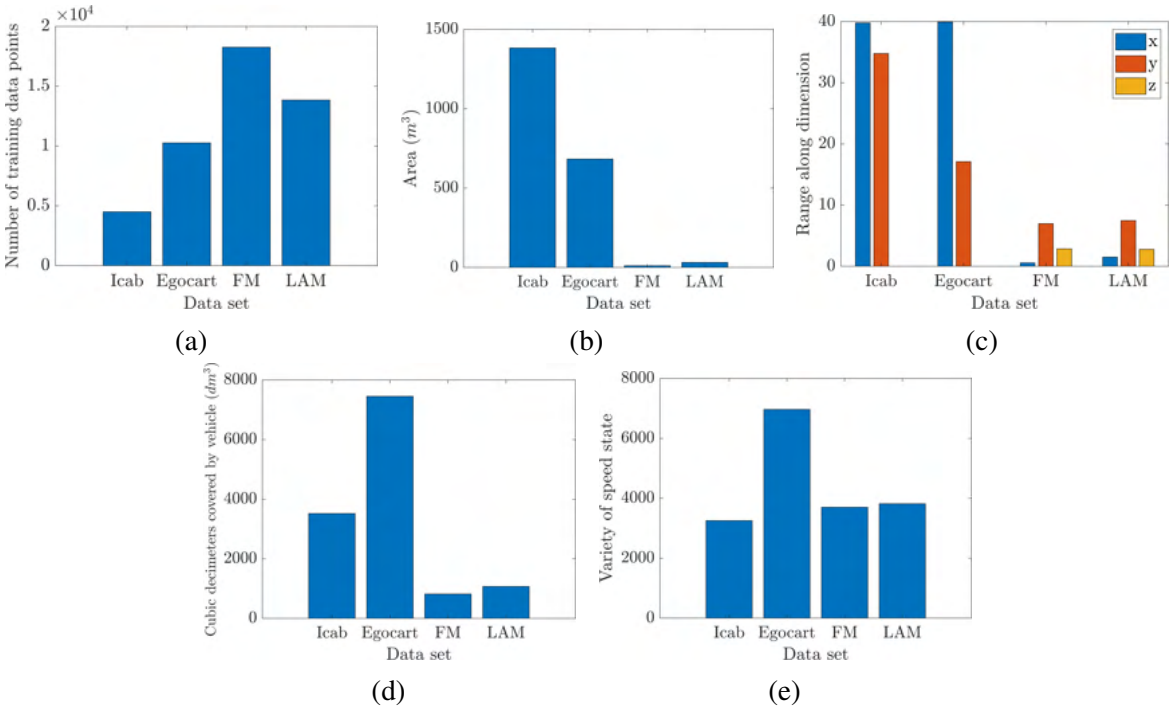


Fig. 8.17 Histograms showing various characteristics of the iCab, Egocart, FM, and LAM datasets: number of data points (a), area in m^3 (b), positional range along the dimensions x, y, and z (c), cubic decimeters covered by the vehicles (d), variety of the velocity information (e).

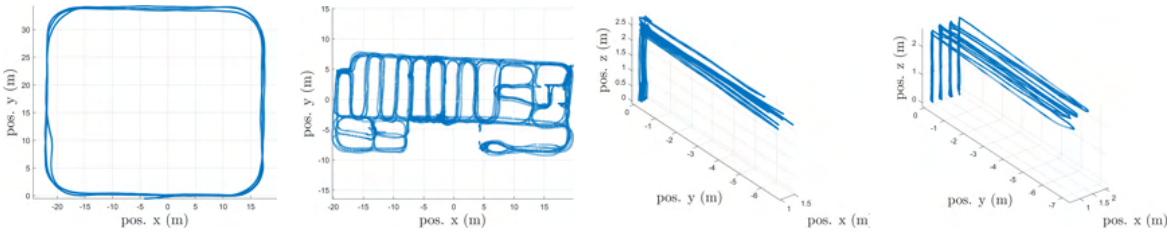
significantly fewer than for the Egocart, even if this covered a smaller area. The bins of the FM and LAM datasets increase in height compared to Fig. 8.17b as well.

Finally, Fig. 8.17e estimates the variety of motions performed in each dataset. The velocity space is divided into the same bins for the four datasets, and the number of bins with at least one element is counted. This shows the motion complexity of the FM and LAM datasets. Despite being extracted on a small area, these two datasets display a variety of motions as vast as the iCab dataset. The Egocart dataset, however, is the most complex from the point of view of motion variety. Fig. 8.19 also shows the velocity space for the four datasets.

Therefore, both terrestrial vehicles cover a much vaster area, iCab more sparsely and Egocart more densely. However, the two drone scenarios have a great variety of viewpoints in the small area where they move and a great variety of motion types.

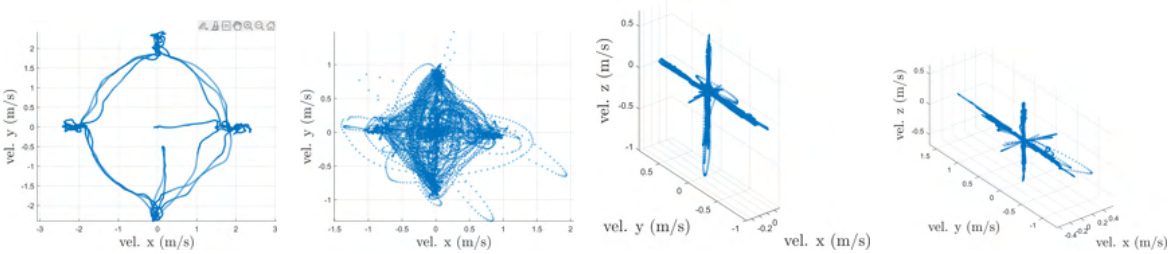
This first analysis considers the odometry characteristics. Let us now examine the visual complexity of the four datasets. We perform this examination in two ways.

Firstly, we standardize the images based on the mean and standard deviation of the pixels across each entire dataset. Then, the entropy is calculated for each image. Finally, the



(a) ICab. (b) Egocart. (c) Drone FM. (d) Drone LAM.

Fig. 8.18 Positions covered by the iCab (a), Egocart (b), FM (c), and LAM (d) datasets.

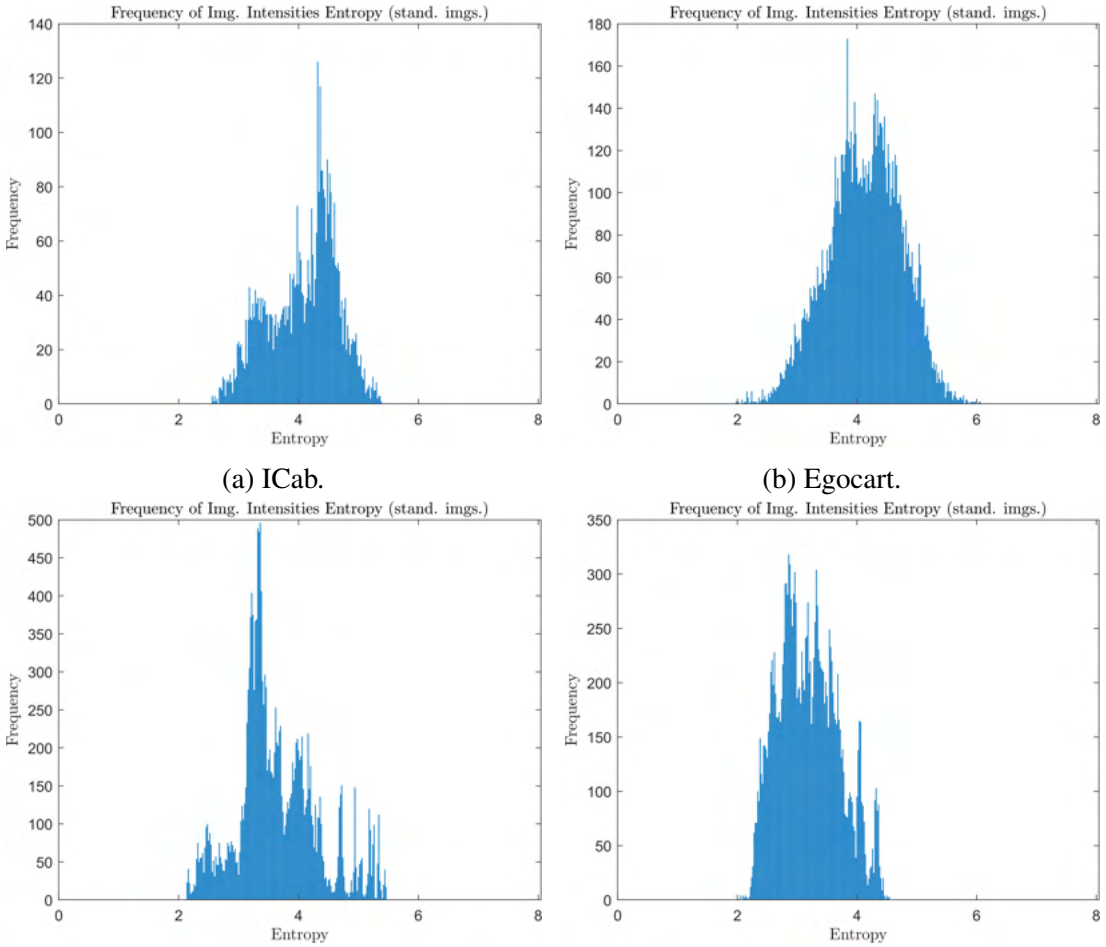


(a) ICab. (b) Egocart. (c) Drone FM. (d) Drone LAM.

Fig. 8.19 Velocities taken by the iCab (a), Egocart (b), FM (c), and LAM (d) datasets.

histogram of image entropies is plotted in Fig. 8.20. These plots illustrate how often highly entropic images are found in the dataset. Complex images with many details tend to be more entropic. We can observe that iCab has many complex images. Egocart has both complex and simpler images; in particular, there are several images with higher entropy than the ones found in the other three datasets. The difference between the FM and LAM is also interesting. Both scenarios include many images with low entropy, such as zones where the drone sees big parts of the red floor or of the blue columns (see Figs. 8.21a and 8.21b). However, the FM includes images with higher entropy than LAM. This is because the LAM scenario has a much more restricted viewpoint, as the whiteboard, the blue columns, or the red floor are always visible in a part of the images, and are uniform elements. In contrast, the FM dataset, while moving forward in the environment, can observe a vast area filled with many objects. To understand this, compare Figs. 8.21c (which belongs to the LAM scenario) and 8.21d (which belongs to the FM scenario).

The computed histograms illustrate the distribution of complex images inside the datasets. However, we could also assess the overall dataset complexity, i.e., the complexity of encoding the entire dataset. To perform this, we first compute the Principal Component Analysis (PCA) of the dataset. Then, we find the percentage of variance (or "proportion of variance") explained by each principal component [111]. It gives a sense of how much information each principal component retains from the original data. Fig. 8.22 displays the percentage



(a) ICab. (b) Egocart.

(c) Drone FM. (d) Drone LAM.

Fig. 8.20 Histogram of the image entropy in the iCab (a), Egocart (b), FM (c), and LAM (d) datasets.

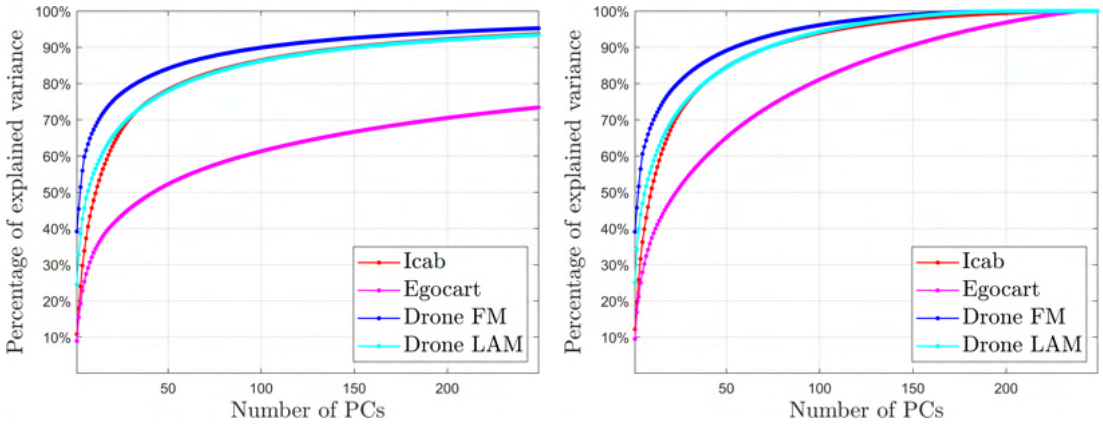
of explained variance across the number of considered principal components (PCs). Not all components are shown: we chose instead to stop at the first 250 for better readability of the plot.

Fig. 8.22a uses the entire dataset to calculate the values in the plot. Instead, to build Fig. 8.22b, we randomly selected only 250 data points from each dataset. In this second way, we ensure that the length of the dataset does not impact the final results. Instead, only the image content is relevant.

More complex datasets need more components to explain the same percentage of the explained variance. The Egocart dataset appears to have a significantly higher complexity than the other datasets. The drone FM scenario is less complex, whereas the drone LAM scenario and iCab display similar complexity. Although it has some complex images, the

(a) (b) (c) (d)

Fig. 8.21 Images from the FM and LAM scenarios. (a) and (b) show a viewpoint shared by both scenarios. (c) and (d) belong to the LAM and FM scenarios, respectively.



(a) (b)

Fig. 8.22 Percentage of the explained variance of each dataset, when considering different Principal Components (PCs) of the PCA.

FM scenario always follows a similar trajectory. In contrast, the LAM case, while having simpler images, starts at five different distances from the whiteboard and, therefore, has a higher variety of viewpoints (see also Fig. 8.17d). The iCab scenario, instead, has a lot of viewpoints (see Fig. 8.17d again), but many of them share a very similar appearance due to the symmetry of the courtyard. Therefore, its overall visual complexity is lower.

Therefore, we can now draw some conclusions from the analysis of the odometry and video complexity of the datasets. Egocart is the most complex dataset among the four, densely covering a large area with a variety of images. Consequently, we expect it to require many more clusters than the other three datasets, and doing a comparison between it and the used drone datasets would not be fair.

On the other hand, we can compare the drone datasets against the iCab one. We observed how the iCab has fewer frames (but also fewer repetitions of the same motion), but covers (sparsely) a larger area. On the other hand, the drone datasets cover a smaller area, but, due to the 3D motion, have more viewpoints of it and a larger variety of motions. Thus, we choose to qualitatively compare the iCab and LAM scenarios, which display a higher

similarity. However, we must observe that the comparison has limitations, due to some marked differences in the two datasets, such as the covered area. To perform the comparison, observe Figs. 8.14 and 8.15. It is worth noting that the image covariance quality metric decreases faster (i.e., requiring fewer clusters) to the regime in the drone dataset than in the terrestrial dataset. In contrast, the opposite happens with the velocity covariance quality metric. This suggests that drone datasets require more clusters due to their more complex motions in 3D environments.

8.6.10 How do visual repetitions in the dataset influence the performance of the method?

We have previously observed that, among the employed evaluation datasets, some displayed bigger and more frequent visual similarities between points at a high positional distance from each other. In particular, the iCab dataset is the one displaying the most evident visual repetitions, as observable in Fig. 8.6. In this section, we analyze the visual repetitions in the iCab, Egocart, FM, and LAM datasets, and we discuss how they can influence the performance of our approach (and of Visual-Based Localization methods in general).

For each dataset, we compute a 2D histogram, as displayed in Fig. 8.24. On the x-axis is the positional distance between couples of frames; on the y-axis is an indicator of the image similarity. The color represents the probability associated to each histogram bin. In this way, we can observe at what distance similar images are typically located.

The histogram is calculated as shown in Fig. 8.23. We give each image x_t belonging to the dataset as input to an Inception-v3 (IV3) network [212] and extract the corresponding latent state ς_t. We then perform a pairwise comparison of each couple of data points t and h. The comparison considers both the image content and positional distance. A vector of positional distances *PD* is found, by computing the Euclidean distance between each position x_t^o and x_h^o. A vector of latent state distances *LD* is also obtained, by calculating the Euclidean distance between each latent state ς_t and ς_h. The space covered by the values in these two vectors is divided into 50 sections each. The joined space is composed, therefore of $50x50 = 2500$ bins. The number of couples t-h falling in each bin is calculated, and the values are normalized, thus obtaining the desired histogram heatmap.

We can now comment on the heatmaps in Fig. 8.24. First, let us note how the iCab heatmap (Fig. 8.24a) displays a sawtoothed progression of the most probable feature distances (i.e., the brighter areas), with the increase of the positional distance. Two zones with high visual similarity at a high positional distance are identified and marked with red squares. The Egocart dataset (Fig. 8.24b), instead, has a sparser distribution across the bins. We can also

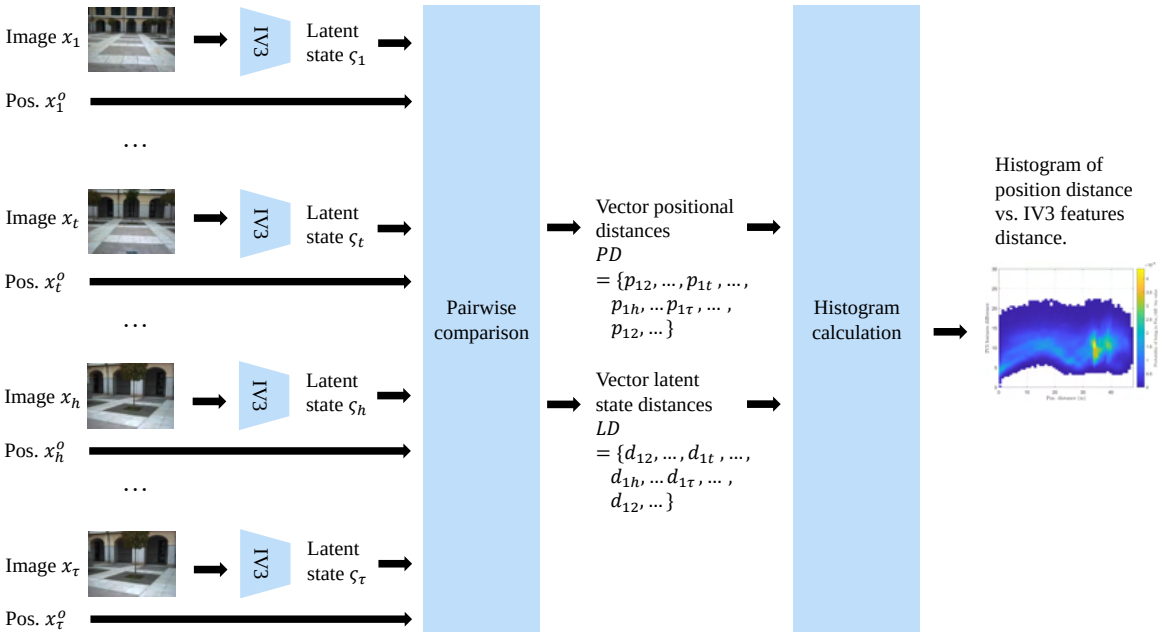
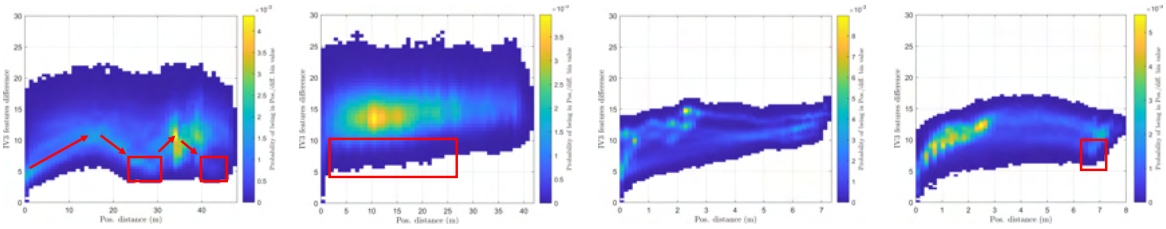


Fig. 8.23 Method for the calculation of the heatmap histogram.



(a) ICab. (b) Egocart. (c) Drone FM. (d) Drone LAM.

Fig. 8.24 Histogram heatmaps for the iCab (a), Egocart (b), FM (c), and LAM (d) datasets. On the x-axis is the positional distance. On the y-axis is the IV3 features distance.

observe that very far (> 25 m) points tend to have a higher feature distance than points at a lower positional distance (0-25 m). The drone FM (Fig. 8.24c) dataset has no particular concentrations of similar images at high positional distances: the feature distance tends to increase with the positional distance. Finally, in the LAM dataset (Fig. 8.24d), the feature distance tends to increase with the positional distance, but, at a high positional distance (~ 6.5 m), there are many visually similar images.

Let us examine the four cases in more detail with some examples.

Fig. 8.25 displays three couples of points from the iCab dataset as examples. On the left, a red cross identifies the bin where the couple is located in the histogram. The frames corresponding to the two points are displayed in the middle. Finally, on the right, the positions of the two points are shown on the trajectory plot. The sawtooth progression observed in

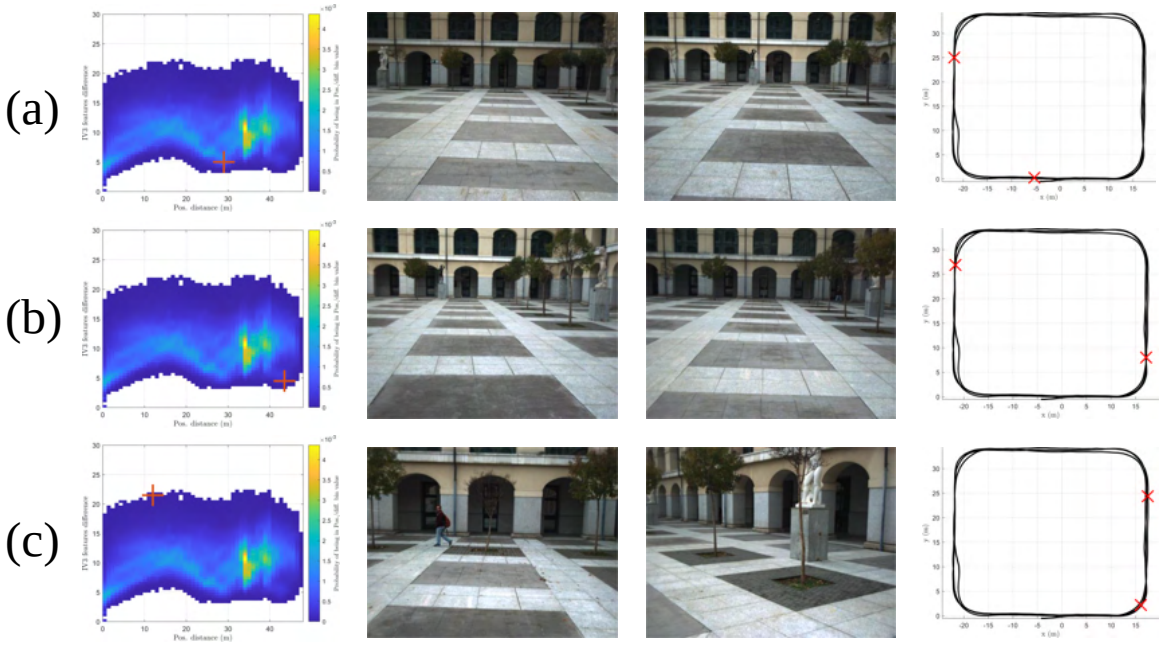


Fig. 8.25 Three examples from the iCab dataset.

the previous paragraph is easily explained from the image-position examples. Starting from two points a couple of centimeters from each other, as the positional distance increases, the visual features distance also increases, until points such as the ones displayed in case (a) in the figure. These two points are located at symmetrical zones in the courtyard, on two consequent sides. Then, increasing the positional distance again, the visual features distance also increases again, until points such as the ones displayed in case (b). These two points are at symmetrical zones in the courtyard, on two opposite sides. Subfigure (c), instead, shows two very different images from the iCab dataset.

The presence of such stark visual repetitions in the iCab dataset constitutes an added difficulty for VBL methods because a point could be predicted as being located on the wrong side of the courtyard. For this reason, many methods in Table IV displayed bad performance. Methods taking into consideration several data points for localization tend to give better results. Through the use of the Particle Filter, our method can wait several time instants before deciding what hypotheses to carry on (i.e., through resampling). Some hypotheses will be on different sides of the courtyard.

This characteristic of the dataset also influences which parameters, during the optimization in Section 8.6.7, tend to be favored. Notably, we have observed how, in the iCab dataset, $N_{th,l}$ tends to be chosen lower than in the Egocart dataset so that the first resampling can be performed after more time instants.

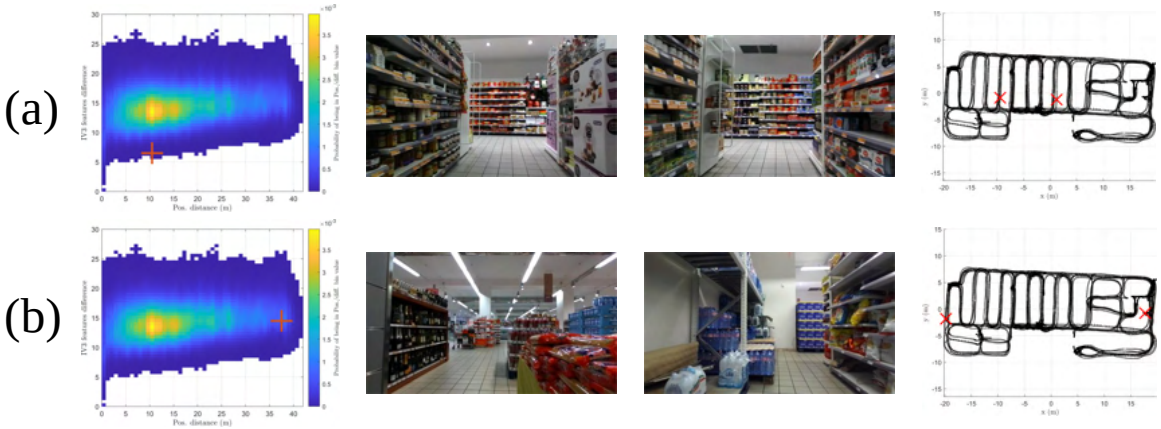


Fig. 8.26 Three examples from the Egocart dataset.

Fig. 8.26 shows two examples from the Egocart dataset. The first example (a) considers points at a distance of $\sim$ 10 m, whereas the second example (b) takes points at a distance of $\sim$ 35 m. In (a), we can observe two aisles with very similar appearance. In contrast, (b) displays very different images. The reason why very different images are located at a very high distance ($>$ 25 m) is due to the structure of the supermarket. In the center, there are many aisles (which tend to display similar appearance). In contrast, the opposite sides of the supermarket have more open areas with more distinctive visual features.

In the SV of our method, using clusters with a lower image covariance (see Fig. 8.15c), allows us to obtain better localization (see Table II) than in the FV because we can distinguish more easily on which aisle, among the visually similar ones, the vehicle is moving.

We don't show examples for the FM drone scenario, since we observed that it does not display areas with high visual similarity at far positions. The absence of these visual repetitions makes the VBL much simpler. Indeed, we observed in Table V that most methods tend to exhibit good and very similar results.

Finally, Fig. 8.27 illustrates two examples from the LAM drone scenario. Firstly, while in the same environment as the FM scenario, this dataset tends to have a higher number of similar images. This can be one explanation for the worse localization results observed on this dataset in Table VI. Many images at a distance of around 6.5 meters from each other also tend to display visual similarity. The reason is illustrated in Fig. 8.27(a). At both these two points, the drone is located in front of the blue columns, which have a similar appearance. Subfigure (b), instead, shows an example of two very visually different points. The first corresponds to when the drone is on the ground, and the second to when it is in the middle of the flight between two columns.

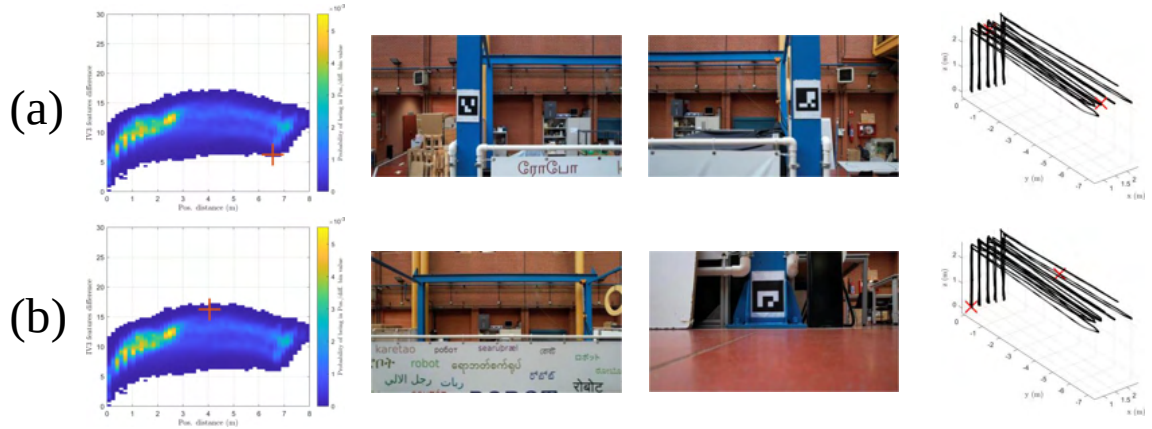


Fig. 8.27 Three examples from the LAM drone scenario.

To summarize, in this section, we have analyzed how visual repetitions constitute an important difficulty in VBL methods, and how they can affect the final results shown in the tables in Section 8.6.3.

8.7 Conclusions

We proposed a DBN-based model for simultaneous localization and anomaly detection, which is inserted in our self-awareness framework for autonomous systems. The proposed method combines Deep Learning with traditional Signal Processing methods for filtering and tracking. The method is hierarchical, probabilistic, explainable, data-driven, and multisensorial. Two alternative versions of the approach are presented, which differ in the information used to perform clustering.

In the result section, we focused on examining the localization accuracy of the method and also evaluated its explainability and its capability for anomaly detection. The two versions of the method are compared. Several important questions are posed and discussed, regarding the difference between the FV and SV of the approach, the precision of the clusters in a 2D case vs. a 3D case, and the presence of visual repetitions in the data.

It is worth noting that one disadvantage of the approach proposed in this chapter is the number of parameters. The grid search can be very time-consuming, even when the same threshold is applied for all anomalies. Therefore, a method to more efficiently choose the parameters could be developed. Reasonings could be made using the topology of the map (as we observed, maps with frequent intersections require a lower $N_{th,h}$) or the visual repetitions in the data (datasets with many visual repetitions in positions far from each other require a lower $N_{th,l}$) or other characteristics of the data and of learned vocabulary.

Only video data are used at testing time in the proposed method. Future work could consider the addition of a second sensor to use during both the training phase and the testing phase, such as the IMU. The IMU would increase localization performance, especially in cases in which the video could not be reliable (e.g., shadows).

Chapter 9

Conclusions and Future Work

In this chapter, we draw some conclusions regarding the work performed in this thesis and we analyze possible future research directions.

9.1 Conclusions

In this thesis, we introduced novel approaches within a self-awareness framework for autonomous vehicles. We analyzed the significance and the evolution of autonomous vehicles, with a focus on how neurobiological studies have recently influenced their development. Our work was inspired by these neurobiological studies and was aimed at proposing multisensorial, data-driven, hierarchical, and interpretable approaches, that adopt Bayesian inference. This is achieved by using a self-awareness framework based on DBNs. Central to this framework is the concept of self-awareness. The minimum requirements for an agent to be defined as self-aware are six capabilities: initialization, memorization, inference, anomaly detection, model creation, and interface with control. These capabilities enable to perform incremental learning of new models. In our work, we focused on the first four capabilities, with a particular emphasis on anomaly detection. Notably, anomaly detection is a fundamental step in the continual learning cycle because it identifies which parts of the model should be updated, based on new information. The hierarchical nature of DBNs allowed us to detect anomalies at different levels, comparing top-down predictions with bottom-up messages derived from the observations. We developed approaches for low-dimensional odometry data, high-dimensional video data, and for the fusion of the two modalities. These methodologies drew inspiration from Friston's Generalized States, theory of linear attractors, and free energy principle. Furthermore, we observed the importance of the vehicle being able to localize itself while navigating an environment. Therefore, in the last chapter, we

made the hypothesis that the vehicle always operated within a learned environment, and we addressed the problem of VBL.

All presented approaches developed novel extensions of the MJPF filter. These extensions were tailored for different use cases: some were integrated with a VAE and high-dimensional data (Chapter 5), others were devised for low-dimensional data (Chapter 6), and others were designed to work with a KVAE and with a model fusing video and odometry data (Chapters 7 and 8).

First, Chapter 5 introduced two alternative approaches for video data. In both approaches, the high-dimensional data were first brought to low-dimensionality through a VAE. A Generalized State was built on the VAE bottleneck. In the first version of the method, the GS was created by combining the VAE latent state at a certain time instant with the difference between it and the previous latent state; in the second version, the GS was obtained by encoding both the camera frame and the OF between subsequent frames. In both versions, clustering was then performed on the GSs, and a vocabulary was extracted, including a prediction neural network for each cluster. When the model was presented with new data, it leveraged the learned vocabulary to make predictions at the different levels of the DBN hierarchy. It then compared these predictions with the observed data to detect anomalies and to identify which portions of the model should be relearned. This chapter showed that the adopted framework, first proposed for low-dimensional data, could be extended for high-dimensional data too. The coherency between how DBN models are learned and employed in the two cases is greater, the higher we move in the hierarchy. We have seen that the cluster level performs predictions in the same way for the two data modalities (i.e., with a transition matrix) and that anomalies can be calculated with the same anomaly distance (i.e., the KLDA). In contrast, the observation model for high-dimensional data cannot be represented by a simple observation matrix; instead, a VAE is used. VAEs are, however, coherent with the adopted framework, because they are probabilistic and generative models. This chapter presents one significant deviation from the low-dimensional data treatment at the state level: the utilization of neural networks instead of piece-wise linear prediction models. We addressed this divergence in Chapters 7 and 8. The two versions of the method proposed in Chapter 5 were tested on the iCab dataset and discussed in detail. It was observed that the second version provided better anomaly detection accuracy. The second version was also tested on the Avenue and Subway anomaly detection benchmark datasets and was shown to be competitive against other state-of-the-art methods.

In Chapter 6, an approach for low-dimensional data was proposed. In this case, no reduction of dimensionality through the VAE was necessary. In this chapter, the concept of linear attractors introduced by Friston was used as inspiration. The developed filter, the G-

MJPF, aimed at separating the characteristics of the motion of the agent towards the attractor and along the orthogonal direction. In this way, a higher level of interpretability could be achieved. The proposed filter was also applied to perform driver behavior classification. In this case, multiple G-MJPFs are built, one for each class. The most significant clusters for each class are identified. Then, all filters run in parallel when classifying data. This chapter showed how the interpretability of the original MJPF model could be further improved. Moreover, it allowed us to examine novel uses of the proposed method for supervised applications such as driver behavior classification. The method was tested on the iCab dataset and on the UAH-DriveSet dataset. A comparison with other methods tested on the UAH-DriveSet dataset was proposed. It is worth reminding the reader that it was difficult to perform a precise accuracy comparison, due to differences in the training procedure across papers. However, the results obtained with the proposed methods appeared to be in line with other approaches from the state of the art.

After considering the low-dimensional and high-dimensional data separately, we studied how to combine them. In Chapter 7, we proposed a CG-KVAE, a KVAE guided in its learning of the video models by the clustering learned over the odometry data. With this approach, the information of the simpler low-dimensional data is leveraged for the learning over the more complex high-dimensional data. Instead of using a VAE and non-linear prediction models as in Chapter 5, a KVAE with a combination of linear prediction models is employed. Therefore, the contribution of this chapter in the context of the self-awareness framework was twofold. Firstly, we tackled the multi-sensorial aspect of the framework and analyzed a novel way of fusing the odometry and video data. Secondly, this chapter acted as a continuation of our experiments in Chapter 5. As mentioned above, in Chapter 5, the DBN state-level prediction still presented some inhomogeneity compared to a low-dimensional case, because non-linear models were adopted. In Chapter 7, we made a first attempt at obtaining a higher degree of homogeneity by using linear prediction models. The proposed method was tested on the iCab dataset and on part of the UAH-DriveSet dataset. We discussed how results vary by modifying the model and its parameters. We also compared the anomaly signals obtained with the proposed approach and with the reconstruction loss of a vanilla VAE, showing how our method provided better results.

Chapter 8 proposed two further extensions of this model, introducing a different assumption compared to the previous chapter. In Chapter 7, we had supposed that video and odometry data were both always present. In contrast, Chapter 8 hypothesized that, during training, both data sources were always available, but, during testing, only the camera information was available. Therefore, we developed two extensions of the CG-KVAE model that learned how to predict the odometry from the video data, through a set of additional

matrices. In the testing phase, the method was employed to perform Visual Based Localization, in addition to anomaly detection. Furthermore, we analyzed how several aspects of the datasets can influence our models, such as if the data are 2D or 3D, or the presence of image repetitions. We also addressed the problem of selecting the appropriate clustering level, to avoid over-clustering and under-clustering, offering a valuable contribution to the overall robustness of our methodology. Moreover, in this chapter, instead of using a combination of linear models as in Chapter 7, we employ a piece-wise formulation, which further increases the homogeneity with the odometry model. We compared the two versions of the proposed approach. The second version displayed higher localization accuracy: we achieved a mean localization error in meters of 1.65, 0.98, 0.23, and 0.87, on the Egocart, iCab, Drone FM, and Drone LAM datasets, respectively. Furthermore, we executed an in-depth analysis of the obtained results. First, we examined the obtained anomalies, the algorithm speed and memory requirements, how the localization results changed depending on the parameters, and on what visual features the learned network concentrated its attention. Finally, we posed and answered three important questions: “how does the difference between the FV and SV affect the training and testing?” (1), “how does the precision of the clusters change in a 2D vs. a 3D dataset?” (2), “how do visual repetitions in the dataset influence the performance of the method?” (3). We observed that a better video clustering metric was obtained in the SV, which explains the better localization accuracy in the SV of the method (1). Moreover, 3D drone datasets required more clusters than 2D datasets from terrestrial vehicles, due to the more complex motions in the environments. Furthermore, we noticed how visual repetitions in the dataset can worsen the performance of the method (3).

In addition to the proposed self-awareness models, we extracted an indoor drone dataset and we presented a method for estimating the drone odometry from the camera, IMU, and lidar observations (see Chapter 4). The obtained mean localization error was below 0.25 m in all scenarios. Part of the extracted dataset was employed later in the thesis for evaluating the proposed self-awareness methods.

All methods presented in the thesis were evaluated on a variety of real-world datasets and on one simulated dataset.

9.2 Drone dataset availability

A link to the extracted dataset described in Chapter 4 will be made public - after publication of the related paper - in the GitHub repository “https://github.com/GiuliaSlavic".

9.3 Future Work

Starting from the work described in this thesis, several future directions can be identified.

9.3.1 Closing the Continual Learning cycle

As mentioned in the previous section, the six self-awareness capabilities are designed to develop a cognitive system that continuously learns from new situations. The fifth capability, model creation, considers the generation of a new model leveraging the abnormal data. In this way, the new model can minimize the prediction error associated with data that was previously classified as abnormal. Therefore, anomalies and GEs are minimized, and so is free energy. When new situations are detected, new models are learned, resulting in the availability of multiple parallel models. During subsequent testing phases, the method can consequently switch between these models, based on their respective probabilities for the given data.

Furthermore, it is important to highlight that, when anomalies are detected, it is not always necessary to learn the entire model from scratch. Instead, the interpretability of the DBN and the presence of anomaly signals at different levels can be exploited to pinpoint which part of the model needs to be learned again. For example, in the method in Chapter 5, if the anomaly is at the reconstruction level, the entire model must be learned again. Instead, if only a high KLDA is detected, the clusters and, consequently, the prediction NNs should be relearned, but the VAE does not need to be modified. Similarly, if the anomaly emerges in the state prediction (or state prediction backpropagated at the observation level), the NNs, or a single NN, should be learned anew. This flexibility during the continual learning phase improves the practical applicability of our framework.

9.3.2 Further explaining the anomalies

Anomalies could be further explained in several ways. We identify here two possible ways.

Firstly, the combination of different sensory modalities can be leveraged to explain the anomalies of one modality from the anomalies of another one. By analyzing video anomalies, we can explain the possible cause of odometry anomalies. For instance, in a scenario with a car stopping due to pedestrians, not all video anomalies are associated with a change in the dynamics of the car. Indeed, some pedestrians walking on the sidewalk beside the car may not be causing the stop, while others crossing in front of the vehicle might be. By analyzing the video and odometry anomaly signals that are happening in a certain time window, it would be possible to understand which video anomalies (in appearance and motion) are more

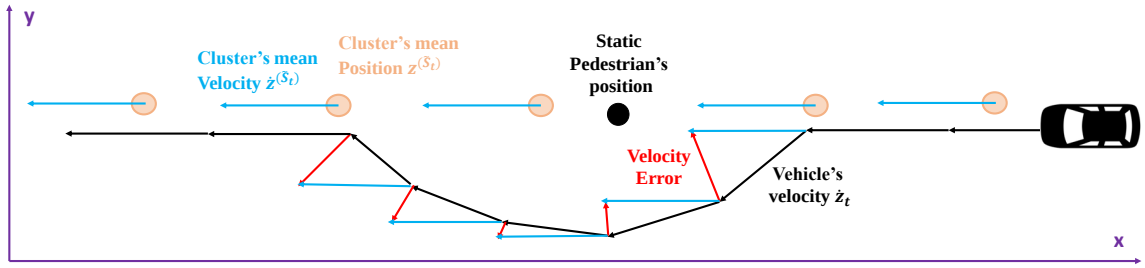
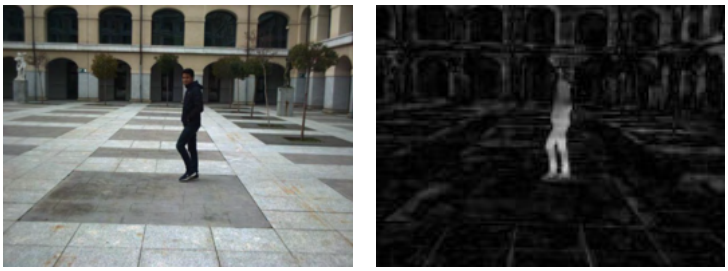


Fig. 9.1 Real vehicle velocity (black arrows) vs. expected velocity (cyan arrows). The red arrows correspond to the difference between the two. The expected velocity corresponds to the mean velocity of the corresponding cluster.



(a) (b)

Fig. 9.2 Original camera image from the Emergency Stop scenario (a) vs. corresponding direct reconstruction error image (b).

likely to be causing the odometry anomalies. In the provided example, this would correspond to understanding that the anomaly due to pedestrians crossing in front of the car (visual anomaly) is more likely to be the cause of the vehicle performing a stop maneuver (abnormal odometry). A similar scenario example to the one of the car and the pedestrians is provided by the FM drone dataset extracted in the context of this thesis. In this case, a drone observes pedestrians on its side and front, with only the latter ones generating motion anomalies. The FM scenario could be employed for further studies of anomaly detection in this direction. The LOM scenario can be adopted for the same purpose, as it contains visual anomalies that generate abnormal motions in the drone (i.e., the yellow objects) and other visual anomalies that do not impact on the movement of the drone (i.e., the brown boxes).

Secondly, not only the anomaly indicator, but also the GEs should be studied further. The anomaly indicator suggests at what level of the hierarchy the anomaly is detected, and, therefore, which models need to be changed. However, to really understand the anomaly, it is necessary to observe the GEs, i.e., how the model needs to be modified. Let us take the example of a vehicle avoiding an obstacle by performing a rotational motion around it (such as in the Pedestrian Avoidance scenario). In the odometry data, a rotational anomaly is

detected. However, the anomaly signal is a scalar value at each time instant and it does not give us information about the presence of an obstacle. Instead, by studying the generalized errors, we can understand how the vehicle had to change its motion to avoid the pedestrian, and we can also approximately estimate where the obstacle was located. Fig. 9.1 illustrates this example in a stylized way, with the vehicle trajectory in black, the expected motion in cyan, and the velocity error in red. Observe how the pedestrian acts as a repulsive force in the first half of the semi-circular motion, where the vehicle is performing the avoidance motion. From the velocity errors, it is possible to detect this repulsive force and estimate the position of the obstacle. Then, an attractive force acts on the vehicle, after the pedestrian has been surpassed because the vehicle is returning towards the original attractors (the clusters in orange).

Another example of the difference between studying the anomaly signals and the GEs can be done using video data. Again, in this thesis, we have studied the frame-level anomaly signals, which provide one scalar abnormality value for each frame. However, to better understand an observation-level camera anomaly, the error at the image level should be examined. An example of this error, confronted with the camera observation at the same time instant, is shown in Fig. 9.2. A first, simple examination of the error could consist in observing whether it is a concentrated dense error (like the one caused by a pedestrian) or an extended error (like the one caused by the observation of a new environment). More complex examinations could be performed, considering other properties of the error.

9.3.3 Further analyzing the anomalies

The anomalies extracted by the methods proposed in the thesis could undergo additional examination. For example, the extracted LAM drone dataset offers the possibility of studying the dimension of the anomalies that can be successfully detected. In the dataset, square visual anomalies of different sizes and colors are present.

9.3.4 Inserting other sensory modalities

Throughout this thesis, we employed camera and odometry data. The framework underlying our work was also employed for vehicular control information [115] and, subsequently, for lidar data [105]. However, we could extend our single-sensor methods to other types of sensory input. In our multi-sensor methods, other couples of sensory modalities could be joined; extensions with more than two sensors could also be developed. For example, the VBL method in Chapter 8 could be improved by the integration of other sensors, such as IMU. We could suppose that, while GPS data are never (or rarely) observable, both camera

and IMU data are available at all instants. This is a frequent condition when agents navigate indoor environments.

References

[1] Adam, A., Rivlin, E., Shimshoni, I., and Reinitz, D. (2008). Robust real-time unusual event detection using multiple fixed-location monitors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(3):555–560.

[2] Aggarwal, C. C. (2016). *Outlier Analysis*. Springer Publishing Company, Incorporated, 2nd edition.

[3] Aguado, E., Milosevic, Z., Hernández, C., Sanz, R., Oviedo, M. A. G., Bozhinoski, D., and Rossi, C. (2021). Functional self-awareness and metacontrol for underwater robot autonomy. *Sensors*, 21(4):1210.

[4] Alshazly, H., Linse, C., Barth, E., and Martinetz, T. (2019). Handcrafted versus cnn features for ear recognition. *Symmetry*, 11:1493.

[5] Amini, A., Schwarting, W., Rosman, G., Araki, B., Karaman, S., and Rus, D. (2018). Variational autoencoder for end-to-end control of autonomous driving with novelty detection and training de-biasing. In *International Conference on Intelligent Robots and Systems*, pages 568–575.

[6] Anderson, J. M., Kalra, N., Stanley, K. D., Sorensen, P., Samaras, C., and Oluwatola, O. A. (2014). Brief history and current state of autonomous vehicles. In *Autonomous Vehicle Technology: A Guide for Policymakers*, pages 55–74. RAND Corporation.

[7] Antipov, G., Berrani, S.-A., Ruchaud, N., and Dugelay, J.-L. (2015). Learned vs. hand-crafted features for pedestrian gender recognition. In *ACM international conference on Multimedia*, pages 1263–1266.

[8] Anzanpour, A., Azimi, I., Gotzinger, M., Rahmani, A. M., Taherinejad, N., Liljeberg, P., Jantsch, A., and Dutt, N. D. (2017). Self-awareness in remote health monitoring systems using wearable electronics. In *Design, Automation & Test in Europe Conference & Exhibition*, pages 1056–1061.

[9] Arnab, A., Zheng, S., Jayasumana, S., Romera-Paredes, B., Larsson, M., Kirillov, A., Savchynskyy, B., Rother, C., Kahl, F., and Torr, P. H. (2018). Conditional random fields meet deep neural networks for semantic segmentation: Combining probabilistic graphical models with deep learning for structured prediction. *IEEE Signal Processing Magazine*, 35(1):37–52.

[10] Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.-R., and Samek, W. (2015). On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLoS ONE*, 10(7).

[11] Badue, C., Guidolini, R., Carneiro, R. V., Azevedo, P., Cardoso, V. B., Forechi, A., Jesus, L. F. R., Berriel, R. F., Paixão, T. M., Mutz, F. W., de Paula Veronese, L., Oliveira-Santos, T., and Souza, A. F. D. (2021). Self-driving cars: A survey. *Expert Systems with Applications*, 165:113816.

[12] Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations*.

[13] Balaji, B. and Friston, K. (2011). Bayesian state estimation using generalized coordinates. In *Signal Processing, Sensor Fusion, and Target Recognition XX*, volume 8050, page 80501Y. SPIE.

[14] Bandyopadhyay, S., Datta, A., Sachan, S., and Pal, A. (2022). Unsupervised driving behavior analysis using representation learning and exploiting group-based training. *CoRR*, abs/2205.07870.

[15] Bastani, V., Marcenaro, L., and Regazzoni, C. S. (2016). Online nonparametric bayesian activity mining and analysis from surveillance video. *IEEE Transactions on Image Processing*, 25(5):2089–2102.

[16] Baur, C., Wiestler, B., Albarqouni, S., and Navab, N. (2018). Deep autoencoding models for unsupervised anomaly segmentation in brain mr images. In *4th International Workshop on Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries*, pages 161–169.

[17] Baydoun, M., Campo, D., Sanguineti, V., Marcenaro, L., Cavallaro, A., and Regazzoni, C. (2018). Learning switching models for abnormality detection for autonomous driving. In *International Conference on Information Fusion*, pages 2606–2613.

[18] Bayer, J. and Osendorfer, C. (2014). Learning stochastic recurrent networks. *ArXiv*, abs/1411.7610.

[19] Becker-Ehmck, P., Peters, J., and van der Smagt, P. (2019). Switching linear dynamics for variational bayes filtering. In *International Conference on Machine Learning*, pages 553–562.

[20] Benferhat, S., Leray, P., and Tabia, K. (2020). Belief graphical models for uncertainty representation and reasoning. In *A Guided Tour of Artificial Intelligence Research: Volume II: AI Algorithms*, pages 209–246. Springer.

[21] Bengio, Y., Deleu, T., Rahaman, N., Ke, R., Lachapelle, S., Bilaniuk, O., Goyal, A., and Pal, C. (2019). A meta-transfer objective for learning to disentangle causal mechanisms. In *International Conference on Learning Representations*.

[22] Bergasa, L. M., Almeria, D., Almazán, J., Torres, J. J. Y., and Arroyo, R. (2014). Drivesafe: An app for alerting inattentive drivers and scoring driving behaviors. In *IEEE Intelligent Vehicles Symposium Proceedings*, pages 240–245.

[23] Bhattacharyya, A. (1946). On a measure of divergence between two multinomial populations. *Sankhyā: The Indian Journal of Statistics (1933-1960)*, 7(4):401–406.

[24] Brahmbhatt, S., Gu, J., Kim, K., Hays, J., and Kautz, J. (2018). Geometry-aware learning of maps for camera localization. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2616–2625.

[25] Brambilla, M., Mascetti, P., and Mauri, A. (2017). Comparison of different driving style analysis approaches based on trip segmentation over GPS information. In *IEEE International Conference on Big Data*, pages 3784–3791.

[26] Bratman, M. (1987). *Intention, plans, and practical reason*. Harvard University Press.

[27] Broggi, A., Cerri, P., Debattisti, S., Laghi, M. C., Medici, P., Molinari, D., Panciroli, M., and Prioletti, A. (2015). Proud—public road urban driverless-car test. *IEEE Transactions on Intelligent Transportation Systems*, 16(6):3508–3519.

[28] Broggi, A., Cerri, P., Felisa, M., Laghi, M., Mazzei, L., and Porta, P. P. (2012). The vislab intercontinental autonomous challenge: An extensive test for a platoon of intelligent vehicles. *International Journal of Vehicle Autonomous Systems*, 10.

[29] Bronstein, A., Das, J., Duro, M., Friedrich, R., Kleyner, G., Mueller, M., Singhal, S., and Cohen, I. (2001). Self-aware services: using bayesian networks for detecting anomalies in internet-based services. *IEEE/IFIP International Symposium on Integrated Network Management Proceedings. Integrated Network Management VII. Integrated Management Strategies for the New Millennium (Cat. No.01EX470)*, pages 623–638.

[30] Campos, C., Elvira, R., Rodríguez, J. J. G., Montiel, J. M. M., and Tardós, J. D. (2021). ORB-SLAM3: an accurate open-source library for visual, visual-inertial, and multimap SLAM. *IEEE Transactions on Robotics*, 37(6):1874–1890.

[31] Cerri, P., Soprani, G., Zani, P., Choi, J., Lee, J., Kim, D., Yi, K., and Broggi, A. (2011). Computer vision at the hyundai autonomous challenge. In *14th International IEEE Conference on Intelligent Transportation Systems*, pages 777–783.

[32] Chakravarty, P., Zhang, A., Jarvis, R., and Kleeman, L. (2007). Anomaly detection and tracking for a patrolling robot. In *Proceedings of the Australiasian Conference on Robotics and Automation*, pages 1–9.

[33] Chandola, V., Banerjee, A., and Kumar, V. (2009). Anomaly detection: A survey. *ACM Computing Surveys*, 41(3):1–58.

[34] Chatila, R., Renaudo, E., Andries, M., García, R. O. C., Luce-Vayrac, P., Gottstein, R., Alami, R., Clodic, A., Devin, S., Girard, B., and Khamassi, M. (2018). Toward self-aware robots. *Frontiers in Robotics and AI*, 5:88.

[35] Chen, T., Bahsoon, R., and Yao, X. (2018). A survey and taxonomy of self-aware and self-adaptive cloud autoscaling systems. *ACM Computing Surveys*, 51(3):1–40.

[36] Chen, T., Faniyi, F., Bahsoon, R., Lewis, P. R., Yao, X., Minku, L. L., and Esterle, L. (2014). The handbook of engineering self-aware and self-expressive systems. *CoRR*, abs/1409.1793.

[37] Cheung, E., Bera, A., Kubin, E., Gray, K., and Manocha, D. (2018). Identifying driver behaviors using trajectory features for vehicle navigation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3445–3452.

[38] Chong, Y. S. and Tay, Y. H. (2015). Modeling representation of videos for anomaly detection using deep learning: A review. *ArXiv*, abs/1505.00523.

[39] Chong, Y. S. and Tay, Y. H. (2017). Abnormal event detection in videos using spatiotemporal autoencoder. In *Advances in Neural Networks - International Symposium on Neural Networks*, volume 10262 of *Lecture Notes in Computer Science*, pages 189–196.

[40] Chung, J., Kastner, K., Dinh, L., Goel, K., Courville, A. C., and Bengio, Y. (2015). A recurrent latent variable model for sequential data. In *Conference on Neural Information Processing Systems*, pages 2980–2988.

[41] Crauel, H. and Flandoli, F. (1992). Attractors for random dynamical systems. *Probability Theory and Related Fields*, 100:365–393.

[42] Cristianini, N. and Shawe-Taylor, J. (2010). *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press.

[43] Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 886–893.

[44] Dalal, N., Triggs, B., and Schmid, C. (2006). Human detection using oriented histograms of flow and appearance. In *European Conference on Computer Vision*, pages 428–441.

[45] Damasio, A. R. (1999). *The Feeling of What Happens: Body and Emotion in the Making of Consciousness*. Harcourt Brace.

[46] Das Gupta, S. (1980). Discriminant analysis. In *R.A. Fisher: An Appreciation*, pages 161–170. Springer New York.

[47] Dawid, A. and LeCun, Y. (2023). Introduction to latent variable energy-based models: A path towards autonomous machine intelligence. *CoRR*, abs/2306.02572.

[48] de Gevigney Valentin, D., Marteau, P., Delhay, A., and Lolive, D. (2020). Video latent code interpolation for anomalous behavior detection. In *International Conference on Systems, Man, and Cybernetics*, pages 3037–3044.

[49] Delfa, G. C. L., Monteleone, S., Catania, V., Paz, J. F. D., and Bajo, J. (2016). Performance analysis of visualmarkers for indoor navigation systems. *Frontiers of Information Technology & Electronic Engineering*, 17:730–740.

[50] Dennis, L. A. and Fisher, M. (2020). Verifiable self-aware agent-based autonomous systems. *Proceedings of the IEEE*, 108(7):1011–1026.

[51] Deshpande, A. (2018). *Graphical Models for Uncertain Data Management*. Springer.

[52] Doan, A., Halevy, A., and Ives, Z. (2012). *Principles of Data Integration*. Elsevier.

[53] Dosovitskiy, A., Ros, G., Codevilla, F., López, A. M., and Koltun, V. (2017). CARLA: an open urban driving simulator. In *1st Annual Conference on Robot Learning*, volume 78, pages 1–16.

[54] Doucet, A., de Freitas, N., Murphy, K. P., and Russell, S. J. (2000). Rao-blackwellised particle filtering for dynamic bayesian networks. In *Conference in Uncertainty in Artificial Intelligence*, pages 176–183.

[55] Dutt, N. D., Jantsch, A., and Sarma, S. (2016). Toward smart embedded systems: A self-aware system-on-chip (soc) perspective. *ACM Transactions on Embedded Computing Systems*, 15(2):1–27.

[56] Dutt, N. D., Regazzoni, C. S., Rinner, B., and Yao, X. (2020). Self-awareness for autonomous systems. *Proceedings of the IEEE*, 108(7):971–975.

[57] Dwek, N., Birem, M., Geebelen, K., Hostens, E., Mishra, A., Steckel, J., and Yudanto, R. (2020). Improving the accuracy and robustness of ultra-wideband localization through sensor fusion and outlier detection. *IEEE Robotics and Automation Letters*, 5(1):32–39.

[58] Elfring, J., Torta, E., and van de Molengraft, R. (2021). Particle filters: A hands-on tutorial. *Sensors*, 21(2):438.

[59] Elmokadem, T. and Savkin, A. V. (2021). Towards fully autonomous uavs: A survey. *Sensors*, 21(18):6223.

[60] Englund, C., Chen, L., Ploeg, J., Semsar-Kazerooni, E., Voronov, A., Bengtsson, H. H., and Didoff, J. (2016). The grand cooperative driving challenge 2016: boosting the introduction of cooperative automated vehicles. *IEEE Wireless Communications*, 23(4):146–152.

[61] Epstein, R. A., Patai, E. Z., Julian, J. B., and Spiers, H. J. (2017). The cognitive map in humans: spatial navigation and beyond. *Nature Neuroscience*, 20:1504–1513.

[62] Ertöz, L., Steinbach, M. S., and Kumar, V. (2003). Finding topics in collections of documents: A shared nearest neighbor approach. In *Clustering and Information Retrieval*, pages 83–104. Kluwer.

[63] Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*.

[64] Faniyi, F., Lewis, P. R., Bahsoon, R., and Yao, X. (2014). Architecting self-aware software systems. In *IEEE/IFIP Conference on Software Architecture*, pages 91–94.

[65] Farnebäck, G. (2003). Two-frame motion estimation based on polynomial expansion. In *Image Analysis*, pages 363–370. Springer Berlin Heidelberg.

[66] Feng, Y., Yuan, Y., and Lu, X. (2017). Learning deep event models for crowd anomaly detection. *Neurocomputing*, 219:548–556.

[67] Fix, E. and Hodges, J. L. (1951). Discriminatory analysis, nonparametric discrimination: Consistency properties. In *Technical Report 4, USAF School of Aviation Medicine, Randolph Field*.

[68] Fong, R. C. and Vedaldi, A. (2017). Interpretable explanations of black boxes by meaningful perturbation. In *IEEE International Conference on Computer Vision*, pages 3449–3457.

[69] Foorthuis, R. (2021). On the nature and types of anomalies: a review of deviations in data. *International Journal of Data Science and Analytics*, 12(4):297–331.

[70] Formentin, S., van Heusden, K., and Karimi, A. (2013). Model-based and data-driven model-reference control: A comparative analysis. In *12th European Control Conference*, pages 1410–1415.

[71] Fraccaro, M., Kamronn, S., Paquet, U., and Winther, O. (2017). A disentangled recognition and nonlinear dynamics model for unsupervised learning. In *Conference on Neural Information Processing Systems*, pages 3601–3610.

[72] Fraccaro, M., Sønderby, S. K., Paquet, U., and Winther, O. (2016). Sequential neural models with stochastic layers. In *Conference on Neural Information Processing Systems*, pages 2199–2207.

[73] Friston, K. (2009). The free-energy principle: a rough guide to the brain? *Trends in cognitive sciences*, 13(7):293–301.

[74] Friston, K. (2010). The free-energy principle: a unified brain theory? *Nature Reviews Neuroscience*, 11:127–138.

[75] Friston, K. (2012). A free energy principle for biological systems. *Entropy*, 14(11):2100–2121.

[76] Friston, K., Kilner, J., and Harrison, L. (2006). A free energy principle for the brain. *Journal of Physiology*, 100(1-3):70–87.

[77] Friston, K., Sengupta, B., and Auletta, G. (2014). Cognitive dynamics: From attractors to active inference. *Proceedings of the IEEE*, 102(4):427–445.

[78] Fritzke, B. (1994). A growing neural gas network learns topologies. In *Conference on Neural Information Processing Systems*, pages 625–632.

[79] Garg, S., Kaur, K., Kumar, N., and Rodrigues, J. J. P. C. (2019). Hybrid deep-learning-based anomaly detection scheme for suspicious flow detection in sdn: A social multimedia perspective. *IEEE Transactions on Multimedia*, 21(3):566–578.

[80] Ghahramani, Z. (1998). Learning dynamic bayesian networks. In *Adaptive Processing of Sequences and Data Structures: International Summer School on Neural Networks*, pages 168–197. Springer Berlin Heidelberg.

[81] Gibson, J. and Carmichael, L. (1966). *The Senses Considered as Perceptual Systems*. Houghton Mifflin.

[82] Girin, L., Leglaive, S., Bie, X., Diard, J., Hueber, T., and Alameda-Pineda, X. (2020). Dynamical variational autoencoders: A comprehensive review. *ArXiv*, abs/2008.12595.

[83] GM, H., Gourisaria, M. K., Pandey, M., and Rautaray, S. S. (2020). A comprehensive survey and analysis of generative models in machine learning. *Computer Science Review*, 38:100285.

[84] Golombek, R., Wrede, S., Hanheide, M., and Heckmann, M. (2010). Learning a probabilistic self-awareness model for robotic systems. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2745–2750.

[85] Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Conference on Neural Information Processing Systems*, pages 2672–2680.

[86] Gregory, R. (1974). *Concepts and Mechanisms of Perception*. Duckworth: London.

[87] Grewal, M. S. and Andrews, A. P. (2010). Applications of kalman filtering in aerospace 1960 to the present [historical perspectives]. *IEEE Control Systems*, 30:69–78.

[88] Guha, S., Rastogi, R., and Shim, K. (2000). ROCK: A robust clustering algorithm for categorical attributes. *Information Systems*, 25(5):345–366.

[89] Götzinger, M., Juhász, D., Taherinejad, N., Willegger, E., Tutzer, B., Liljeberg, P., Jantsch, A., and Rahmani, A. M. (2020). Rosa: A framework for modeling self-awareness in cyber-physical systems. *IEEE Access*, 8:141373–141394.

[90] Habibzadeh, F., Habibzadeh, P., and Yadollahie, M. (2016). On determining the most appropriate test cut-off value: the case of tests with continuous results. *Biochemia medica (Zagreb)*, 26(3).

[91] Hajian-Tilaki, K. (2013). Receiver operating characteristic (roc) curve analysis for medical diagnostic test evaluation. *Caspian Journal of Internal Medicine*, 4(2):627–635.

[92] Harootonian, S. K., Ekstrom, A. D., and Wilson, R. C. (2022). Combination and competition between path integration and landmark navigation in the estimation of heading direction. *PLoS Computational Biology*, 18(2):1–26.

[93] Hasan, M., Choi, J., Neumann, J., Roy-Chowdhury, A. K., and Davis, L. S. (2016). Learning temporal regularity in video sequences. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 733–742.

[94] Haykin, S. and Fuster, J. M. (2014). On cognitive dynamic systems: Cognitive neuroscience and engineering learning from each other. *Proceedings of the IEEE*, 102(4):608–628.

[95] Heinze, S., Narendra, A., and Cheung, A. (2018). Principles of insect path integration. *Current Biology*, 28(17):1043–1058.

[96] Hoffmann, H., Jantsch, A., and Dutt, N. D. (2020). Embodied self-aware computing systems. *Proceedings of the IEEE*, 108(7):1027–1046.

[97] Hoffmann, H., Maggio, M., Santambrogio, M. D., Leva, A., and Agarwal, A. (2013). A generalized software framework for accurate and efficient management of performance goals. In *Proceedings of the International Conference on Embedded Software*, pages 1–10.

[98] Hohwy, J. (2013). *The Predictive Mind*. Oxford University Press UK.

[99] Holmström, J. (2002). Examensarbete dv 3 2002-0830 growing neural gas experiments with gng, gng with utility and supervised gng.

[100] Huang, D., Mu, D., Yang, L., and Cai, X. (2018). Codetect: Financial fraud detection with anomaly feature detection. *IEEE Access*, 6:19161–19174.

[101] Huang, Z. and Wu, Y. (2022). A survey on explainable anomaly detection for industrial internet of things. In *IEEE Conference on Dependable and Secure Computing*, pages 1–9.

[102] Iakovidis, D. K., Georgakopoulos, S. V., Vasilakakis, M., Koulaouzidis, A., and Plagianakos, V. (2018). Detecting and locating gastrointestinal anomalies using deep learning and iterative cluster unification. *IEEE Transactions on Medical Imaging*, 37:2196–2210.

[103] Iqbal, H., Campo, D., Baydoun, M., Marcenaro, L., Gomez, D. M., and Regazzoni, C. (2019). Clustering optimization for abnormality detection in semi-autonomous systems. In *1st Int. Workshop on Multimodal Understanding and Learning for Embodied Applications*, page 33–41.

[104] Iqbal, H., Campo, D., Marcenaro, L., Martin Gomez, D., and Regazzoni, C. (2021). Data-driven transition matrix estimation in probabilistic learning models for autonomous driving. *Signal Processing*, 188:108170.

[105] Iqbal, H., Campo, D., Marin-Plaza, P., Marcenaro, L., Gómez, D. M., and Regazzoni, C. (2022). Modeling perception in autonomous vehicles via 3d convolutional representations on lidar. *IEEE Transactions on Intelligent Transportation Systems*, 23(9):14608–14619.

[106] Iqbal, H., Campo, D., Slavic, G., Marcenaro, L., Gómez, D. M., and Regazzoni, C. S. (2020). Optimization of probabilistic switching models based on a two-step clustering approach. In *IEEE Workshop on Signal Processing Systems*, pages 1–6.

[107] Jacobs, R. A. and Shams, L. (2010). Visual learning in multisensory environments. *Topics in Cognitive Science*, 2(2):217–225.

[108] Jiang, Y., Gong, J., Xiong, G., Zhai, Y., Zhao, X., Zhou, S., Jiang, Y., Hu, Y., and Chen, H. (2012). Design of a universal self-driving system for urban scenarios - BIT-III in the 2011 intelligent vehicle future challenge. In *IEEE Intelligent Vehicles Symposium*, pages 506–510.

[109] Jiao, J., Jiao, J., Mo, Y., Liu, W., and Deng, Z. (2019). Magicvo: An end-to-end hybrid CNN and bi-lstm method for monocular visual odometry. *IEEE Access*, 7:94118–94127.

[110] Johnson, M., Duvenaud, D., Wiltschko, A., Adams, R., and Datta, S. (2016). Composing graphical models with neural networks for structured representations and fast inference. In *Conference on Neural Information Processing Systems*, pages 2946–2954.

[111] Jolliffe, I. T. and Cadima, J. (2016). Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2065).

[112] Kalaitzakis, M., Cain, B., Carroll, S., Ambrosi, A., Whitehead, C., and Vitzilaios, N. I. (2021). Fiducial markers for pose estimation. *Journal of Intelligent and Robotic Systems*, 101(4):71.

[113] Kalaitzakis, M., Carroll, S., Ambrosi, A., Whitehead, C., and Vitzilaios, N. (2020). Experimental comparison of fiducial markers for pose estimation. In *International Conference on Unmanned Aircraft Systems*, pages 781–789.

[114] Kalman, R. E. and Others (1960). A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45.

[115] Kanapram, D., Campo, D., Baydoun, M., Marcenaro, L., Bodanese, E., Regazzoni, C., and Marchese, M. (2019). Dynamic bayesian approach for decision-making in ego-things. In *IEEE 5th World Forum on Internet of Things*, pages 909–914.

[116] Karim, F., Majumdar, S., Darabi, H., and Harford, S. (2019). Multivariate lstm-fcns for time series classification. *Neural Networks*, 116:237–245.

[117] Karl, M., Soelch, M., Bayer, J., and van der Smagt, P. (2017). Deep variational bayes filters: Unsupervised learning of state space models from raw data. In *5th International Conference on Learning Representations*.

[118] Ke, N. R., Bilaniuk, O., Goyal, A., Bauer, S., Larochelle, H., Pal, C., and Bengio, Y. (2019). Learning neural causal models from unknown interventions. *arXiv*, abs/1910.01075.

[119] Kendall, A., Grimes, M., and Cipolla, R. (2015). Posenet: A convolutional network for real-time 6-dof camera relocalization. In *IEEE International Conference on Computer Vision*, pages 2938–2946.

[120] Khodairy, M. A. and Abosamra, G. (2021). Driving behavior classification based on oversampled signals of smartphone embedded sensors using an optimized stacked-lstm neural networks. *IEEE Access*, 9:4957–4972.

[121] Kim, J. and Grauman, K. (2009). Observe locally, infer globally: A space-time MRF for detecting abnormal activities with incremental updates. In *Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2921–2928.

[122] Kingma, D. P. and Welling, M. (2014). Auto-encoding variational bayes. In *International Conference on Learning Representations*.

[123] Kingma, D. P. and Welling, M. (2019). An introduction to variational autoencoders. *Foundations and Trends in Machine Learning*, 12(4):307–392.

[124] Kiran, B. R., Thomas, D., and Parakkal, R. (2018). An overview of deep learning based methods for unsupervised and semi-supervised anomaly detection in videos. *Journal of Imaging*, 4(2):36.

[125] Kohonen, T. (1990). The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480.

[126] Kohonen, T. (2001). *Self-Organizing Maps, ser. Physics and astronomy online library*. Springer Berlin Heidelberg.

[127] Koller, D. and Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*. The MIT Press.

[128] Kosak, O., Wanninger, C., Angerer, A., Hoffmann, A., Schiendorfer, A., and Seebach, H. (2016). Towards self-organizing swarms of reconfigurable self-aware robots. In *IEEE 1st International Workshops on Foundations and Applications of Self* Systems*, pages 204–209.

[129] Kotseruba, I. and Tsotsos, J. K. (2020). 40 years of cognitive architectures: core cognitive abilities and practical applications. *Artificial Intelligence Review*, 53(1):17–94.

[130] Kounev, S., Kephart, J. O., Milenkoski, A., and Zhu, X., editors (2017). *Self-Aware Computing Systems*. Springer International Publishing.

[131] Krayani, A., Alam, A. S., Marcenaro, L., Nallanathan, A., and Regazzoni, C. S. (2022). An emergent self-awareness module for physical layer security in cognitive UAV radios. *IEEE Transactions on Cognitive Communications and Networking*, 8(2):888–906.

[132] Krayani, A., Baydoun, M., Marcenaro, L., Alam, A. S., and Regazzoni, C. (2020). Self-learning bayesian generative models for jammer detection in cognitive-uav-radios. In *IEEE Global Communications Conference*, pages 1–7.

[133] Krishnan, R., Shalit, U., and Sontag, D. (2015). Deep kalman filters. *ArXiv*, abs/1511.05121.

[134] Kullback, S. and Leibler, R. A. (1951). On information and sufficiency. *Annals of Mathematical Statistics*, 22(1):79–86.

[135] Kumaran, D. and Maguire, E. A. (2007). Match-mismatch processes underlie human hippocampal responses to associative novelty. *Journal of Neuroscience*, 27(32).

[136] LeCun, Y. (2022). *A Path Towards Autonomous Machine Intelligence.* OpenReview Archive.

[137] LeCun, Y., Chopra, S., Hadsell, R., Ranzato, M., and Huang, F. (2006). A tutorial on energy-based learning.

[138] Lee, S. J., Kim, D., Hwang, S. S., and Lee, D. (2021). Local to global: Efficient visual localization for a monocular camera. In *IEEE Winter Conference on Applications of Computer Vision*, pages 2230–2239.

[139] Leglaive, S., Alameda-Pineda, X., Girin, L., and Horaud, R. (2020). A recurrent variational autoencoder for speech enhancement. *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 371–375.

[140] Lewis, P. R. (2017). Self-aware computing systems: From psychology to engineering. In *Proceedings of the Conference on Design, Automation & Test in Europe*, page 1044–1049. European Design and Automation Association.

[141] Lewis, P. R., Chandra, A., Parsons, S., Robinson, E., Glette, K., Bahsoon, R., Torresen, J., and Yao, X. (2011). A survey of self-awareness and its application in computing systems. In *5th IEEE Conference on Self-Adaptive and Self-Organizing Systems Workshops*, pages 102–107.

[142] Leão, T., Madeira, S. C., Gromicho, M., de Carvalho, M., and Carvalho, A. M. (2021). Learning dynamic bayesian networks from time-dependent and time-independent data: Unraveling disease progression in amyotrophic lateral sclerosis. *Journal of Biomedical Informatics*, 117:103730.

[143] Li, D., Chen, D., Shi, L., Jin, B., Goh, J., and Ng, S. (2019). Mad-gan: Multivariate anomaly detection for time series data with generative adversarial networks. In *International Conference on Artificial Neural Networks*, pages 703–716.

[144] Li, J., Huang, Q., Du, Y., Zhen, X., Chen, S., and Shao, L. (2022a). Variational abnormal behavior detection with motion consistency. *IEEE Transactions on Image Processing*, 31:275–286.

[145] Li, Y. and Mandt, S. (2018). Disentangled sequential autoencoder. In *International Conference on Machine Learning*, pages 5656–5665.

[146] Li, Z., Zhu, Y., and van Leeuwen, M. (2022b). A survey on explainable anomaly detection. *CoRR*, abs/2210.06959.

[147] Lim, H. and Lee, Y. S. (2009). Real-time single camera slam using fiducial markers. In *ICCAS-SICE*, pages 177–182.

[148] Lipton, Z. C. (2018). The mythos of model interpretability. *Communications of the ACM*, 61(10):36–43.

[149] Liu, W., Luo, W., Lian, D., and Gao, S. (2018). Future frame prediction for anomaly detection - a new baseline. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 6536–6545.

[150] Liu, Y., Wang, H., Wang, J., and Wang, X. (2021). Unsupervised monocular visual odometry based on confidence evaluation. *IEEE Transactions on Intelligent Transportation Systems*, 23(6):5387–5396.

[151] Lu, C., Shi, J., and Jia, J. (2013). Abnormal event detection at 150 FPS in MATLAB. In *IEEE International Conference on Computer Vision*, pages 2720–2727.

[152] Lu, Y., Maheshkumar, K., shahabeddin Nabavi, S., and Wang, Y. (2019). Future frame prediction using convolutional vrnn for anomaly detection. In *IEEE International Conference on Advanced Video and Signal Based Surveillance*, pages 1–8.

[153] Luo, W., Liu, W., and Gao, S. (2017a). Remembering history with convolutional LSTM for anomaly detection. In *International Conference on Multimedia and Expo*, pages 439–444.

[154] Luo, W., Liu, W., and Gao, S. (2017b). A revisit of sparse coding based anomaly detection in stacked RNN framework. In *IEEE International Conference on Computer Vision*, pages 341–349.

[155] Mahalanobis, P. C. (1936). On the generalized distance in statistics. *Proceedings of the National Institute of Sciences (Calcutta)*, 2:49–55.

[156] Mahmoud, S., Billing, E., Svensson, H., and Thill, S. (2022). Where to from here? on the future development of autonomous vehicles from a cognitive systems perspective. *Cognitive Systems Research*, 76:63–77.

[157] Marın-Plaza, P., Beltrán, J., Hussein, A., Musleh, B., Martın, D., de la Escalera, A., and Armingol, J. M. (2016). Stereo vision-based local occupancy grid map for autonomous navigation in ros. In *International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, pages 703–708.

[158] Mascaro, S., Nicholson, A., and Korb, K. (2014). Anomaly detection in vessel tracks using bayesian networks. *International Journal of Approximate Reasoning*, 55:84–98.

[159] Medel, J. R. and Savakis, A. (2016). Anomaly detection in video using predictive convolutional long short-term memory networks. *ArXiv*, abs/1612.00390.

[160] Mienye, I. D. and Sun, Y. (2022). A survey of ensemble learning: Concepts, algorithms, applications, and prospects. *IEEE Access*, 10:99129–99149.

[161] Monamo, P., Marivate, V., and Twala, B. (2016). Unsupervised learning for robust bitcoin fraud detection. In *Information Security for South Africa*, pages 129–134.

[162] Morin, A. (2006). Levels of consciousness and self-awareness: A comparison and integration of various neurocognitive views. *Consciousness Cognition*, 15:358–371.

[163] Mráz, E., Rodina, J., and Babinec, A. (2020). Using fiducial markers to improve localization of a drone. In *23rd International Symposium on Measurement and Control in Robotics*, pages 1–5.

[164] Nahangi, M., Heins, A., McCabe, B., and Schoellig, A. (2018). Automated localization of uavs in gps-denied indoor construction environments using fiducial markers. In *Proceedings of the 35th International Symposium on Automation and Robotics in Construction*, pages 88–94.

[165] Nahm, F. S. (2022). Receiver operating characteristic curve: overview and practical use for clinicians. *Korean J Anesthesiol*, 75(1):25–36.

[166] Nguyen, K., Dinh, D., Do, M. D., and Tran, M. (2020a). Anomaly detection in traffic surveillance videos with gan-based future frame prediction. In *ACM International Conference on Multimedia Retrieval*, pages 457–463.

[167] Nguyen, K., Fookes, C., and Sridharan, S. (2020b). Context from within: Hierarchical context modeling for semantic segmentation. *Pattern Recognition*, 105:107358.

[168] Ntalampiras, S., Potamitis, I., and Fakotakis, N. (2011). Probabilistic novelty detection for acoustic surveillance under real-world conditions. *IEEE Transactions on Multimedia*, 13(4):713–719.

[169] Nugroho, K. A. (2018). A comparison of handcrafted and deep neural network feature extraction for classifying optical coherence tomography (oct) images. In *International Conference on Informatics and Computational Sciences*, pages 1–6.

[170] Oprea, S., Martinez-Gonzalez, P., Garcia-Garcia, A., Castro-Vargas, J. A., Orts-Escolano, S., Garcia-Rodriguez, J., and Argyros, A. (2020). A review on deep learning techniques for video prediction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(6):2806–2826.

[171] Oxenham, A. J. (2011). Helmholtz: From enlightenment to neuroscience. *The Journal of Clinical Investigation*, 121(6):2064–2064.

[172] Palacios, P. and Colombo, M. (2021). Non-equilibrium thermodynamics and the free energy principle in biology. *Biology & Philosophy*, 36.

[173] Parr, T., Pezzulo, G., and Friston, K. (2022). *Active Inference: The Free Energy Principle in Mind, Brain, and Behavior*. The MIT Press.

[174] Peng, G., Lu, Z., Chen, S., He, D., and Xinde, L. (2021). Pose estimation based on wheel speed anomaly detection in monocular visual-inertial slam. *IEEE Sensors Journal*, 21(10):11692–11703.

[175] Piasco, N., Sidibé, D., Demonceaux, C., and Gouet-Brunet, V. (2018). A survey on visual-based localization: On the benefit of heterogeneous data. *Pattern Recognition*, 74:90–109.

[176] Pingel, T. J., Clarke, K. C., and McBride, W. A. (2013). An improved simple morphological filter for the terrain classification of airborne lidar data. *ISPRS Journal of Photogrammetry and Remote Sensing*, 77:21–30.

[177] Plato (1943). *The Republic*. New York: Books, Inc.

[178] Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, 1:81–106.

[179] Ramachandra, B., Jones, M. J., and Vatsavai, R. R. (2022). A survey of single-scene video anomaly detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(5):2293–2312.

[180] Ravanbakhsh, M., Baydoun, M., Campo, D., Marín, P., Martín, D., Marcenaro, L., and Regazzoni, C. (2020). Learning self-awareness for autonomous vehicles: Exploring multisensory incremental models. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–15.

[181] Ravanbakhsh, M., Sangineto, E., Nabi, M., and Sebe, N. (2019). Training adversarial discriminators for cross-channel abnormal event detection in crowds. In *IEEE Winter Conference on Applications of Computer Vision*, pages 1896–1904.

[182] Regazzoni, C. S. (2021). Bayesian emergent self awareness. In *IEEE International Conference on Autonomous Systems*, pages 1–1.

[183] Regazzoni, C. S., Marcenaro, L., Campo, D., and Rinner, B. (2020). Multisensorial generative and descriptive self-awareness models for autonomous systems. *Proceedings of the IEEE*, 108(7):987–1010.

[184] Ribeiro, I. (2004). Kalman and extended kalman filters concept, derivation and properties.

[185] Rivera, A. R., Khan, A., Bekkouch, I. E. I., and Sheikh, T. S. (2020). Anomaly detection based on zero-shot outlier synthesis and hierarchical feature distillation. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–11.

[186] Robinson, J. W. and Hartemink, A. J. (2008). Non-stationary dynamic bayesian networks. In *Advances in Neural Information Processing Systems 21, Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems*, pages 1369–1376.

[187] Rodrigues, R., Bhargava, N., Velmurugan, R., and Chaudhuri, S. (2020). Multi-timescale trajectory prediction for abnormal human activity detection. In *IEEE Winter Conference on Applications of Computer Vision*, pages 2615–2623.

[188] Rodriguez, I., Astigarraga, A., Ruiz, T., and Lazkano, E. (2017). Body self-awareness for social robots. In *International Conference on Control, Artificial Intelligence, Robotics and Optimization*, pages 69–73.

[189] Romera, E., Bergasa, L. M., and Arroyo, R. (2015). A real-time multi-scale vehicle detection and tracking approach for smartphones. In *IEEE 18th International Conference on Intelligent Transportation Systems*, pages 1298–1303.

[190] Romera, E., Bergasa, L. M., and Arroyo, R. (2016). Need data for driver behaviour analysis? presenting the public uah-driveset. In *19th IEEE International Conference on Intelligent Transportation Systems*, pages 387–392.

[191] Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215.

[192] Russell, S. and Norvig, P. (2010). *Artificial Intelligence: A Modern Approach*. Prentice Hall, 3 edition.

[193] Sabokrou, M., Fayyaz, M., Fathy, M., Moayed, Z., and Klette, R. (2018). Deep-anomaly: Fully convolutional neural network for fast anomaly detection in crowded scenes. *Computer Vision and Image Understanding*, 172:88–97.

[194] Saleh, K., Hossny, M., and Nahavandi, S. (2017). Driving behavior classification based on sensor data fusion using lstm recurrent neural networks. In *IEEE 20th International Conference on Intelligent Transportation Systems*, pages 1–6.

[195] Salotti, J. M. (2018). Bayesian network for the prediction of situation awareness errors. *International Journal of Human Factors Modelling and Simulation*, 6:119–126.

[196] Sarafijanovic-Djukic, N. and Davis, J. (2019). Fast distance-based anomaly detection in images using an inception-like autoencoder. In *International Conference on Discovery Science*, pages 493–508.

[197] Saypadith, S. and Onoye, T. (2021). An approach to detect anomaly in video using deep generative network. *IEEE Access*, 9:150903–150910.

[198] Schölkopf, B. (2019). Causality for machine learning. *arXiv*, abs/1911.10500.

[199] Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D. (2020). Grad-cam: Visual explanations from deep networks via gradient-based localization. *International Journal of Computer Vision*, 128(2):336–359.

[200] Seth, A. (2021). *Being You: A New Science of Consciousness*. Faber & Faber.

[201] Shi, X., Chen, Z., Wang, H., Yeung, D., Wong, W., and Woo, W. (2015). Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems*, pages 802–810.

[202] Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *3rd International Conference on Learning Representations*, pages 1–14.

[203] Singh, A., Jones, M. J., and Learned-Miller, E. G. (2023). EVAL: explainable video anomaly localization. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18717–18726.

[204] Song, H., Sun, C., Wu, X., Chen, M., and Jia, Y. (2020). Learning normal patterns via adversarial attention-based autoencoder for abnormal event detection in videos. *IEEE Transactions on Multimedia*, 22(8):2138–2148.

[205] Sosa, A. J., Fontaine, Y. J., Hengst, P. T., Jezior, B. A., Lasher, W. T., Motsek, G. J., and Wells, B. E. (1998). Ay 97 compendium army after next project. Technical report, Strategic Studies Institute, US Army War College.

[206] Spera, E., Furnari, A., Battiato, S., and Farinella, G. M. (2018). Egocentric shopping cart localization. In *International Conference on Pattern Recognition*, pages 2277–2282.

[207] Spera, E., Furnari, A., Battiato, S., and Farinella, G. M. (2021). Egocart: a benchmark dataset for large-scale indoor image-based localization in retail stores. *IEEE Transactions on Circuits and Systems for Video Technology*, 31:1253–1267.

[208] Subagdja, B. and Tan, A. (2017). Towards a brain inspired model of self-awareness for sociable agents. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 4452–4458.

[209] Sucar, L. E. (2015). Probabilistic graphical models. *Advances in Computer Vision and Pattern Recognition*, 10:978–1.

[210] Sun, C., Jia, Y., Hu, Y., and Wu, Y. (2020). Scene-aware context reasoning for unsupervised abnormal event detection in videos. In *28th ACM International Conference on Multimedia*, pages 184–192.

[211] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S. E., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9. IEEE Computer Society.

[212] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826.

[213] Szymanowicz, S., Charles, J., and Cipolla, R. (2021). X-MAN: explaining multiple sources of anomalies in video. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 3224–3232.

[214] Szymanowicz, S., Charles, J., and Cipolla, R. (2022). Discrete neural representations for explainable anomaly detection. In *IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1506–1514.

[215] Talebloo, F., Mohammed, E. A., and Far, B. H. (2021). Deep learning approach for aggressive driving behaviour detection. *CoRR*, abs/2111.04794.

[216] Tong, A., Wolf, G., and Krishnaswamy, S. (2020). Fixing bias in reconstruction-based anomaly detection with lipschitz discriminators. In *IEEE International Workshop on Machine Learning for Signal Processing*.

[217] Tran, H. and Hogg, D. (2017). Anomaly detection using a convolutional winner-take-all autoencoder. In *British Machine Vision Conference*, pages 139.1–139.12.

[218] Valada, A., Radwan, N., and Burgard, W. (2018). Deep auxiliary learning for visual localization and odometry. In *IEEE International Conference on Robotics and Automation*, pages 6939–6946.

[219] Vondrick, C. and Torralba, A. (2017). Generating the future with adversarial transformers. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2992–3000.

[220] Walch, F., Hazirbas, C., Leal-Taixé, L., Sattler, T., Hilsenbeck, S., and Cremers, D. (2017). Image-based localization with spatial lstms. In *IEEE International Conference on Computer Vision*.

[221] Wan, E. A. and van der Merwe, R. (2000). The unscented kalman filter for nonlinear estimation. *IEEE Adaptive Systems for Signal Processing, Communications, and Control Symposium*, pages 153–158.

[222] Wang, H., Bah, M. J., and Hammad, M. (2019). Progress in outlier detection techniques: A survey. *IEEE Access*, 7:107964–108000.

[223] Wang, S., Clark, R., Wen, H., and Trigoni, N. (2017). Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks. In *IEEE International Conference on Robotics and Automation*, pages 2043–2050.

[224] Wang, X., Che, Z., Jiang, B., Xiao, N., Yang, K., Tang, J., Ye, J., Wang, J., and Qi, Q. (2022). Robust unsupervised video anomaly detection by multipath frame prediction. *IEEE Transactions on Neural Networks and Learning Systems*, 33(6):2301–2312.

[225] Watter, M., Springenberg, J. T., Boedecker, J., and Riedmiller, M. (2015). Embed to control: A locally linear latent dynamics model for control from raw images. In *Conference on Neural Information Processing Systems*, pages 2746–2754.

[226] Wei, H., Li, K., Li, H., Lyu, Y., and Hu, X. (2019). Detecting video anomaly with a stacked convolutional lstm framework. In *International Conference on Computer Vision Systems*, pages 330–342.

[227] Wu, B., Tian, H., Yan, X., and Guedes Soares, C. (2020). A probabilistic consequence estimation model for collision accidents in the downstream of yangtze river using bayesian networks. *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of risk and reliability*, 234(2):422–436.

[228] Wu, C. (2013). Towards linear-time incremental structure from motion. In *International Conference on 3D Vision*, pages 127–134.

[229] Xiong, L., Kang, R., Zhao, J., Zhang, P., Xu, M., Ju, R., Ye, C., and Feng, T. (2022). G-vido: A vehicle dynamics and intermittent gnss-aided visual-inertial state estimator for autonomous driving. *IEEE Transactions on Intelligent Transportation Systems*, 23(8):11845–11861.

[230] Xu, B., Li, S., Razzaqi, A. A., Guo, Y., and Wang, L. (2021). A novel measurement information anomaly detection method for cooperative localization. *IEEE Transactions on Instrumentation and Measurement*, 70:1–18.

[231] Xue, F., Wang, X., Yan, Z., Wang, Q., Wang, J., and Zha, H. (2019a). Local supports global: Deep camera relocalization with sequence enhancement. In *IEEE/CVF International Conference on Computer Vision*, pages 2841–2850.

[232] Xue, Q., Wang, K., Lu, J. J., and Liu, Y. (2019b). Rapid driving style recognition in car-following using machine learning and vehicle trajectory data. *Journal of Advanced Transportation*, 2019.

[233] Yan, S., Smith, J., Lu, W., and Zhang, B. (2020). Abnormal event detection from videos using a two-stream recurrent variational autoencoder. *IEEE Transactions on Cognitive and Developmental Systems*, 12:30–42.

[234] Yi, D., Su, J., Liu, C., Quddus, M., and Chen, W.-H. (2019). A machine learning based personalized system for driving state recognition. *Transportation Research Part C: Emerging Technologies*, 105:241–261.

[235] Yousif, K., Bab-Hadiashar, A., and Hoseinnezhad, R. (2015). An overview to visual odometry and visual slam: Applications to mobile robotics. *Intelligent Industrial Systems*, 1:289–311.

[236] Yu, F., Koltun, V., and Funkhouser, T. (2017). Dilated residual networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 636–644.

[237] Yuan, S., Wang, H., and Xie, L. (2021). Survey on localization systems and algorithms for unmanned systems. *Unmanned Systems*, 9(2):129–163.

[238] Yurtsever, E., Lambert, J., Carballo, A., and Takeda, K. (2020). A survey of autonomous driving: Common practices and emerging technologies. *IEEE Access*, 8:58443–58469.

[239] Zaal, H., Baydoun, M., Marcenaro, L., Tokarchuk, L. N., and Regazzoni, C. S. (2019). Abnormality detection using graph matching for multi-task dynamics of autonomous systems. *16th IEEE International Conference on Advanced Video and Signal Based Surveillance*, pages 1–8.

[240] Zeiler, M. D. and Fergus, R. (2014). Visualizing and understanding convolutional networks. In *European Conference on Computer Vision*, pages 818–833.

[241] Zeng, X., Jiang, Y., Ding, W., Li, H., Hao, Y., and Qiu, Z. (2023). A hierarchical spatio-temporal graph convolutional neural network for anomaly detection in videos. *IEEE Transactions on Circuits and Systems for Video Technology*, 33(1):200–212.

[242] Zhang, L., Lu, L., Wang, X., Zhu, R., Bagheri, M., Summers, R. M., and Yao, J. (2020a). Spatio-temporal convolutional lstms for tumor growth prediction by learning 4d longitudinal patient data. *IEEE Trans. Medical Imaging*, 39(4):1114–1126.

[243] Zhang, S., Pöhlmann, R., Wiedemann, T., Dammann, A., Wymeersch, H., and Hoeher, P. A. (2020b). Self-aware swarm navigation in autonomous exploration missions. *Proceedings of the IEEE*, 108(7):1168–1195.

[244] Zhang, X., Fang, J., Yang, B., Chen, S., and Li, B. (2023). Hybrid attention and motion constraint for anomaly detection in crowded scenes. *IEEE Transactions on Circuits and Systems for Video Technology*, 33(5):2259–2274.

[245] Zhao, Y., Deng, B., Shen, C., Liu, Y., Lu, H., and Hua, X. (2017). Spatio-temporal autoencoder for video anomaly detection. In *ACM on Multimedia Conference*, pages 1933–1941.

[246] Zhou, S., Shen, W., Zeng, D., Fang, M., Wei, Y., and Zhang, Z. (2016). Spatial-temporal convolutional neural networks for anomaly detection and localization in crowded scenes. *Signal Processing Image Communication*, 47:358–368.

[247] Zhuang, H., Jiang, Y., Cheng, Y., Chu, H., Gao, B., and Chen, H. (2022). Long-time driving style recognition using gated recurrent unit with attention. In *6th CAA International Conference on Vehicular Control and Intelligence*, pages 1–6.