Supporting humans with autonomous systems: deep learning for activity, state and environment recognition

Giacomo Dabisias

Supporting humans with autonomous systems: deep learning for activity, state and environment recognition

Perceptual Robotics Laboratory, Tecip

Scuola Superiore Sant'Anna

Author:

Giacomo Dabisias

Supervisor:

Emanuele Ruffaldi

Ph.D Course
Perceptual Robotics

Academic Year
2016 / 2017

March 12th, 2018

Abstract

Autonomous Systems can support human activities in several situations, ranging from daily tasks to specific working activities. All these systems have in common the need of understanding their environment and the state of the human interacting with them. Once such information has been assessed by the system, it can either perform autonomously actions, suggest them or simply present additional information about the activity or environment to the user.

It is necessary to consider that different activities need different levels of confidence in the decision making process. Critical systems, such as vehicles in autonomous driving scenarios, need the highest possible accuracy given that they take autonomous decisions; on the other hand, critical safe systems, need a lower level of confidence given that they can just provide a feedback to the user to ease the decision making process.

It is also important to notice that Autonomous Systems interact usually with 3D environments while sensors acquire mostly 2D images. This requires the ability to reconstruct precisely the surrounding environment from multiple views using triangulation techniques.

All this brings up several challenges given the high variability of both activities, environments and people, making traditional computer vision approaches less adequate. This is due also to the fact that often it is not possible to identify clearly the input variables of the system, given the high correlation between them or the high dimensionality of the input space. Machine learning has shown promising results in such scenarios and in particular deep learning is the evolution of machine learning that has shown most effective results in terms of quality and performance of the learning tasks. Deep Learning can cope well with variable scenarios by scaling to highly dimensional decision spaces that typically suffer the problem of feature identification and selection.

This work will show some developed applications in activity, state and environment recognition, presenting how human decisions can be supported by autonomous systems using deep learning techniques. The first part of the thesis will present the state of the art solutions to the aforementioned problems along with the latest deep learning techniques. In the second part

of this work, we will describe in depth three different developed applications in activity, state and environment recognition. Finally we will present possible future works along with remaining open research questions.

Acknowledgments

Firstly I would like to thank my tutor Emanuele for all the time he spent with me discussing, researching and teaching me. It's because of him if I can finally write this chapter ending this journey. He was not only my tutor, but also a mentor and friend during all these years, helping me with good advice in every situation.

Thanks Filippo for completing with me all our studies. We started together the bachelor and are now together at the end of our PhD. Meeting you at the first year was the best possible thing which could happen.

Thanks to everyone at Percro for making my time more relaxed every day; for the rubber band wars with Massimo and Paolo, Peppolonis' inside stories, the Mexican tequila with Juan, the technical discussions with Alessandro Filippeschi, the great lunches with Maddaloni and for the friendship with everyone else.

Thanks Conte Landolfi for your everyday wisdom.

A special thanks goes to Michel Sainville, the best house mate one could ever imagine, with whom I grew everyday in the last three years discussing, laughing and crying.

Thanks to my girlfriend Sofia, who is supporting me in everyday life with happiness and joy.

Thanks to all my friend that are dearest to me; Sara, Stella, Luca F., Luca G., Marco, Mattia, Francesca Elisa, Leonardo, Alberto, Sofiya, Laura and all the others: you are the best friends one could ever imagine and you have always helped me during my academic years.

I would like to thank also my entire family for the continuous support towards this goal, was it with good advice or amazing food.

A special thanks goes to all the team of Blue Vision Labs were I spent my internship. I did not only learn an incredible amount of new things, but I also made a lot of dear friends. Thanks again Hugo, Peter, Lukas, Ross, Michal, Filip S., Filip H., Suraj, Robert, Guido, Ivan and all the others!

To my grandfather

Contents

List of Figures

List of Tables

Acronyms

AS Autonomous System

AR Activity Recognition

ML Machine Learning

NN Neural Network

DL Deep Learning

DNN Deep Neural Network

CNN Convolutional Neural Network

RNN Recurrent Neural Network

R-CNN Recurrent Convolutional Neural Network

DBN Deep Belief Networks

MCI Mild Cognitive Impairments

PELARS Practice-based Experiential Learning Analytics Research and Support

RAMCIP Robotic Assistant for MCI Patients at home

VALUE Large Scale Voting-based Automatic Labelling for Urban Environments

MMSE WS Mini Mental State Examination Weighted Sum Score

AD Alzheimer's Disease

ROS Robot Operating System

CoM Center of Mass

MMLA Multi Modal Learning Analytics

GPU Graphics Processing Unit

PAF Part Affinity Fields

ILP Integer Linear Programming

ASIC Application Specific Integrated Circuit

FPGA Field Programmable Gate Array

PBL Project Based Learning

RA Reference Axis

RF Reference Frame

Chapter 1

Introduction

> "If you look at the field of robotics today, you can say robots have been in the deepest oceans, they've been to Mars, you know? They've been all these places, but they're just now starting to come into your living room. Your living room is the final frontier for robots."
>
> Cynthia Breazeal, [110]

We are starting to live exactly at the edge of time in which Robots are coming to our living rooms.

Autonomous Systems (AS) are becoming an ubiquitous reality in modern day society. In general, an AS can be defined as a robot or system that performs behaviors or tasks with a high degree of autonomy. The main goal of such systems is to improve the overall human quality of life in all possible aspects; from self driving vehicles to assistive robots. Usually robotic systems are triggered by human actions, acting on the inputs of a user. ASs instead shift this paradigm, making robots capable of anticipating human needs and actions. This implies that ASs have to be able to take decisions based on the performed activity, the general state of the person interacting with it and the surrounding environment. To do this, ASs need to have some kind of intelligence which has been studied deeply in the field of Machine Learning (ML).

ML, according to Arthur Samuel, gives "computers the ability to learn without being explicitly programmed." The term was coined by him in 1959 while at IBM. Despite being already studied for several decades, ML had only lately a major success with the rediscovery of Neural Networks (NN). Warren McCulloch and Walter Pitts (1943) created a computational model for NNs based on mathematics and algorithms called threshold logic. Since then, NNs had some minor success at character recognition in 1990 when they were used to recognise automatically recipients of post correspondence and digits on cheques. After that, NNs were not investigated much given that the high computational load could not be processed in a reasonable amount of

time on present calculators. This changed recently with the advances in GPU hardware and software and at the same time with the research advances in Deep Learning (DL) using Deep Neural Networks (DNN).

DL was introduced to the ML community by Rina Dechter in 1986 and got a major success in 2006 with a publication by Hinton et al. [69]. Since then DNN have been used to investigate several branches of computer science, creating many new fields of research and making autonomous systems a viable possibility.

Recent GPU's have started to target ML computation, DNNs more precisely, implementing dedicated hardware to be able to overcome the computational problem which was afflicting them. Given that quickly GPUs were able to cope with the power demand of simple NNs, more complex networks have been created, which were able to solve complex challenges, as image recognition and detection, to a certain degree of precision. Almost human performance has been reached in the last years thanks to the discovery of Convolutional Neural Networks (CNN), which gave autonomous systems new capabilities, making the interaction between humans and machines almost natural.

Thanks to these advancements, we are starting to have the first prototypes of fully autonomous vehicles and robots, which brings up a series of challenges in the interaction between human and machine.

1.1 Research aims and questions

The aim of this thesis is to investigate the ability of ASs to detect an activity (Research Aim 1 **(RA1)**, evaluate the state of the person interacting with the AS (Research Aim 2 **(RA2)** and perceive their environment (Research Aim 3 **(RA3)** through modern ML techniques.

Source data is usually noisy, given that it comes from unconstrained real world scenarios. We would like to avoid the filtering of it given that usually the noise input model is unknown. ML has proven very effective at extracting strong features from input streams, making the task simpler.

This leads to the following research questions:

Research Question 1 (RQ1): Is it possible for autonomous systems to extract high level information from a system represented by an ensemble of noisy sources of data?

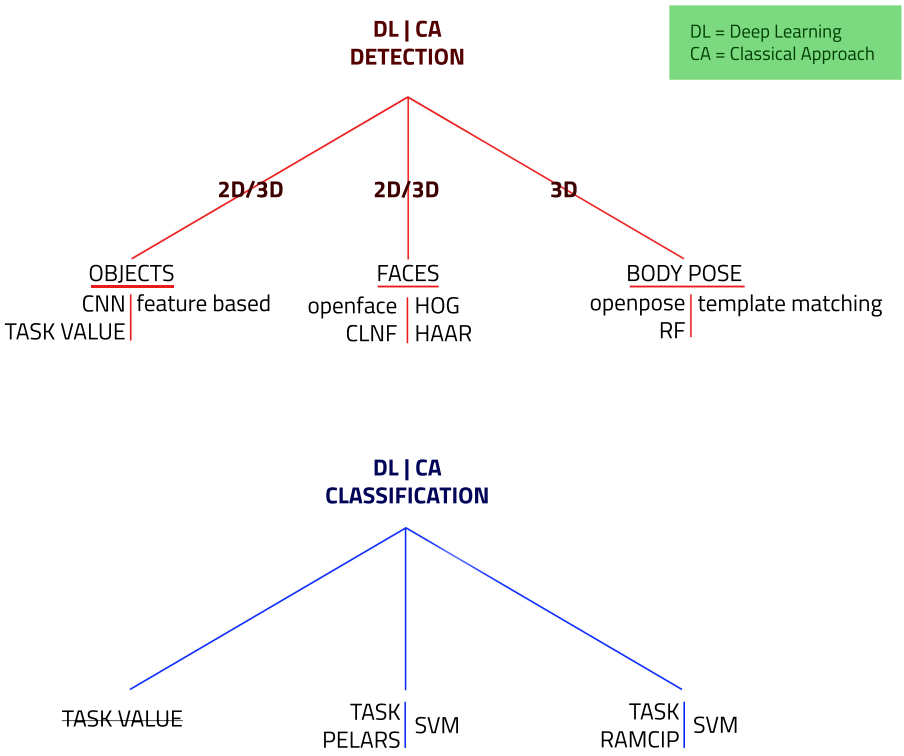


FIGURE 1.1: General overview of the Detection and Classification split analysed in this work. Each computer vision activity performed by an AS can be split in Classification and Detection. In each subfigure we present on the left side of each bar the available ML solutions and the task involved in the assessment while on the right side we present the classical approach.

Each of the described tasks can be split in **Detection** and **Classification**, see figure 1.1. Detection deals with the extraction of particular information from a larger stream of information while Classification deals with recognizing, differentiating and understanding ideas and objects. In this work Detection is used to extract semantic content from an observed scene, which can be then processed and utilised by a Classification system in order to make decisions and take actions. This leads us to the question:

Research Question 2 (RQ2): What kind of ML technique is able to solve the Detection/Classification problem in the different explained scenarios?

Given that this work focuses on the interaction between humans and machines, we will analyse three different categories of detection: Objects, Faces and Body poses. For each of them we want to answer the following questions:

Research Question 3 (RQ3): Is it possible for autonomous systems to detect with sufficient precision objects?

Research Question 4 (RQ4): Is it possible for autonomous systems to detect, with sufficient precision faces?

Research Question 5 (RQ5): Is it possible for autonomous systems to detect, with sufficient precision body poses?

It is important to notice also that all the depicted interactions between AS and humans are happening in 3D space. This poses a series of problems and questions regarding the possibility of inferring precisely a 3D position given 2D information. Nowadays 3D sensors are available, which produce depth images of space, but are more complex to interact with and have a series of limitations, which are not present in common 2D cameras. This brings us to the question:

Research Question	*Constributions*	*Focus*
RQ1	**C1, C3**	Scene Understanding
RQ2	**C1, C2, C3**	Machine Learning
RQ3	**C1, C3**	Object Detection
RQ4	**C1**	Face detection
RQ5	**C2**	Pose estimation
RQ6	**C3, C4**	Triangulation techniques

TABLE 1.1: Mapping of RQs onto Contributions

Research Question 6 (RQ6): Is it possible to reconstruct, accurately enough for an ASs, 3D information given a series of 2D measurements, or is a 3D sensors necessary for the interaction between AS and Humans?

1.2 Contributions

To answer the aforementioned research questions we developed three different systems. Each of these system will answer partially the proposed research questions as depicted in Table 1.1.

C1: The first system has been developed to create an autonomous system capable of recognizing activities. This has been done by recognizing the actions of students during different hands on learning activities.

C2: The second system consists of an autonomous robot interacting with elderly patients. In this context, we researched the possibility of extracting non invasive measures of the state of the patient, using a 3D sensor. This was done to adapt the behavior of the robot to suit best the needs of the patient.

C3: The third system has been developed to test the ability of an autonomous system to augment a urban environment. In this scenario we developed specifically a system capable of detecting and 3D localising traffic lights for autonomous driving.

C4: Additionally, to test the possibility of using 2D data to interact with a 3D environment, we evaluated different triangulation techniques which are present in all three aforementioned systems

1.3 Involved Projects

Several projects have contributed to the research developed in this work. This section briefly introduces them.

- **Practice-based Experiential Learning Analytics Research and Support** (PELARS) is a European project that studies how people learn about science, technology and mathematics when using their hands as well as their heads. A big part of the project is making more explicit the implicit practices of science teachers: "Lab demos" and hands-on experiments have been a big part of science teaching for as long as anyone can remember, but how to model and analyse these practice, while empowering teachers, is far less understood. So, the PELARS project aims at finding ways of generating "analytics" (data about the learning process and analysis of this data), which helps learners and teachers by providing feedback from hands-on, project-based and experiential learning situations. This project will be discussed in detail in Chapter 6.

- **Robotic Assistant for MCI Patients at home** (RAMCIP) is a European Project which aims to research and develop real robotic solutions for assistive robotics for the elderly and those suffering from Mild Cognitive Impairments (MCI) and dementia. This is a key step to developing a wide range of assistive technologies. We will adopt existing technologies from the robotics community, fuse those with user-centred design activities and practical validation, trying to create a step-change in robotics for assisted living. This project will be discussed in detail in Chapter 7.

- **Large Scale Voting-based Automatic Labelling for Urban Environments** (VALUE). We developed a system capable of recognizing 3D contents in a urban environment in order to add them to a semantic map that can be used by autonomous agents to navigate it. Contents are firstly detected in 2D iamges and then views of the same content are grouped together in order to produce its 3D location in space. This project was developed jointly with Blue Vision Labs (BVL), London, UK. This project will be discussed in detail in Chapter 8.

1.4 Thesis Structure

This thesis is organised as follows:

In **Chapter 2** we will present the state of the art in ASs and in classical detection systems. **Chapter 4** and **Chapter 5** will analyse in depth the available ML solutions to the presented problems. In **Chapter 6** we present the PELARS system along with its results, which will demonstrate the ability of an AS to do Activity Recognition (AR). After this, we will show in **Chapter 7** an example of non invasive extraction of human state using a 3D sensor. In **Chapter 8** we discuss the construction and evaluation of a system capable of augmenting 3D maps. Finally in **Chapter 9** we will discuss the achieved results and present possible future work.

Chapter 2

State of the Art

2.1 History

The idea of AS dates back several centuries, starting with early Greek myths of Hephaestus and Pygmalion who include concepts of animated statues or sculptures [31]. The first automatic devices were called "automata", which is defined as a self-operating machine or control mechanism designed to automatically follow a predetermined sequence of operations, or respond to predetermined instructions.

There are other examples in ancient China, for example in the Lie Zi text, written in the 3rd century BC. It contains a description of a much earlier encounter between King Mu of Zhou (1023-957 BC) and a mechanical engineer known as Yan Shi, an 'artificer'.

Later on, around the 8th century, we find the first wind powered automata, which were defined as "statues that turned with the wind over the domes of the four gates and the palace complex of the Round City of Baghdad". We still can't speak of autonomous systems given the complete lack of decision making and human interaction.

First simple ASs can be found in 1206 when Al-Jazari described complex programmable humanoid automata, which he designed and constructed in the *Book of Knowledge of Ingenious Mechanical Devices*. One example of an early AS was a boat with four automatic musicians that floated on a lake to entertain guests at royal drinking parties. The mechanism had a programmable drum machine with pegs (cams) that bump into little levers that operate the percussion. It was possible to play different rhythms and drum patterns if the pegs were moved around.

During the Reneissance the studies of automata witnessed a considerable revival. Numerous clockworks were built in the 16th century, mainly by the goldsmiths of central Europe. Leonardo da Vinci started sketching and building hundreds of automatic machines, making the interest in such devices grow in the next centuries.

All these examples can not be regarded as ASs given that they were still mainly mimicking human and were mainly considered objects of art than of engineering. In more recent history, a new field of science was born, Cybernetics, which filled the gap between automatas and ASs . Norbert Wiener defined cybernetics in 1948 as "the scientific study of control and communication in the animal and the machine" [167]. He was a mathematics professor at MIT working to develop automated rangefinders for anti-aircraft guns with “intelligent” behavior [68].

This theory motivated the first generation of ASs research in which simple sensors and effectors were combined with analog control electronics to create systems that could demonstrate a variety of interesting reactive behaviors. In 1964 one of the first AS was built by the APL Adaptive Machines Group, led by Leonard Scheer at the John Hopkins University. They built an autonomous rover system capable of navigating the APL's hallways, identifying objects as electrical outlets in the walls, which it could use to plug itself in to recharge its battery, see Figure 2.1.

FIGURE 2.1: An image showing the autonomous rover built by the APL group in 1964[1].
[1] http://cyberneticzoo.com/tag/autonomous/

With the advent of digital electronic controllers and the new born interest in artificial intelligence [166, 56], more complex autonomous systems were built. The new systems were used in several different fields, but mainly for military purpose initially. We have many examples in several domains as maritime, air, ground and space vehicles [156].

In modern days, ASs became widespread and a series of common subproblems were categorized. Each AS needs to self maintain itself, sense and navigate the environment, perform tasks, learn from its history.

2.2 Self Maintenance

A fundamental requirement for complete physical autonomy is the ability of a robot to be aware of its internal state. This ability is called "proprioception". Many of the commercial robots available in the market today can find and connect autonomously to a charging station, like Sony's Aibo (Figure 2.2a) or Ugobe's Pleo (Figure 2.2b).

(A) (B)

FIGURE 2.2: Example of two robots capable of connecting autonomously to a charging station. Figure (a) shows Sony's Aibo[1] and Figure (b) shows Ugobe's Pleo[2].
[1] http://www.sony-aibo.com/
[2] http://www.pleoworld.com/

In the battery charging example, the robot can tell proprioceptively that its batteries are low, and it then seeks the charger. Other examples of proprioceptive sensors are thermal, optical and haptic, as well as those measuring the Hall effect (electric) [63]. These abilities will be required more and more for robots in order to work autonomously near people and in harsh environments. Autonomous rovers used to explore extraterrestrial planets need often to work autonomously for several years, without the possibility of any physical maintenance [159].

This brought to the development of even more advanced systems capable of self repairing [52]. In this case the system is able not only to asses an incorrect self state, but also to remedy it by taking appropriate actions. It is important to notice that this kind of failures can affect both hardware and software.

2.3 Sensing and Navigating

Autonomous systems need to sense the environment in order to navigate it [93, 42]. To do this they are usually equipped with a set of sensors, which allow them to perceive the space surrounding them. The most common sensors are 2D cameras and depth sensors [9]. Both are usually used to

construct a 3D map of the explored space in order to avoid obstacles and to map unknown areas, Figure 2.3.

FIGURE 2.3: An example of 3D mapped environment from the IROS 2014 Challenge[1].
[1] https://github.com/introlab/rtabmap/wiki/IROS-2014-Kinect-Challenge

Examples of other sensors used to understand the environment are: electromagnetic spectrum, sound, touch, smell, temperature and altitude.

Within the depth sensors, a great variety of models is available, from commercial ones as the Kinect v1 and Kinect v2 [164] to more precise models like laser scanners. The first ones are less accurate (v1 $\sim$ 1.5 mm at 50 cm. About 5 cm at 5 m), but cheap, while laser scanners are much more precise (<1mm), but are more expensive. It is also important to notice that the different systems have different update frequencies. This it not relevant if static scenes are scanned, but it can cause problems when fast moving objects are scanned. While commercial sensors have frequencies between 30Hz and 60Hz, laser scanners can reach over 1KHz for single scanned line.

Many ASs are hence equipped with stereo cameras, which are also capable of inferring the depth. These cameras have the advantage that they are usually cheaper, don't need specialised software and have more software support. We will analyse in a Section 8.2 the advantages and disadvantages of 2D sensors against depth sensors.

2.4 Performing tasks

Once the environment has been sensed, the ASs need to take actions in order to fulfill a certain objective. To do this, the AS need to be able to take conditional decision, which are usually determined by some machine learning algorithm [120].

An example of such a system can be found i the Cataglyphis rover. This robot has demonstrated, during the final NASA Sample Return Robot Centennial Challenge in 2016, fully autonomous navigation, decision-making, sample detection, retrieval, and return capabilities. The rover relied on a fusion of measurements from inertial sensors, wheel encoders, Lidar, and camera for navigation and mapping.

More common examples of self adapting systems can be found for example in robotic lawn mowers [45], which can adapt their programming by detecting the speed at which grass grows or some vacuum cleaning robots [137], which are able to sense how much dirt is being picked up and use this information to plan the amount of time spent in different areas.

Usually these systems try to maximize an objective score given by an objective function. Classically this could be done using calculus trying to minimize a cost function or maximizing an objective score. Nowadays Reinforcement Learning (RL) is used to train directly an AS by examples without having the need to define an objective function, but by just defining an objective score. This made it possible to train easily non linear function which gave promising results lately [101].

Very often ASs need to manipulate objects. To do this they need to firstly identify an object in space and then compute the objects pose. A pose is defined by 6 degrees of freedom, 3 for rotation and 3 for translation. A classically approach to solve this problem consists in the use of descriptors, which can be seen as strong invariant features of an object. Techniques which are based on this have the disadvantage of being influenced strongly by external factors as lighting conditions. Nowadays DL techniques are used to overcome the problems which come from having different environments. These techniques are able to find stronger features which are no man crafted, adapting better to real world scenarios. We will present in the next chapter the classical approaches and in Chapter 4 the more recent techniques.

Chapter 3

Classic Detection

> "There is nothing more deceptive than an obvious fact."
>
> Arthur Conan Doyle, [40]

3.1 3D Object Detection

This chapter will introduce the state of the art in detection algorithms not based on ML, but on classical computer vision techniques. The general pipeline for a 3D object recognition algorithm, which identifies a model in a scene, consists of the following steps:

- Preprocessing of the depth and RGB images. This step is used to filter the image from noise and unnecessary information. The two images are then merged together to create a point cloud. Segmenting and filtering objects by color can improve significantly performances and should always be done when possible.
- Extraction of keypoints from the model and the scene. Keypoints are point of interest that carry a big amount of information given their position and color.
- Creation of a descriptor for each keypoint. Given that keypoints could change from the model to the scene due to light, different scales and deformations, a descriptor is used to describe each keypoint. In general a descriptor consists of a set of attributes, which are scale invariant and flexible to deformation and noise, of the keypoint and its surrounding.
- Matching of scene and model descriptors. The general strategy consists in calculating a distance between each descriptor of the model and the scene. If the distance is lower than a certain threshold then the two descriptors are matched.
- Clustering of the found correspondences between descriptors. This step is used to avoid sparse correspondences which are of no interest.

The results are clustered since a good match between the model and the scene is found only when a big number of correspondences is found in a relative compact space.

- Examine a plausible solution and estimate a possible object pose. All the clusters are evaluated using some voting scheme and, if good enough ,a possible pose is estimated using the previously found correspondences.

The first three steps have to be performed on the model and on the scene, Figure 3.1. It is important to notice that they are computed only the first time on the model and have to be recomputed for each new scene point cloud.

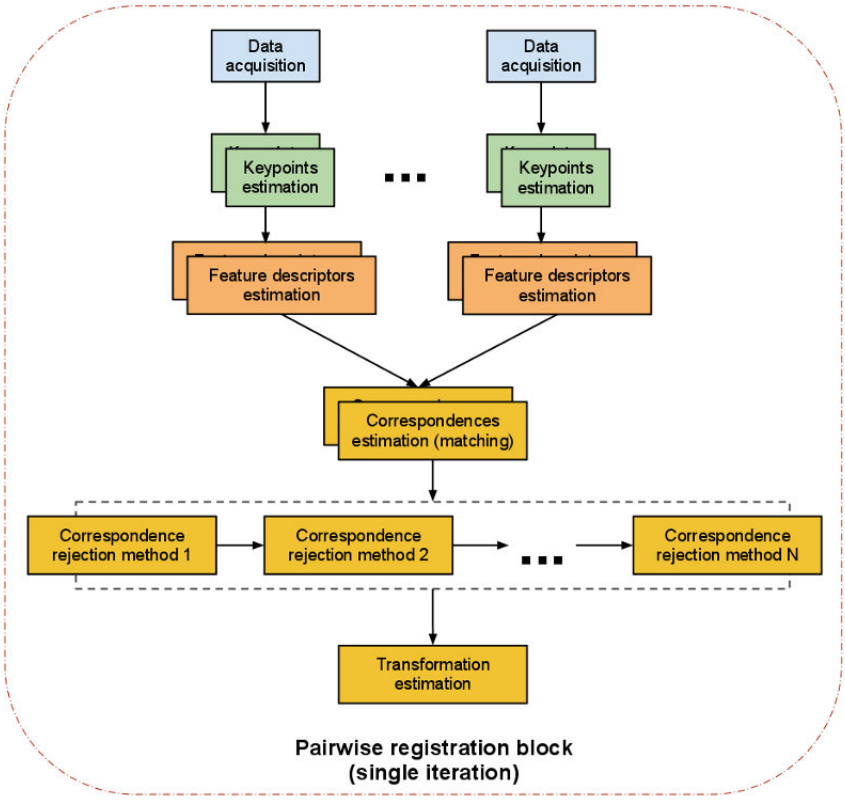


FIGURE 3.1: 3D object recognition pipeline[1]. For each input image, keypoints are extracted. The next step consists in the computation on features to describe the local region around keypoint. Matches between descriptors are computed based on a distance metric and positive ones are clustered. Finally a transformation from the input object to the clustered points is estimated.

[1] http://pointclouds.org/documentation

3.1.1 Models

Creating 3D models remains a quite difficult task, expecially when creating models of small objects like a cup or a fork. It is possible to create model using low cost depth cameras, but there are mainly two problems:

- Homogeneous objects have few geometric features making it hard to understand the current object position while moving around the scanner. To avoid this problem it is often enough to place some other objects around the model or to use markers. The drawback of this solution is that there is the need of some post processing to remove the additional objects.
- Most scene reconstruction algorithms are made for rooms or big scenes and tend to approximate small objects; this leads to the problem that a lot of key-geometrical structures get lost. As an example if you try to scan a rubick cube you will probably just get a uniform cube.

There are some specialized libraries for scene reconstruction like kinectfusion and the opensource version kinfu, but both suffer of the described problems. There are also specialized 3D scanners of different dimensions which give professional results, but they come at a higher price.

3.1.2 Keypoint Extraction

Keypoint extraction is done using either RGB or depth information. There are three main categories of keypoints that can be subdivided based on the cloud information which are used. A genera survey can be found in [154]. Here we will present just a few of the most commonly used keypoint detectors.

1. RGB:
    - SIFT [97] (Scale-Invariant Feature Transform) is one of the most used algorithms , which works analysing the RGB values of the point cloud. To find points of interest SIFT tries to locate points which have a high color gradient with the sorrounding. This points usually belong to figure edges or relevant images on uniform surfaces.
    - SURF [10] (Speeded Up Robust Feature) is several times faster and more robust against different image transformations than SIFT. It uses an integer approximation of the determinant of Hessian blob detector. The determinant of the Hessian matrix is used as

a measure of local change around the point and keypoints are chosen where this determinant is maximal.

- FAST [162] (Features from Accelerated Segment Test) is a corner detection method. The algorithm is very efficient computationally and in general is one of the fastest keypoint detectors available. To detect keypoinys, FAST evaluates a circle of 16 pixels to classify whether a candidate point is a corner. If a set of contiguous pixels in the circle are all brighter than the intensity of the candidate pixel plus a threshold value or they are all darker than the intensity of candidate pixel minus a threshold, then the candidate point is classified as corner.
- ORB [131] (Oriented FAST and Rotated BRIEF) aims to provide a fast and efficient alternative to SIFT. ORB consists in a fusion of the FAST and BRIEF descriptor with several modifications to enhance performance. It uses FAST to detect keypoints and Harris to detect the best ones among them. An additional enhancement has been added to make the detector rotation invariant.
- SUSAN [143] (Smallest Univalue Segment Assimilating Nucleus) uses a circular region to detect if a candidate pixel is a keypoint.; the candidate pixel is called nucleus. Similarly to FAST is uses a comparison function to evaluate the pixels in the area and determine if the nucleus is a keypoint or not. The function is based mainly on the brightness difference threshold of the pixels in the area.
- Harris [39] is capable of identifying similar regions in images taken from different viewpoints that are related by a simple geometric transformation: scaling, rotation and shearing. To do this, the algorithm follows a sequence of steps which consists in: Identify initial region points using scale-invariant Harris-Laplace Detector, normalize each region to be affine invariant using affine shape adaptation, select proper integration scale, differentiation scale and spatially localize interest points, update the affine region using these scales and spatial localisations and finally iterate the previous steps if the stopping criterion is not met.

2. DEPTH:

- *NARF* [147] (Normal Aligned Radial Feature) works by analysing the depth values of the point cloud. To find points of interest

NARF tries to find points for which the depth value changes rapidly in their sorrounding. This technique allows to find edge points which are of particular interest in 3D images.

- 3D SURF [83] is an extension of SURF which is based on the voxelization of an input cloud. Each produced bin is then described using a modified version of SURF.

3. RGB and/or Depth: *Sampling* has two working modes: Uniform and Random. Uniform sampling is very fast and scans the whole scene, but in principle acquires more points than necessary if the model in the scene occupies only a small fraction of it. Random sampling in contrast can be tuned on the number of desired points but could in principle get keypoints of undesired areas.

3.1.3 Keypoint Descriptors

Descriptors are needed since matching single keypoints is unfeasible given scale, light and rotation changes from the model to the scene. To avoid all these problems, not single keypoints are matched, but regions surrounding the keypoints. This includes cubic regions, spheres (Figure 3.2), couples of points with their direction relatively to the keypoints and so on. In general a descriptor defines a keypoint and its surrounding.

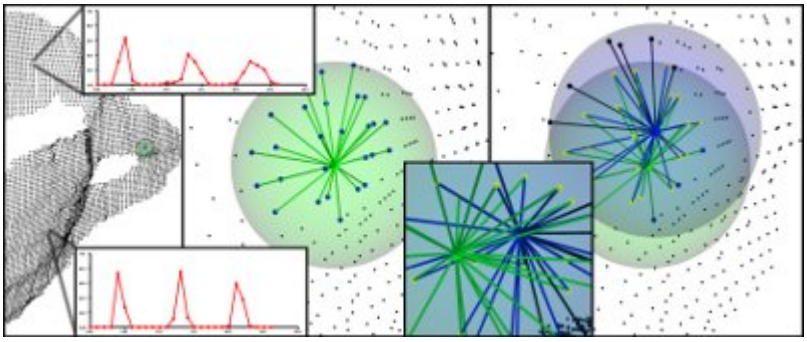


FIGURE 3.2: Spherical reference frame for a descriptor[1].
[1] http://pointclouds.org/documentation/tutorials/how_features_work.php

Descriptors can be *local*, Figure 3.4, or *global*, Figure 3.3; the first ones consider the keypoints relatively to a small surrounding region, while the second ones consider the entire scene . There are three main descriptors which represent the main descriptor categories: *FPFH* [134] , *SIFT* [97] and *Shot* [135]; all of them can be found in different flavors with small variations. In general we have *signature based methods* and *histogram based methods*. The first ones describe the 3D surface neighborhood of a given point by defining

an invariant local Reference Frame (RF). This frame encodes, according to the local coordinates, one or more geometric measurements computed individually on each point of a subset of the support. Histogram-based methods describe the support by accumulating local geometrical or topological measurements (e.g. point counts, mesh triangle areas) into histograms according to a specific quantized domain (e.g. point coordinates, curvatures ...). This domain requires the definition of either a Reference Axis (RA) or a local RF. In broad terms, signatures are potentially highly descriptive thanks to the use of spatially well localized information, whereas histograms trade-off descriptive power for robustness by compressing geometric structure into bins. FPFH is a histogram based descriptor, SIFT is a signature based descriptor and Shot is both a signature and histogram based descriptor.

1. *FPFH* (Fast Point Feature Histogram) is a faster to calculate version of *PFH* (Point Feature Histogram). The goal of the *PFH* formulation is to encode a point's k-neighborhood geometrical properties by generalizing the mean curvature around the point using a multi-dimensional histogram of values. The figure below presents an influence region diagram of the PFH computation for a query point, marked with red and placed in the middle of a circle (sphere in 3D) with radius r, and all its k neighbors (points with distances smaller than the radius r) are fully interconnected in a mesh. The final *PFH* descriptor is computed as a histogram of relationships between all pairs of points in the neighborhood, and thus has a computational complexity of $O(k^2)$.

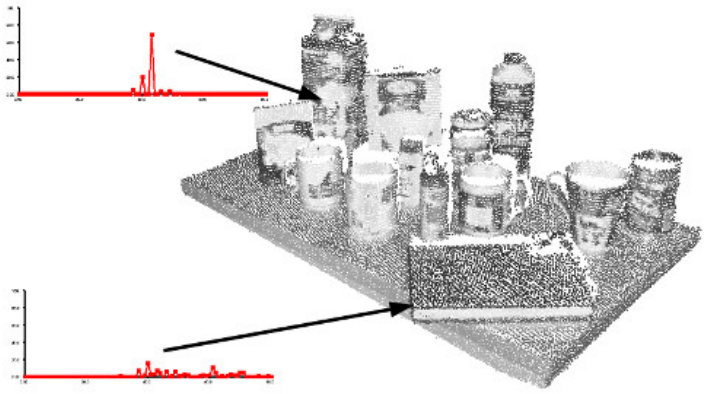


FIGURE 3.3: Histogram based descriptor[1].

[1] https://computervisionblog.wordpress.com/tag/point-cloud-library-2

2. *SIFT* (Scale Invariant Feature Transform) descriptor, a signature based descriptor, uses the gradient information of a region around the keypoint to create a descriptor. Color gradient is used to encode the changes around the keypoint. To be more precise SIFT relies on a set of local histograms, that are computed on specific subsets of pixels defined by a regular grid superimposed on the patch. Later on also RGB has been introduced as an additional feature in some versions of the SIFT descriptor

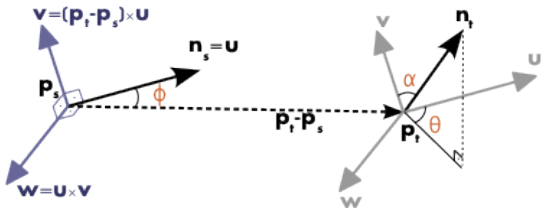


FIGURE 3.4: Signature based descriptor[1].

[1] http://pointclouds.org/documentation/tutorials/pfh_estimation.php

3. *SHOT* (Signature of Histograms of OrienTations) descriptor combines the advantages from both signature based methods and histogram based methods. It encodes histograms of basic first-order differential entities (i.e. the normals of the points within the support), which are more representative of the local structure of the surface compared to plain 3D coordinates. The use of histograms brings in the filtering effect required to achieve robustness to noise. Having defined an unique and robust 3D local RF, it is possible to enhance the discriminative power of the descriptor by introducing geometric information concerning the location of the points within the support, thereby mimicking a signature. This is done by first computing a set of local histograms over the 3D volumes defined by a 3D grid superimposed on the support and then grouping together all local histograms to form the actual descriptor.

4. BRIEF [22] (Binary Robust Independent Elementary Features) was developed to lower the memory usage of keypoint detectors and descriptors, in order to be used in memory constraint systems. BRIEF uses smoothened image patches and selects a set of location pairs in an unique way. Then some pixel intensity comparisons are done on these location pairs. For each location a quick comparison function is evaluated to produce a binary string which can be used as a descriptor.

5. HOG [38] (Histogram of oriented Gradients) will be described more in detail in connection with its use for face detection in 3.2

3.1.4 Matching Descriptors

Given two sets of descriptor vectors coming from two acquired scans, there are two methods to find corresponding descriptors. It is possible to match descriptors or keypoints, depending if the point clouds are organized; a point cloud is "organized" if there is a viewpoint in space where the point cloud can be described like an image (X, Y coordinates) and a last parameter: depth. It is possible to match points or descriptors, even if, as stated previously, point matching is not a good solution. The following three methods are available:

1. Brute force.
2. Indexing using KD-trees, K-Means trees etc.
3. Using organized data to search for a correspondence in a small region.

To find the best matches between model and scene descriptors, FLANN (Fast Library for Approximate Nearest Neighbors) [103] is one the most popular choices. FLANN is a library for fast approximate nearest neighbor searches in high dimensional spaces. It is a specific implementation of a KD-tree, see Figure 3.5, or k-dimensional tree, which is a data structure used for organizing points in a space with k dimensions. Model descriptors or scene descriptors are inserted into a KD-tree; for each descriptor of the other set, the nearest neighbor in the KD-tree is found. Once two desriptors are matched, a correspondence is created and stored. In general all algorithms used for minimum distance search can be used to match descriptors, depending on the search space dimensions.

Naturally, not all estimated correspondences are correct. Since wrong correspondences can negatively affect the estimation of the final transformation, they need to be rejected. This could be done using RANSAC (RANdom SAmple Consensus) or by trimming down the amount of correspondences using only a certain percent. RANSAC is an iterative method to estimate parameters of a mathematical model from a set of observed data that contains outliers. It is a non-deterministic algorithm in the sense that it produces a reasonable result only with a certain probability, with this probability increasing as more iterations are allowed. RANSAC works very well with well structured objects which have a "geometrical" shape. This method tries to extract keypoints that lie on the geometrical structure specified; for example

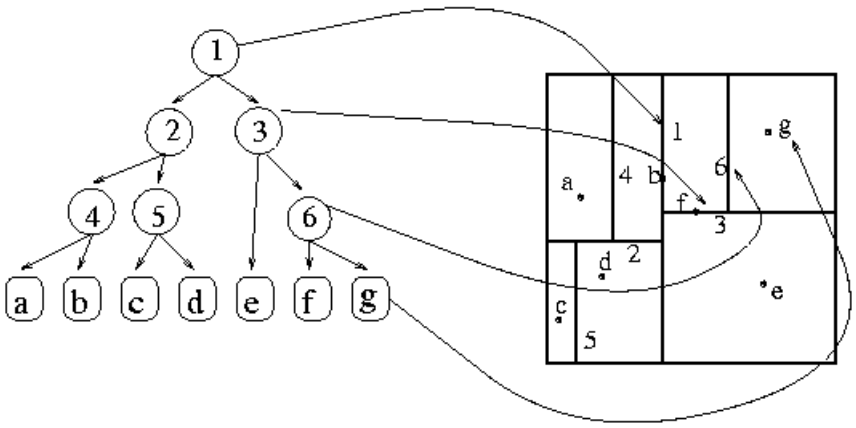


FIGURE 3.5: Example of a KD-tree[1].
[1]http://groups.csail.mit.edu/graphics/classes/6.838/S98/meetings/m13/kd.html

if a plane is specified then RANSAC will try to find keypoints which lie on a plane.

There is also the possibility of using PPF (Point Pair Features)[43], four-dimensional descriptors of the relative position and normals of pairs of oriented points on the surface of an object ; this algorithm covers the whole recognition pipeline, implementing a custom version of each step. Compared to traditional approaches based on point descriptors, which depend on local information around points, this algorithm creates a global model description based on oriented point pair features and matches the model locally using a fast voting scheme. The global model description consists of all model point pair features and represents a mapping from the point pair feature space to the model, where similar features on the model are grouped together. Such a representation allows to use much sparser object and scene point clouds, resulting in very fast performance. Recognition is done locally using an efficient voting scheme, similar to the Generalized Hough Transform, to optimize the model pose, which is parametric in terms of points on the model and rotation around the surface normal on a reduced two-dimensional search space. Descriptors are matched using a hash table that allows an efficient lookup during the matching phase. This method could be used in principle to match any kind of descriptor.

3.1.5 Clustering Correspondences

This step is used to avoid sparse correspondences which are of no interest. The found correspondences are clustered since a good match between the model and the scene is found, see Figure 3.6, only when a big number of correspondences is found in a relative compact space. A possible method

is called General Hough Trasform (GHT) [6]; this method is based on the Hough Transform (HT), which is a popular computer vision technique originally introduced to detect lines in 2D images. Successive modifications allowed the HT to detect analytic shapes such as circles and ellipses. Overall, the key idea is to perform a voting of the image features (such as edges and corners) in the parameter space of the shape to be detected. Votes are accumulated in an accumulator whose dimensionality equals the number of unknown parameters of the considered shape class. For this reason, although general in theory, this technique can not be applied in practice to shapes characterized by too many parameters, since this would cause a sparse, high-dimensional accumulator leading to poor performance and high memory requirements. By means of a matching threshold, peaks in the accumulator highlight the presence of a particular shape in the image. The GHT extends the HT to detect objects with arbitrary shapes, with each feature voting for a specific position, orientation and scale factor of the shape being sought. To reduce the complexity, the gradient direction is usually computed at each feature position to quickly index the accumulator.

GHT has though well known limitations to deal with 3D shapes and 6-degree-of-freedom poses (in particular, curse of dimensionality and sparseness of the voting space). To avoid this problem a more general RANSAC approach is often used.

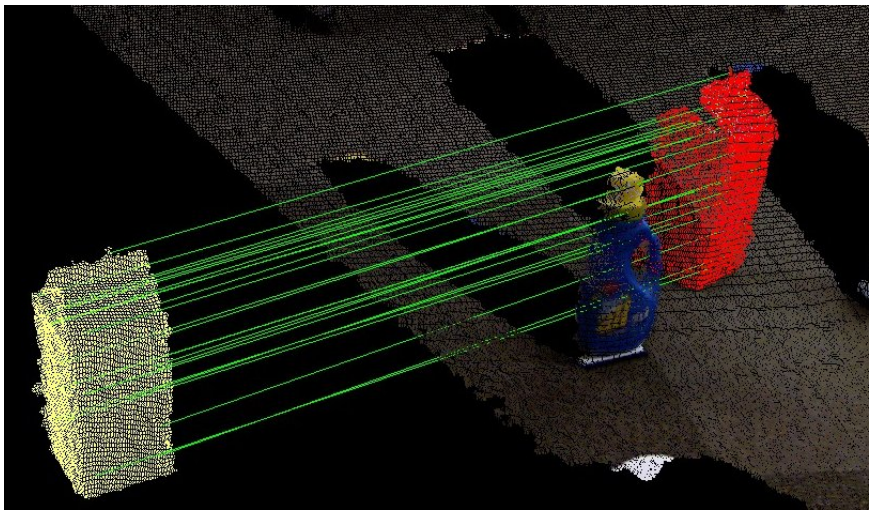


FIGURE 3.6: An example of "Good" correspondences[1].
[1] http://pointclouds.org/documentation/tutorials/correspondence_grouping.php

Another possible algorithm is Geometric Consistency (GC) [24]. This algorithm is based on a mathematical condition that checks the consistency of pairs of correspondences in the 3D space. It can be used in combination

with a local surface descriptor for surface representation. A local surface descriptor is characterized by its centroid, its local surface type and a 2D histogram. The 2D histogram shows the frequency of occurrence of shape index values vs. the angles between the normal of reference feature point and that of its neighbors. Instead of calculating local surface descriptors for all the 3D surface points, they are calculated only for feature points that are in areas with large shape variation. In order to speed up the retrieval of surface descriptors and to deal with a large set of objects, the local surface patches of models are indexed into a hash table. Given a set of test local surface patches, votes are cast for models containing similar surface descriptors. Local surface patches candidate models are hypothesized based on potential corresponding.

The first algorithm performs better in most situations, but is slower while the second one is faster, but generates a lot of false positives that have to be filtered. GHT has also the advantage that it can be fine tuned, while GC allows for almost no tuning.

3.1.6 Estimate Object Pose

After clustering the accepted correspondences and voting for each result, a set of possible solutions is extracted along with a possible pose using the GHT. The pose can then be refined using a registration algorithm. The registration procedure consists in the process of aligning two point clouds to obtain the minimum distance between them. A famous iterative example of such an algorithm is Iterative Closest Point (ICP) [173]; in this algorithm, one point cloud, the reference or target, is kept fixed, while the other one is moved to best match the reference. This algorithm iteratively revises the transformation (combination of translation and rotation) needed to minimize the distance from the source to the reference point cloud. The basic steps are the following:

1. For each point in the source cloud find the nearest point in the reference cloud.

2. Estimate the combination of rotation and translation using a mean squared error cost function that will best align each source point to its match found in the previous step.

3. Transform the source points using the obtained transformation.

4. Iterate over the previous steps until some convergence criterion is met, usually a fixed number of iterations or a distance error between source and reference point cloud.

The main disadvantage of such an algorithm relies in its high resource demand due to the ICP complexity; in fact to obtain real time results quality parameters have to be lowered. The main advantage of this algorithm is that it achieves very good alignment results, refining the initially estimated pose. If used to make only small alignments, fast results can be achieved with minimum effort.

A faster version of ICP exists and is called projective-ICP [50], but it needs a good initialization. There exists also a GPU implementation of such algorithm which improves drastically the speed and is used in applications as KinectFusion [109]/

3.1.7 Qualitative Analysis

The following tables 3.1, 3.2, 3.3 show a qualitative analysis of the different proposed methods in terms of the different pipeline steps required in an object recognition pipeline.

	Quantity	Quality	Used info
NARF	Low	High	Depth
SIFT	High	Medium	Color
Sampling	Variable	Randomic	-
PPFE	Low	High	Depth

TABLE 3.1: Qualitative evaluation for the keypoint extraction.

	Time	Quality	Type
SHOT	Fast	High	Hist. & Sig.
FPFH	Slow	Medium	Signature
SIFT	Slow	High	Histogram
PPFE	Fast	High	Signature
PFH	Very Slow	Medium	Signature

TABLE 3.2: Qualitative evaluation for the keypoint descriptors.

	Time	Quality	Flexibility
Hough	Fast	High	High
Geometric	Fast	Low	Low

TABLE 3.3: Qualitative evaluation for the clustering algorithms.

3.1.8 Measurements

It is important to measure the quality of the recognized object in the scene; to do this three main quantitative measures have to be considered, which have to be added to the qualitative measures displayed in Figure 3.1.7:

- If the object is recognized.
- What is the error in the pose estimation.
- Robustness against cluttering and noise.

The first parameter can be evaluated on standard datasets, calculating false positives and false negatives. It is quite complex to find false positives and false negatives without having a human analyzing the scene and comparing it to the detected object. One possible solution is to align the recognized object with the scene using a registration algorithm as ICP and then calculate the fitness of the alignment. The main problem in this case is to find a good threshold for the fitness value, which can discriminate false positives and false negatives.

The second parameter is evaluated calculating the distance and rotation error between the estimated object and the real object [75]. A lot of different measures (Φ) have been implemented to compute the difference of two rotations expressed as quaternions (q_1, q_2) or rotation matrices (R_1, R_2), like:

1. Inner Product of Unit Quaternions:
   $\Phi_3 : S^3 \times S^3 \rightarrow R^+$,
   $\Phi_3(q_1, q_2) = \arccos(|q_1 \cdot q_2|)$
2. Geodesic on the Unit Sphere:
   $\Phi_6 : SO(3) \times SO(3) \rightarrow R^+$,
   $\Phi_6(R_1, R_2) = \| \log(R_1 {R_2}^T) \|_F$
   where $\| \dots \|_F$ denotes the frobenious norm, $\log(R) = \frac{\theta}{2\sin(\theta)}(R - R^T)$ and $1 + 2\cos(\theta) = Tr(R)$.

The third error can be estimated adding a constant uniform gaussian noise, occluding a fixed percentage of the object and then calculating the previously described errors. It is important to notice also that all this measures make sense assuming that the algorithm remains real-time or almost real-time since we are evaluating ASs that interact with humans. This gives a strong constraint when trying to optimize the quality of the detector. By this we mean that relaxing this condition allow to get almost arbitrary good results.

3.1.9 Multi Object Detection

Detecting N objects in a scene increases a lot the computational requirements of the algorithm. Some stages of the pipeline remain almost the same and do not add any overhead, like the calculation of keypoints and descriptors for each model which can be done offline in a preliminary phase. The main performance issue is given in the comparison of the descriptors from the models and the scene; in the worst case we have $O(k^2)$ comparisons, which slows down a lot the detection process. A first solution could be to create in parallel N different detectors, but this depends on the number of available processing cores. Acceptable results can still be achieved having small point clouds and a good pre-filtering stage that removes all the non interesting regions [29].

3.1.10 Final Remarks

As described before there is no perfect solution to the problem of detecting an object in a 3D scene in real time. The pipeline remains very complex and needs a lot of parameter tuning to obtain good results, depending too much on the input, which has to be recognized. The algorithms need to be implemented efficiently on GPU to obtain good performance results, since now the computational complexity is too demanding. ML approaches simplify drastically the problem given that the parameters are learned automatically by the system and computation can be easily done on GPUs by default.

3.2 Face Detection

The most used classical face detection algorithm remains the Viola and Jones [161] one that is based on Haar features. Another well known algorithm is based on HoG. Both approaches are based on Support Vector Machines (SVM), which was one to the most used ML techniques before the rediscovery of NNs, as described previously. We will describe in the following both approaches specifying the critical aspects that made such systems perform worst than newer systems based on CNNs.

3.2.1 Haar Detector

A Haar detector is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images. This kind of detector is based on Haar features that are usually computed over rectangular pixel areas.

A Haar-like feature considers adjacent rectangular regions at a specific location in a detection window. The value of a feature corresponds to the difference between sums of pixel intensities in different regions. A 2-rectangle feature, Figure 3.7 (a), consists in the sum of the pixels in the white rectangle area minus the sum of the pixels in the black rectangle area. A three-rectangle feature computes the sum within two outside rectangles subtracted from the sum in a center rectangle, Figure 3.7 (b). Finally a four-rectangle feature computes the difference between diagonal pairs of rectangles, Figure 3.7 (c).

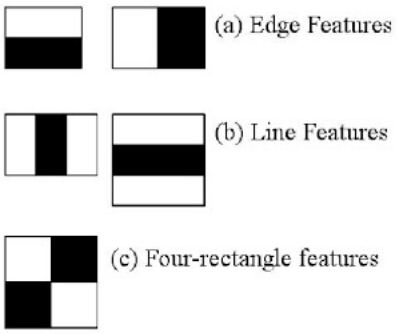


FIGURE 3.7: Example of Haar features. (a) represents a 2-rectangle feature, (b) represents a 3-rectangle feature and (c) represents a 4-rectangle feature[1].
[1]http://docs.opencv.org/trunk/d7/d8b/tutorial_py_face_detection.html

This kind of features can be used, for example, to identify faces noticing that the region around the eyes is darker than the region of the cheeks. Therefore a common Haar feature for face detection is a set of two adjacent rectangles that lie above the eye and the cheek region, Figure 3.8. The position of these rectangles is defined relative to a detection window that acts like a bounding box to the target object .

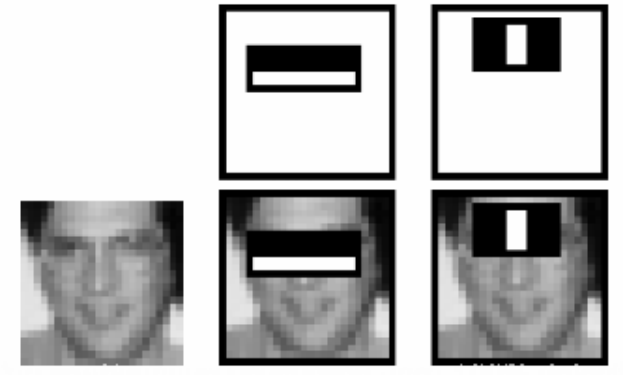


FIGURE 3.8: Example of Haar features in face recognition[1].
[1]http://docs.opencv.org/trunk/d7/d8b/tutorial_py_face_detection.html

Initially, the algorithm needs a lot of positive and negative examples to train the classifier. The next step consists in computing descriptors for the given examples. For this, Haar features shown in Figure 3.7 are used. Each feature is a single value obtained by subtracting the sum of the pixels under white rectangle from the sum of the pixels under the black rectangle.

The speed with which features may be evaluated does not adequately compensate for their number. For example, in a 24x24 pixel sub-window there are a total of M = 162,336 possible features, and it would be prohibitively expensive to evaluate them all when testing an image. Thus, the object detection framework employs a variant of the learning algorithm AdaBoost to both select the best features and to train classifiers that use them. This algorithm constructs a "strong" classifier as a linear combination of weighted simple "weak" classifiers.

The algorithm can be improved even more given that most areas of an image don't contain faces, so it is possible to use Cascading. in cascading, each stage consists of a strong classifier. All the features are grouped into several stages where each stage has certain number of features.

The job of each stage is to determine whether a given sub-window is definitely not a face or may be a face. A given sub-window is immediately discarded as not a face if it fails in any of the stages.

The Viola and Jones approach offers real-time performance and scale-location invariance, but it still has a few disadvantages as intolerance to object rotations, sensitivity to illumination variations, occlusion etc.

3.2.2 HoG Detector

The essential thought behind the HoG descriptors is that objects and shapes in an image can be described by the color intensity gradients or edge directions. An image is divided into small connected regions called cells, and for the pixels within each cell, a histogram of gradient directions is computed. The image descriptor is the concatenation of the computed histograms.

The HoG descriptor has a few key advantages over other descriptors. Since it operates on local cells, it is invariant to geometric and photometric transformations, except for object orientation. Such changes would only appear in larger spatial regions. The HoG descriptor is thus particularly suited for human detection in images.

In the example of face recognition, an SVM is trained with examples of HoG features computed for faces (positive and negative examples), Figure

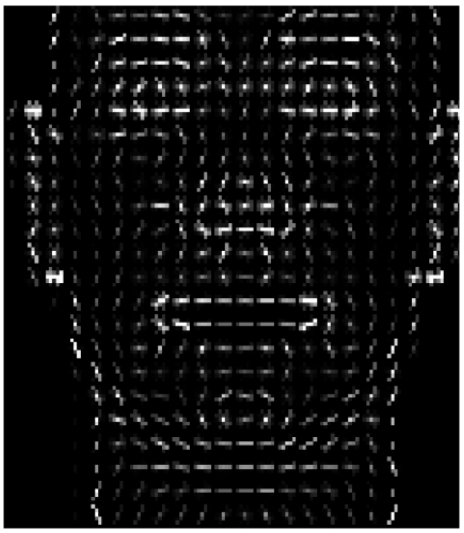


FIGURE 3.9: Example of gradients computed on a face [130].

3.9. HoG suffers mostly of the same disadvantages of Haar except that it is robust to illumination changes.

3.2.3 Haar Vs HoG

HoG features are capable of capturing object outline/shape better than Haar features. On the other hand, simple Haar features can detect regions brighter or darker than their immediate surrounding region better than HoG features. In general HoG features can describe shape better than Haar features and Haar features can describe shading better than HoG features. That is also why Haar features are good at detecting frontal faces and not so good for detecting profile faces. This is because the frontal face has features such as the nose bridge that is brighter than the surrounding face region. But the profile face most prominent feature is it's outline or shape, hence HoG would perform better for profile faces. HoG and Haar features are complementary features, hence combining them might even result in better performance.

Given the work of Abrah et al. [2] and Negri et al.[108] it is possible to see that Haar performs generally better than HoG. It is important to notice that both detectors have to be executed on windows of different scales, which have to be slid over the figure resulting in high computational cost. Also both approaches suffer of several problems like side views, illumination, occlusion etc. Newer ML methods as CNNs overcome this problems as described in Chapter 4.

3.3 Body Pose Estimation

A body pose consists in general of rigid parts and joints. The human body has 244 degrees of freedom with 230 joints, making pose estimation a very hard problem. Algorithms must account for large variability introduced by differences in appearance due to clothing, body shape, size, and hairstyles. Additionally self occlusion is something very common, given that often parts of a body are covered by other parts during movements. Typically body pose estimation systems are based on a template matching approach, in which the pose estimation is achieved by maximizing the similarity between an observation and a template model.

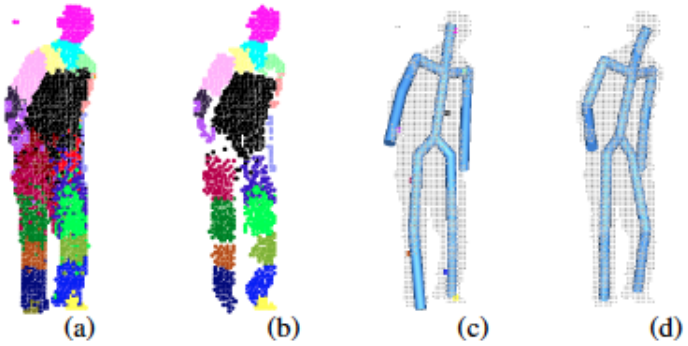


FIGURE 3.10: Example of estimated body poses[1].
[1] https://www.slideshare.net/cameraculture/kinect-tutorial

The most commonly used model to reconstruct a human pose is based on part-models, see Figure 3.10. To represent mathematically a human body, parts are connected to each other using springs. Hence this kind of model is also known as Spring model [15]. The position of each part is expressed by the compression and expansion level of the springs. It is important to notice that there are geometric constrain on the orientation of the springs. For example, limbs of legs cannot move 360 degrees. Hence parts cannot have that extreme orientation. This reduces the possible poses space making it more tractable for algorithms.

The spring model forms a graph G(V,E) where nodes (V) corresponds to the parts and edges (E) represent springs connecting two neighboring parts. Each location in image can be expressed by the x and y coordinates of the pixel location. Let $\mathbf{p}_i(x, y)$ be the point at $\mathbf{i}^{th}$ location. Then, the cost associated in joining the spring between $\mathbf{i}^{th}$ and the $\mathbf{j}^{th}$ point can be

given by $S(\mathbf{p}_i,\ \mathbf{p}_j) = S(\mathbf{p}_i - \mathbf{p}_j)$. Hence, the total cost associated in placing l components at locations $\mathbf{P}_l$ is given by

$$S(\mathbf{P}_l) = \sum_{i=1}^{l} \sum_{j=1}^{i} \mathbf{s}_{ij}(\mathbf{p}_i,\ \mathbf{p}_j) \tag{3.1}$$

The above equation simply represents the spring model used to describe body pose.

Several approaches have been used successfully to estimate and track body poses [102]. Most of them are based on template matching, achieving good results, but suffering often from the occlusion problems [61].

One of the most successful approaches, which was used in conjunction with Kinect v1 camera, is the work of Shotton et al. [139] that uses a single depth image to estimate the body pose. It is based on Random Decision Forests which are trained on a dataset of labeled body poses. In addition, several synthetic images have been created to enrich the training dataset given the high variability of the possible input poses. Final results show over 90% confidence in the detection of almost all body parts at $\sim 200Hz$ on an Xbox GPU.

Chapter 4

Deep Learning

> "We need to go deeper."
>
> Inception, [76]

Here we will briefly introduce DL to cover the state of the art and the topics of the following chapters. DL is based on DNN which are larger NNs with a higher number of layers, bigger training sets and specialized layers as convolutional, recurrent, LSTM ...

4.1 Deep Neural Network

DNNs are computing systems inspired by the biological neural networks that constitute animal brains. Such systems learn tasks by examples, generally without task-specific programming. They have found most use in applications where it is difficult to find features, where input data is noisy, or where the input space is very large given the ability of DNNs to extract meaningful information discarding the rest.

A DNN is based on a collection of connected units called artificial neurons, analogous to axons in a biological brain. Each connection (synapse) between neurons can transmit a signal to another neuron, Figure 4.1. The receiving (postsynaptic) neuron can process the signal(s) and then signal downstream neurons connected to it. Neurons may have a state, generally represented by real numbers, typically between 0 and 1. Neurons and synapses may also have a weight that varies as learning proceeds, which can increase or decrease the strength of the signal that it sends downstream. Further, they may have a threshold such that only if the aggregate signal is below (or above) that level, the downstream signal is sent. These thresholds are called activation functions and several of them are available based on the learning problem, Figure 4.2. In general they are necessary since a sequence of linear operations can be aggregated as a single linear operation and so having multiple layers would be useless. This way the neural network is not a sequence of linear operations, but a sequence of linear operations

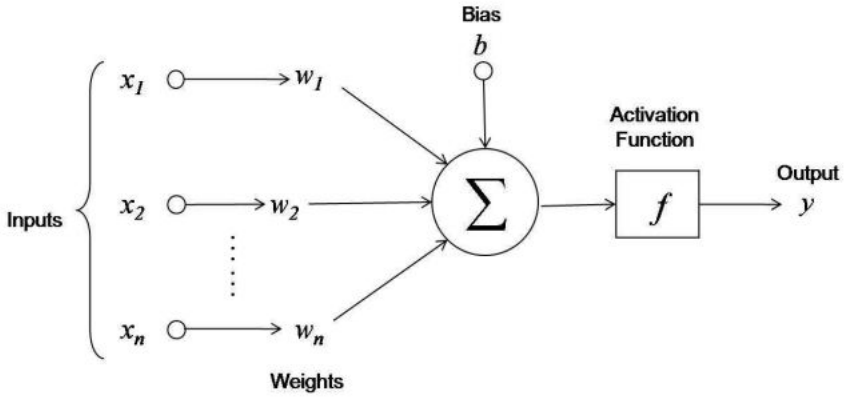


FIGURE 4.1: Example of artificial neuron. Input values x_i are multiplies by weights w_i and then summed up along with a bias vector b. The output is passed through an activation function f to produce the final output y. The actual Neuron is only the first part of the image summing pu all values along with the bias.[1]

[1] https://tex.stackexchange.com/questions/132444/diagram-of-an-artificial-neural-network

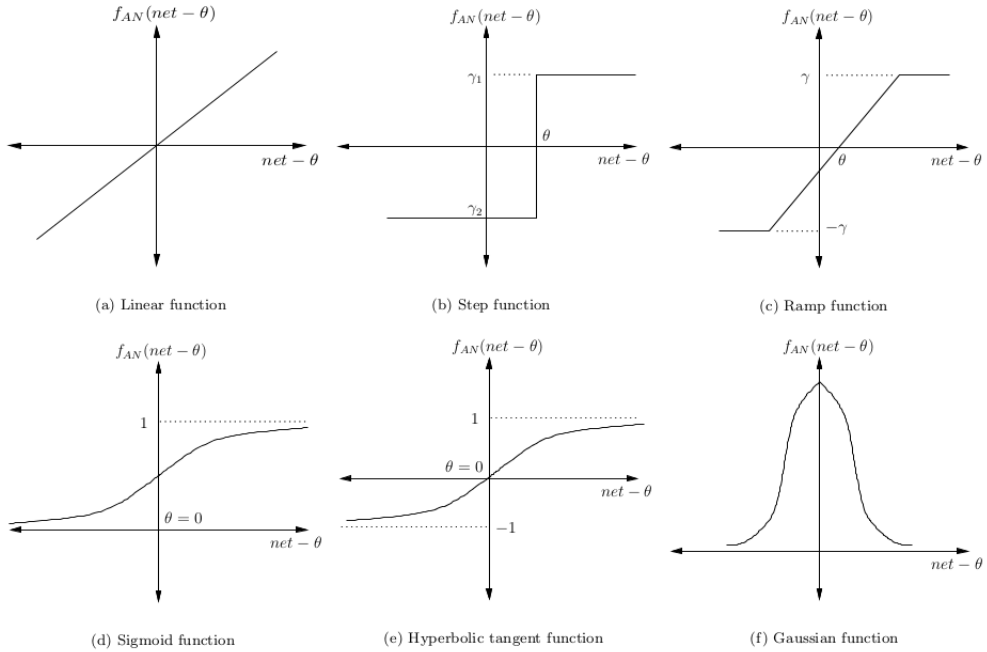
interleaved with non linear ones, which makes it possible for the network to learn more complex functions.

Typically, Artificial Neural Networks are organized in layers. Different layers may perform different kinds of transformations on their inputs. Signals travel from the first (input) to the last (output) layer, possibly after traversing the layers multiple times. When each neuron of one layer is connected to each neuron of the next layer, the system is called a fully connected network, Figure 4.3.

A DNN is composed of a pipeline of linear matrix multiplications, which are followed, after each stage, by a nonlinear function called activation function. These non linear functions are usually a sigmoid function as *TanH* or *Relu*. The general behavior can be synthesized as follows: given an input vector x, a series of matrices A_i composed of weights $w_{(k,j)}$, a bias vector b, an activation function F and an output Y_i, it is possible to write stage i as:

$$Y_i = F(A_i x + b) \tag{4.1}$$

The output Y_i will then be the input of the next stage of the pipeline, until reaching the end where a classifier or regressor computes the final output. DNNs can be used for nonlinear classification or regression. In the first case the network is trained to obtain a label indicating the category to which the input belongs, while in the latter, the network learns to fit an unknown



(a) Linear function (b) Step function (c) Ramp function

(d) Sigmoid function (e) Hyperbolic tangent function (f) Gaussian function

FIGURE 4.2: Examples of possible activation functions.[1]

[1] http://www.turingfinance.com/misconceptions-about-neural-networks

function using the input and output data in order to estimate points which are not present in the input set.

In DNNs, each layer of nodes trains on a distinct set of features based on the previous layer's output. The further you advance into the neural net, the more complex are the features your nodes can recognize, since they aggregate and recombine features from the previous layer, see Figure 4.7. This is known as feature hierarchy, and it is a hierarchy of increasing complexity and abstraction. It makes deep-learning networks capable of handling very large, high-dimensional data sets with billions of parameters that pass through nonlinear functions.

The original goal of the DNN approach was to solve problems in the same way that a human brain would. To train such networks a *backpropagation* algorithm is used, which back propagates gradients through the network to adjust it to the correct output. This kind of networks suffer some times from producing low level features or overfitting the input data, given that each neuron receives inputs from all the previous ones. To overcome this problem, a technique called Dropout has been introduced [145]. Dropout is expressed as a probability value which deactivates a connection between two neurons with a probability D. This encourages the network to find higher level features given that neurons can not rely on having always the

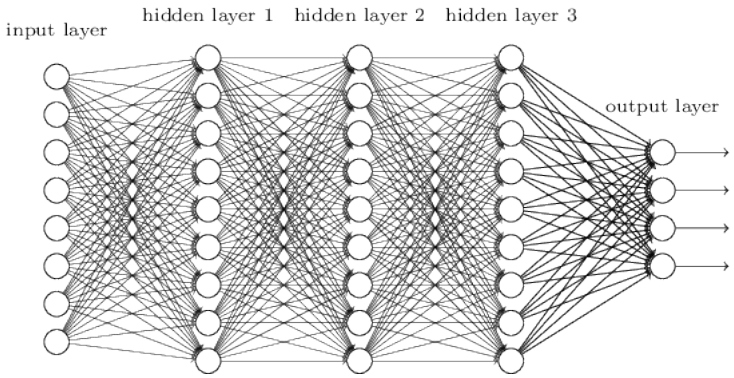


FIGURE 4.3: Example of multilayer NN with fully connected layers.[1]

[1] http://neuralnetworksanddeeplearning.com/chap6.html

same information from the previous layer. In general, techniques used to reduce the problem of overfitting are called regularization.

In recent times, DL has been proved to be an improved solution of NN capable of extracting higher-level complex information through a hierarchical abstraction and learning process [107]. DL demonstrated to be a valuable approach in solving complex problems, such as human action recognition [126, 138], image segmentation [26] and clustering [115].

Several applications have been developed utilizing DL to complement RGB-D sensors in ML and to obtain a higher accuracy despite imperfect sensor data. Examples can be found in several domains. In action recognition [138], where Shahroudy et al. proposed a new deep learning network for hierarchical RGB-D features factorization for action recognition and a ML algorithm to improve action classification. In object recognition [46], Eitel et al. proposed a system based on two separate CNNs, one for the RGB data stream and one for the depth; the two streams are consecutively combined with a late fusion network to improve the robustness of the approach.

4.2 Convolutional Neural Networks

CNNs are NNs where a certain number of layers is composed of Convolutional layers; these kind of networks have been mainly used for image analysis. The use of these networks started after the great results achieved with Alexnet [88], which improved greatly over the state of the art methods in image classification.

Convolutional layers apply a convolution operation to the input, passing the result to the next layer [44]. The idea is to have a sliding window of size K that is convolved over the whole image in parallel. Each convolutional neuron processes data only for its receptive field (window). It performs a

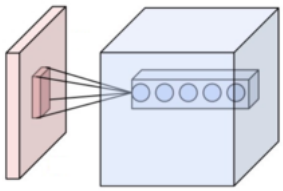


FIGURE 4.4: Examples of a Convolutional layer. The depth of the features increases in the next layer[1].
[1]https://www.slideshare.net/JunhoCho1/convolutional-neural-network-76817816

weighted sum between his receptive field and a weight matrix called Kernel of Filter. Each neuron can apply several filters making the output image "deeper", Figure 4.4. Each produced output image from using one filter is called activation map, Figure 4.5.

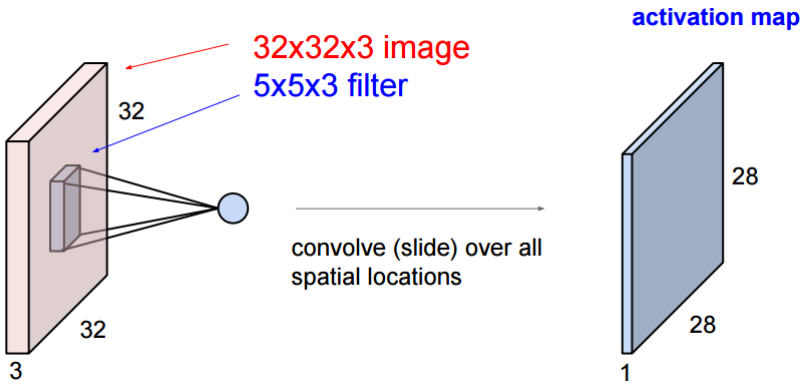


FIGURE 4.5: Examples of an activation map. The size of the image is reduces given that the kernel is convolved only inside the image[1].
[1]https://www.slideshare.net/JunhoCho1/convolutional-neural-network-76817816

Often Convolutiona Layers are followed by a pooling operation that reduces the size of the input image. The pooling layer operates on a window aggregating values into a single value, Figure 4.6. This can be done, for example, using a $max()$ function (Max Pooling) or $avg()$ (Average Pooling).

Given the two operations of Convolution and Pooling, an image is made smaller and deeper while progressing through a CNN. This corresponds to fixing attention to some areas and extracting higher level features at each level of the CNN, Figure 4.7.

It would be possible, in principle, to use Fully Connected NNs to process images, but the number of parameters would be intractable. The convolution operation brings a solution to this problem as it reduces the number of free parameters, allowing the network to be deeper with fewer parameters. In other words, it resolves the vanishing or exploding gradients problem in

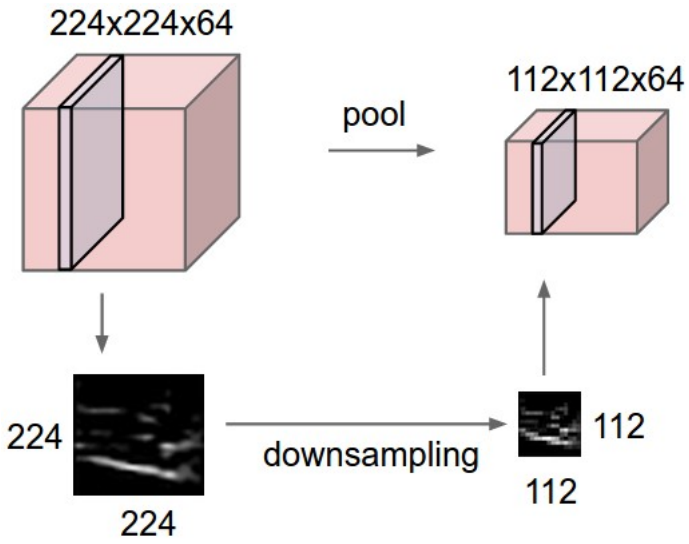


FIGURE 4.6: Examples of a pooling layer. The image is down sampled, making it smaller, but the feature depth is unchanged[1].
[1] http://cs231n.github.io/convolutional-networks/

training traditional multi-layer neural networks with many layers by using backpropagation.

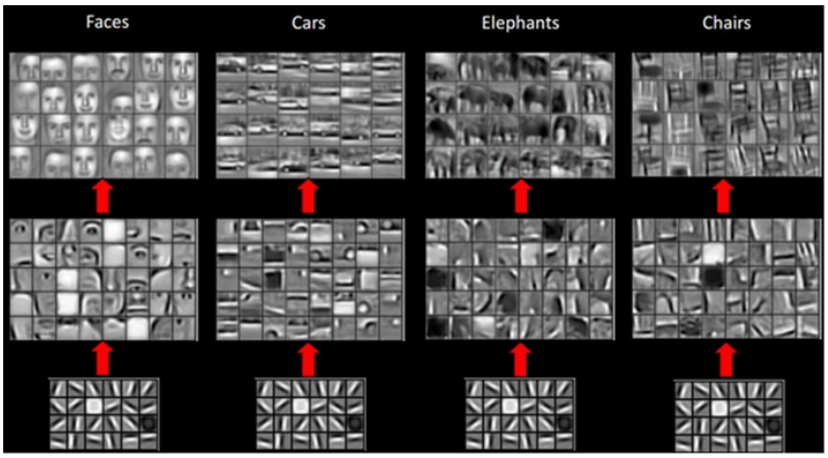
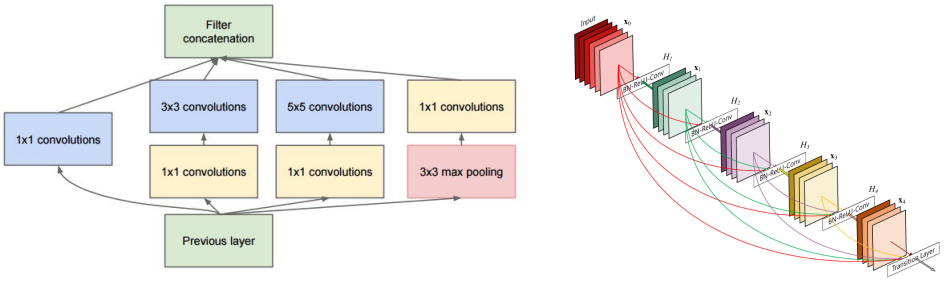


FIGURE 4.7: Examples of CNN features. The initial layers are similar while later layers specialise for the detected content[1].
[1] https://www.slideshare.net/akshaymuroor/deep-learning-24650492

The structure of CNNs also allows fast training and inference on GPUs making CNNs the most commonly used networks type for detection and classification on images.

Latest works have improved CNNs using inception [150] and skip layers [73].

The inception layers consists of a series of variable size convolutional filters that get combined at the end into a deep output, Figure 4.8a. Skip



(A) (B)

FIGURE 4.8: Figure (a) shows the inception layers as a composition of several filters[150]. Figure (b) shows skip layers allow input parameters to flow through the network making it capable of learning the identity function[73].

layers are layers which can be skipped, meaning that there is a connection between each layers and all the successive ones, Figure 4.8b. This makes it possible to learn the identity function, which is usually hard to train in CNNs.

4.3 Recurrent Neural Networks

The idea behind Recurrent Neural Networks (RNN) is to enhance NNs with the ability of tracking time. This means that each neuron need the ability of tracking previous examples to make the current output not only dependent on the current input, but also of the previous one. The decision a recurrent net reached at time step $t-1$ affects the decision it will reach one moment later at time step t. So, recurrent networks have two sources of input, the present and the recent past, which combine to determine how they respond to new data.

That sequential information is preserved in the recurrent network's hidden state, which manages to span many time steps as it cascades forward to affect the processing of each new example.

Recurrent networks rely on an extension of backpropagation called backpropagation through time. Time, in this case, is simply expressed as the combination of a series of steps that have to be computed one after the other. Adding a time element only extends the series of functions for which we calculate derivatives with the chain rule. This can be summarized as unfolding a RNN in time. The unfolded network contains k inputs and outputs, but every copy of the network shares the same parameters. Then the backpropagation algorithm is used to find the gradient of the cost with respect to all the network parameters.

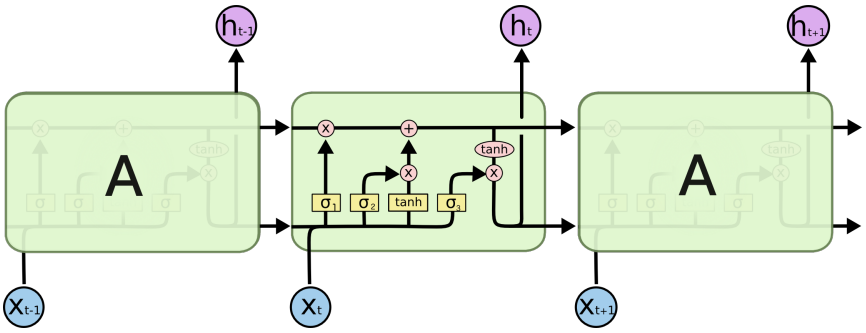


FIGURE 4.9: Schematics representing a sequence of LSTM neurons[1].

[1] https://deeplearning4j.org/lstm.html

It is important to remember that RNNs suffer from the vanishing and exploding gradients problem [117, 70], but there are several solutions present in literature to overcome it [13].

4.3.1 Long Short Term Memory

Long Short Term Memory (LSTM) [71] has been introduced to overcome some problems affecting RNNs and to give networks the ability to decide when to use memory and when to erase it. They have been explicitly designed to avoid the long-term dependency problem. These networks have been widely used for speech recognition [60] and for object tracking [55] where context and previous state are fundamental.

Figure 4.9 shows the schematics of an LSTM neuron. The first decision of an LSTM neuron is to erase or not old information. This decision is made by a sigmoid layer σ_1 called the "forget gate layer". It looks at h_{t-1} and x_t, and outputs a number between 0 and 1 for each number in the cell state C_{t-1}. A 1 represents store this while a 0 represents erase this. It is important to notice that the value is a continuous value between 0 and 1 and so different levels of keep/erase can be obtained.

The next step is to decide what part of the new information has to be kept. First, an input gate layer decides which values to update, σ_2. Next, a *tanh* function creates a vector of new candidate values which is combined with the current input and the previous state to create the new input.

The last step consists in deciding what to output. This output will be based on the current state, but will be filtered. First, we run a sigmoid layer σ_3 that decides what parts of the cell state to output. Then, the output is passed through a tanh function (to push the values to be between -1 and 1) and multiply it by the output of the sigmoid gate, in order to output only the selected components.

Chapter 5

Deep Learning Detection

> "A baby learns to crawl, walk and then run. We are in the crawling stage when it comes to applying Artificial Intelligence."
>
> Dave Waters, [149]

5.1 Object Recognition

Several different techniques have been researched for object recognition [80]. The field can be split between 2D and 3D detection, with possible pose estimation in both cases. Research is still focused mainly on using 2D images for detection given the human ability of solving this problem without depth information. Adding an additional dimension to the problem might bring to performance issues if not addressed correctly. Current DNNs have already hundreds of layers and millions of parameters when working with relatively small images (640x480 for example). Adding a depth value to the input can be tractable if the image is used as a 2D image with 4 channels (rgb-d) using 2D convolutions. The use of 3D convolutions in space instead makes the problem a lot more complicated given that it is necessary to convolve a 3D region over a 3D space (possibly at different scales). Other possible solutions are available, but it is important to notice that the additional data might not bring any benefit; 3D datasets are not widely available and are relatively small compared to existing 2D datasets. The most widely used 2D datasets for benchmarking deep learning algorithms are:

- MNIST: handwritten digits (http://yann.lecun.com/exdb/mnist/)
- NIST: similar to MNIST, but larger (https://www.nist.gov/srd/nist-special-database-19)
- CIFAR10 / CIFAR100: 32×32 natural image dataset with 10/100 categories (http://www.cs.utoronto.ca/~kriz/cifar.html)
- Caltech 101: pictures of objects belonging to 101 categories (http://www.vision.caltech.edu/Image_Datasets/Caltech101/)

- Caltech 256: pictures of objects belonging to 256 categories (http://www.vision.caltech.edu/Image_Datasets/Caltech256/)
- Caltech Silhouettes: 28×28 binary images contains silhouettes of the Caltech 101 dataset (http://people.cs.umass.edu/~marlin/data.shtml)
- STL-10 dataset is an image recognition dataset for developing unsupervised feature learning, deep learning, self-taught learning algorithms. It is inspired by the CIFAR-10 dataset but with some modifications. (http://www.stanford.edu/~acoates//stl10/)
- The Street View House Numbers (SVHN) (http://ufldl.stanford.edu/housenumbers/)
- NORB: binocular images of toy figurines under various illumination and pose (http://www.cs.nyu.edu/~ylclab/data/norb-v1.0/)
- Imagenet: image database organized according to the
- WordNethierarchy (http://www.image-net.org/)
- Pascal VOC: various object recognition challenges (http://pascallin.ecs.soton.ac.uk/challenges/VOC/)
- Labelme: A large dataset of annotated images (http://labelme.csail.mit.edu/Release3.0/browserTools/php/dataset.php)
- COIL 20: different objects imaged at every angle in a 360 rotation(http://www.cs.columbia.edu/CAVE/software/softlib/coil-20.php)
- COIL100: different objects imaged at every angle in a 360 rotation (http://www1.cs.columbia.edu/CAVE/software/softlib/coil-100.php)
- COCO: a large-scale object detection, segmentation, and captioning dataset (http://http://cocodataset.org/)

There are also several 3D image datasets as described in [49], but the main disadvantage of those is that they are created in artificial environments, making it really hard for NNs to generalize features for real world scenarios.

These facts brings up the problem that NNs need always training datasets that are not required in most of the classical approaches to detection, or at least they need smaller ones. Also the size of the datasets which are necessary

for training can be very big given that the objective of NNs is to generalize features, while in classical approaches features are created manually.

5.1.1 2D Detection

2D detection is based mainly on CNNs having as output either bounding boxes or heat maps. The most common networks used in the first approach are YOLO [128], Recurrent CNN (R-CNN) [58], Fast R-CNN [57] and Faster R-CNN [129]. All this networks have a GPU implementation, which makes them very fast, but it is important to notice that for some tasks classical approaches might be faster. This is due to the very high number of layers which are present in many DNNs, which make the computation perform at a speed which is less than real time (assuming 30fps). For example ResNet [67], which was the 2015 ILSVRC winner, has 157 layers and is very deep compared to Alexnet having only 8 layers.

YOLO is composed of single neural network that is applied to the full image. This network divides the image into regions, and predicts bounding boxes and probabilities for each region. These bounding boxes are weighted by the predicted probabilities. Since the whole detection pipeline is a single network, it can be optimized end-to-end directly on detection performance. The networks structure is close to a fully convolutional neural network, outputting bounding boxes in the image. The network is extremely fast and open source, which made it widely used. It has about 9 layers in the original version and 3 times more in YOLO9000; Table 5.1 and 5.2 shows the structure of the classical YOLO and YOLO9000 networks .

R-CNN is based on the creation of region proposals, using a process called Selective Search. Selective Search splits the input image into windows of different sizes, and for each size it groups together adjacent pixels by texture, color or intensity to identify objects. Once the proposals are created, R-CNN creates bounding boxes around the proposed regions and passes them through a modified version of AlexNet (the winning submission to ImageNet 2012).

Fast R-CNN improves over R-CNN using a technique known as RoIPool (Region of Interest Pooling). At its core, RoIPool shares the forward pass of a CNN for an image across its subregions. The CNN features for each region are obtained by selecting a corresponding region from the CNN's feature map. Then, the features in each region are pooled (usually using max

TABLE 5.1: YOLO Layers.

Layer	Kernel	Stride	Output Shape
Convolution	3×3	1	(416, 416, 16)
MaxPooling	2×2	2	(208, 208, 16)
Convolution	3×3	1	(208, 208, 32)
MaxPooling	2×2	2	(104, 104, 32)
Convolution	3×3	1	(104, 104, 64)
MaxPooling	2×2	2	(52, 52, 64)
Convolution	3×3	1	(52, 52, 128)
MaxPooling	2×2	2	(26, 26, 128)
Convolution	3×3	1	(26, 26, 256)
MaxPooling	2×2	2	(13, 13, 256)
Convolution	3×3	1	(13, 13, 512)
MaxPooling	2×2	1	(13, 13, 512)
Convolution	3×3	1	(13, 13, 1024)
Convolution	3×3	1	(13, 13, 1024)
Convolution	1×1	1	(13, 13, 125)

TABLE 5.2: YOLO9000 Layers.

Layer	Kernel	Stride	Output Shape
Convolution	3×3	1	(224, 224, 32)
MaxPooling	2×2	2	(112, 112, 32)
Convolution	3×3	1	(112, 112, 64)
MaxPooling	2×2	2	(56, 56, 64)
Convolution	3×3	1	(56, 56, 128)
Convolution	1x1	1	(56, 56, 64)
Convolution	3×3	1	(56, 56, 128)
MaxPooling	2×2	2	(28, 28, 128)
Convolution	3×3	1	(28, 28, 256)
Convolution	1x1	1	(28, 28, 128)
Convolution	3×3	1	(28, 28, 256)
MaxPooling	2×2	2	(14, 14, 256)
Convolution	3×3	1	(14, 14, 512)
Convolution	1x1	1	(14, 14, 256)
Convolution	3x3	1	(14, 14, 512)
Convolution	1×1	1	(14, 14, 256)
Convolution	3x3	1	(14, 14, 512)
MaxPooling	2×2	2	(7, 7, 512)
Convolution	3×3	1	(7, 7, 1024)
Convolution	1x1	1	(7, 7, 512)
Convolution	3x3	1	(7, 7, 1024)
Convolution	1×1	1	(7, 7, 512)
Convolution	3x3	1	(7, 7, 1024)
Convolution	1×1	1	(7, 7, 1000)

pooling). Faster R-CNN also jointly trains the CNN, classifier and bounding box regressor in a single model.

Faster R-CNN improves the region proposer by reusing the features of the image computed by the forward pass of the CNN instead of using Selective Search as in Fast R-CNN. The Region Proposal Network works by passing a sliding window over the CNN feature map, outputting the potential bounding boxes and probabilities.

Techniques which output heat maps are usually called segmentation networks. In this category we have mainly fully convolutional networks as Dense Convolutional Network (DenseNet) [73] and Mask R-CNN [66]. It is important to notice though that DenseNet is a generic multilayered CNN which can be used for many other purposes.

DenseNet has each convolutional layer connected to every other layer in a feed-forward fashion. Whereas traditional CNNs with L layers have L connections, one between each layer and its subsequent layer, DenseNet has $L * \frac{L+1}{2}$ direct connections. For each layer, the feature-maps of all preceding layers are used as inputs, and its own feature-maps are used as inputs into all subsequent layers. DenseNets has several compelling advantages: they alleviate the vanishing-gradient problem, strengthen feature propagation, encourage feature reuse and substantially reduce the number of parameters.

Mask R-CNN does prediction on single pixel level. It does this by adding a branch to Faster R-CNN that outputs a binary mask that says whether or not a given pixel is part of an object. Some other minor modifications have been implemented in order to make the network produce correctly segmentation and classification. Some example output are shown in Figure 5.1.

5.1.2 3D Detection

3D object recognition using DL is becoming popular only recently. Classical approaches are still the majority and are usually based on template matching and voting schemes as described in Chapter 3. There are though some interesting approaches that have been researched with success in the field.

One possible approach consists in mimicking the work of CNNs, but doing it in 3D space sliding a 3D volume over a point cloud. This is not properly a ML technique, but it resembles it functionality. It has been

FIGURE 5.1: Example bounding boxes and predicted classes using the Mask R-CNN network [66]. Each segmented object is colored with a different color and a label is associated to the surrounding bounding box.

implemented efficiently by Wang et al. in [163] with an additional voting scheme. A proper 3D CNN is also possible and has been implemente by Maurana et al. in [100] and by Zhou et Tuzel in [174]. The idea behind this works is to create an architecture that integrates a volumetric Occupancy Grid representation with a supervised 3D Convolutional Neural Network (3D CNN). VoxNet, for example, achieves accuracy beyond the state of the art on the most used RGB-D datasets.

Another possible approach comes from the use of Deep Belief Networks (DBN). In ML, a DBN is a generative graphical model, or alternatively a class of deep neural network, composed of multiple layers of latent variables, with connections between the layers, but not between units within each layer. A DBN can learn to probabilistically reconstruct its inputs when trained on a set of examples without supervision. After this step, the network can be further trained with supervision to learn classification tasks. Such networks has been trained with success by Nair et al. [106], presenting state of the art results on the NYU Object Recognition Benchmark (NORB) dataset [90], and by Wu et al [172].

Recently Qi et al. [123] have developed a NN which uses directly a 3D pointcloud as input without convolutional operations on it. The network is composed of a first stage which does classification and a second stage

which does segmentation of the point cloud to obtain in the end smaller point clouds of detected objects along with labels. There has been also an improved version of this work again by Qi et al. [124] which improves the previous work by using hierarchical NN recursively on a nested partitioning of the input pointcloud.

5.2 Face Detection

Face detection has reached super human capabilities in the last years. The best algorithms achieve over 99% detection rate using very deep CNNs trained on several hundreds of millions of parameters [136, 151]. This results have bee achieved thanks to the big datasets that have been build in a distribute way using the photo tagging feature. Millions of users have tagged images on social networks creating dataset of hundreds of millions of images.

An opensource implementation of such a system is OpenFace [3], which is a Python and Torch implementation of face recognition with DNNs. It is actually a face recognition system that is also capable of clustering and classifying faces. We used this in Chapter 6 to recognise the faces and the gaze of students performing activities. Figure 5.3 shows the typical workflow of OpenFace for face recognition.

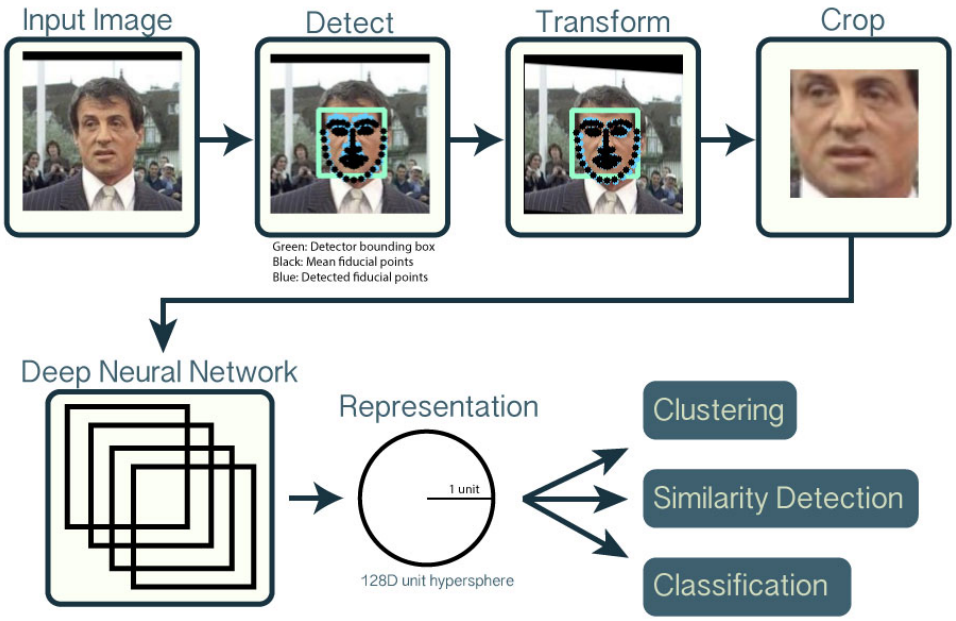


FIGURE 5.2: Example of face recognition pipeline in OpenFace[1].

[1] https://cmusatyalab.github.io/openface/

The following overview shows the workflow for a single input image.

- Detect faces with a pre-trained models using dlib or OpenCV.
- Prepare the image for the NN. OpenPose uses dlib's real-time pose estimation with OpenCV's affine transformation to make the eyes and bottom lip appear in the same location in every image.
- Use a DNN to create a 128-dimensional unit hypersphere. This is a generic representation for anybody's face. Unlike other face representations, this embedding has the nice property that a larger distance between two faces embeddings means that the faces are likely not of the same person. This property makes clustering, similarity detection and classification tasks easier than other face recognition techniques, where the Euclidean distance between features is not meaningful.
- Apply a clustering or classification technique to the features to complete the recognition task.

5.3 Body Pose Estimation

Body pose estimation can be done using RGB or depth images. Most of the work in this field is done using rgb images for the reasons explained previously. The best classical approach is still the Microsoft system explained in Section 3.3 that is based on Random Forests and uses depth information.

A first simple approach is to train CNNs directly from the rgb images to the output pose. Tompson et al. [155] did something similar in the sense that they trained a CNN to detect body parts in RGB images, and then used these as input for a spatial model to remove false positives. Thoshev et al. [157] instead trained directly a 7 layer CNN from the rgb image to the output pose. This gives a rough output but has the advantage of using the whole image and thus capturing its context. To refine the pose, a cascade of classifiers is used, which is trained to predict a displacement of the joint locations from previous stage to the true location. Thus, each subsequent stage can be thought of as a refinement of the currently predicted pose.

An example of work that instead uses single depth images is Huangs et al. [74] work in which they train a CNN from the depth images to the joint positions in 2D camera coordinates. To do this they define a special loss function that takes account of the human body constrains. This type of approach has the disadvantage that it needs a lot of data given the high variability of possible human poses. They overcome the problem with the classical technique of generating mock depth images of body poses using the MakeHuman software [153].

DeepCut [121] is a state-of-the-art approach to multi-person pose estimation based on integer linear programming (ILP) that jointly estimates poses of all people present in an image by minimizing a joint objective. This objective aims to jointly partition and label an initial pool of body part candidates into consistent sets of body-part configurations corresponding to distinct people. In 2016 an improved version of DeepCut was proposed, which is called DeeperCut [77] .

OpenPose [23, 141, 165] is an opensource library for multi-person keypoint detection written in C++ using OpenCV and Caffe. It can be used to detect body poses, faces and hands using 2D rgb images. It improves over DeeperCut achieving a higher accuracy. The used algorithm is based on Part Affinity Fields (PAFs) that are a set of 2D vector fields that encode the location and orientation of limbs over an image domain. The detection system takes, as input, a color image and produces, as output, the 2D locations of anatomical keypoints for each person in the image. First, a NN simultaneously predicts a set of 2D confidence maps S of body part locations and a set of 2D vector fields L of part affinities, which encode the degree of association between parts. Finally, the confidence maps and the affinity fields are parsed by greedy inference to output the 2D keypoints for all people in the image. The general pipeline is depicted in Figure 5.3.

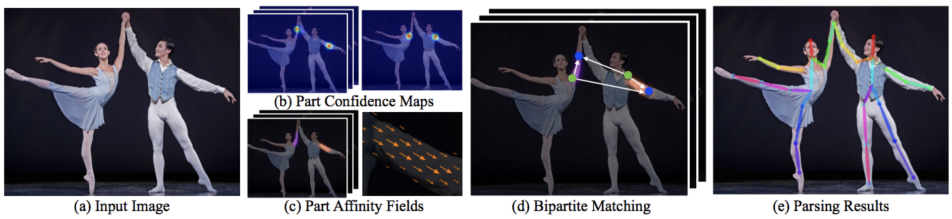


FIGURE 5.3: General detection pipeline of OpenPose[1]
[1] https://arxiv.org/pdf/1611.08050.pdf

The main disadvantage of this library comes from the prediction speed. We evaluated the library using an Nvidia Geforce 1080 GPU, which is currently one of the best performing off the shelf GPUs in the consumer sector. We were not able to achieve real time performances, but got roughly 10fps with 0.5s latency. Shottons algorithm instead is extremely quick with roughly 200fps on lower end consumer hardware without the use of GPUs. This is mainly due to the fact that the evaluation is done on only 3 decision trees of depth 20. The classical approach has also the advantage of being able to run on embedded hardware without GPUs, which are often not available on ASs. It has the disadvantage though of needing depth information, while OpenPose can work with just rgb data.

5.4 Performance

A DNN has an initial computationally intense step that consists in the training of the network. Once the networks has been trained, it is possible to do inference on new data.

5.4.1 Training

To train a NN first an input value is passed through the network computing the output. This output is compared to the expected output and an error vector is computed. Using this, the error is back propagated through he network changing the weights of the network to converge to the expected result. This is an iterative process which should converge to a minimum. Usually it is better to input the whole dataset, or a part of it (batch), and compute an aggregated error vector which is used to update the network weights. This allows also to compute the error gradient in parallel, since different bathes can be passed in parallel through several copies of the same network.

This can be done on one or multiple GPUs; batches are distributed across GPUs memory along with the network weights. It is possible to instantiate multiple networks on the same card in order to exploit parallelism also on a single one. Also matrix multiplication can be parallelised slightly achieving a complexity of $\Theta(n^2)$ instead of $\Theta(n^3)$.

Several problems arise when using distributed machines for parallel training of DNN. The objective is to maintain a linear speedup with the number of used GPUs. IBM Research obtained close to ideal scaling [28] with new distributed deep learning software, which achieved record communication overhead and 95% scaling efficiency on the Caffe deep learning framework over 256 NVIDIA GPUs in 64 IBM Power systems. Previous best scaling was demonstrated by Facebook AI Research of 89% for a training run on Caffe2 [59], at higher communication overhead. IBM Research also beat Facebook's time by training the model in 50 minutes, versus the 1 hour Facebook took. Using this software, IBM Research achieved a new image recognition accuracy of 33.8% for a neural network trained on a very large data set (7.5M images). The previous record published by Microsoft demonstrated 29.8% accuracy [27].

> "Ironically, this problem of orchestrating and optimizing a deep learning problem across many servers is made much more difficult as GPUs get faster. This has created a functional gap in

> deep learning systems that drove us to create a new class of DDL software to make it possible to run popular open source codes like Tensorflow, Caffe, Torch and Chainer over massive scale neural networks and data sets with very high performance and very high accuracy. Here a variant of the "Blind Men and the Elephant" parable is helpful in describing the problem that we are solving and context for the promising early results we have achieved. Per Wikipedia: "...Each blind man feels a different part of the elephant body, but only one part, such as the side or the tusk. They then describe the elephant based on their partial experience and their descriptions are in complete disagreement on what an elephant is." Now, despite initial disagreement, if these people are given enough time, they can share enough information to piece together a pretty accurate collective picture of an elephant. Similarly, if you have a bunch of GPUs slogging through the task of processing elements of a deep learning training problem – in parallel over days or weeks, as is typically the case today – you can synch these learning results fairly easily.
>
> But as GPUs get much faster, they learn much faster, and they have to share their learning with all of the other GPUs at a rate that isn't possible with conventional software. This puts stress on the system network and is a tough technical problem. Basically, smarter and faster learners (the GPUs) need a better means of communicating, or they get out of sync and spend the majority of time waiting for each other's results. So, you get no speedup–and potentially even degraded performance–from using more, faster-learning GPUs."
>
> Hillery Hunter

5.4.2 Inference

Inference consists in a single forward pass of an input through the network. In general this step is very quick, compared to the training phase, but it might be not quick enough for real time results needed in some ASs. There are some techniques to speed up he computation; for example it is possible to write the computed network on an FPGA [89] (Field Programmable Gate Array) board or even an ASIC [116] (Application Specific Integrated Circuit) chip which is then able to pass input values through the network in just a few clock cycles instead of several milliseconds on off the shelf GPUs.

Other possible solutions consist in using specialized off the shelf hardware as for example the Movidius usb stick [78], which is providing a low energy consumption deep learning architecture. Embedded GPUs are also available in products as the Jetson TX1 and Tegra K1 boards, which are already used in many ASs as for example Drones and self driving cars. Figure 5.4 shows a performance comparison of some widely used architectures.

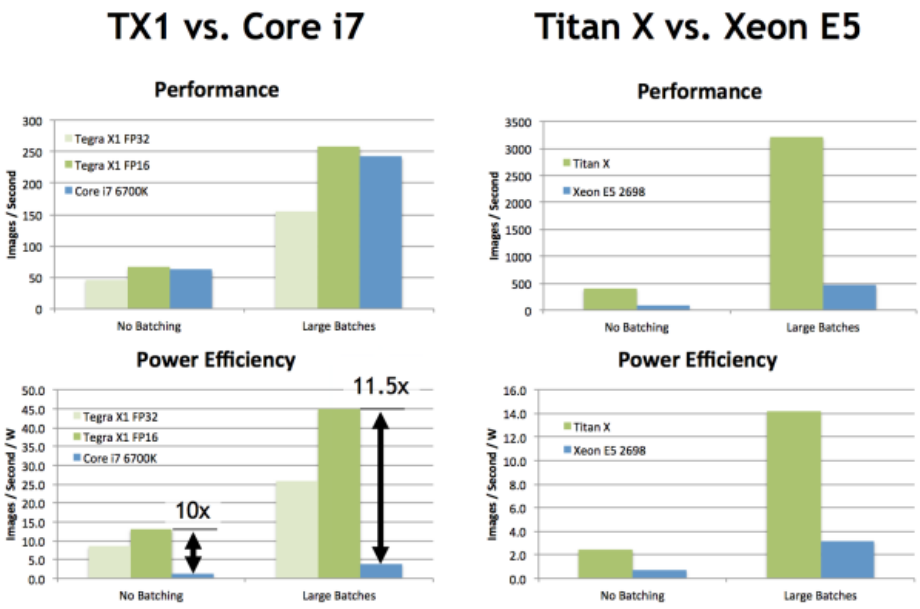


FIGURE 5.4: Performance comparison between an integrated TX1 GPU, an Intel core i7 CPU and between a Desktop Titan X GPU and an intel Xeon processor[1].

[1] https://devblogs.nvidia.com/parallelforall/inference-next-step-gpu-accelerated-deep-learning/

Chapter 6

Activity Recognition

> "Facial recognition software can pick out a person in a crowd, but the vending machine at work can't recognize a dollar with a bent corner."
>
> Jeff Dwoskin, [160]

To understand if it is possible to do successfully activity recognition using ML we developed the PELARS system, see Section 1.3. We used the data collected with this system to test if it is possible to infer if students, working on a project in a small group, are actually developing a successful project or not. This research is focused at answering **RQ1**, **RQ2**, **RQ3**, **RQ4** by showing that it is actually possible for an ML system to take as input an ensemble of noisy data sources and predict with high accuracy the outcome of an observed activity. We also show that NNs perform best compared to traditional approaches.

6.1 PELARS Project

PELARS is a project about learning and making. It studies how people learn about science, technology and mathematics when they use their hands as well as their heads. A big part of the project is making more explicit the implicit practices of science teachers: "Lab demos" and hands-on experiments have been a big part of science teaching for as long as anyone can remember, but how to model and analyse these practice, while empowering teachers, is far less understood. So, the PELARS project aims at finding ways of generating "analytics" (data about the learning process and analysis of this data), which helps learners and teachers by providing feedback from hands-on, project-based and experiential learning situations.

There are many tried and true practices in the teaching and learning of science, technology, engineering and mathematics (STEM) that involve experiential, practice and hands-on learning. Science and engineering teachers

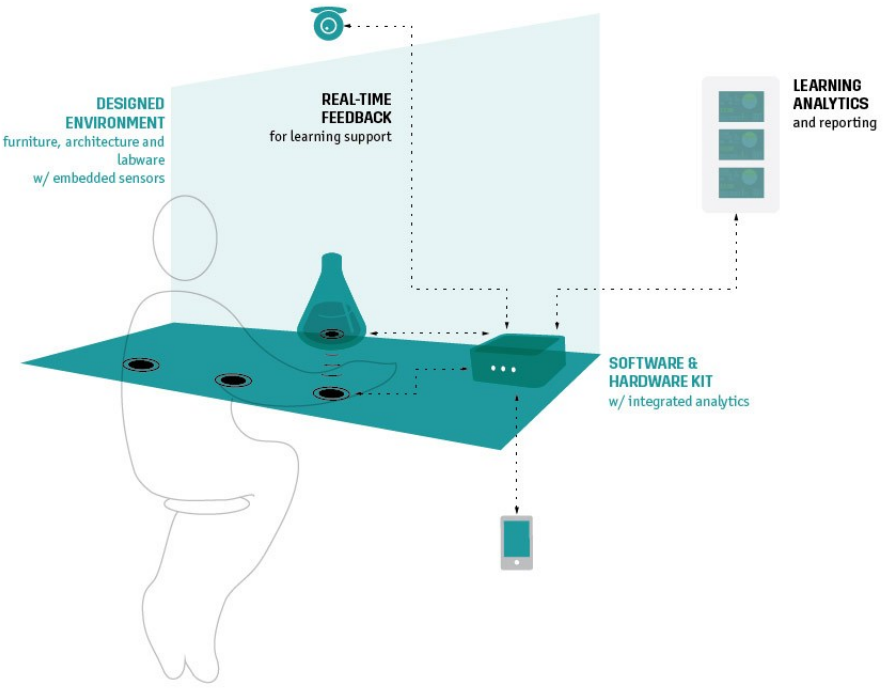


FIGURE 6.1: Mock up of the PELARS system[1].
[1] http://www.learningmaking.eu/about-pelars/

understand the importance and value of the "demonstration" and the "laboratory", although many of these practices are historical or traditionally routed, and unstudied. Coming less from a tradition of constructivist education and embodied learning, these laboratory and experimental, project-based learning techniques come "from the lab", and have an immeasurable, if known, value to the nurturing and training of creative STEM thinkers and doers. Further, particular trends in Do-It-Yourself learning of STEM subject matter (through systems and platforms, communities and resources online and off) prove inspiration for understanding and designing systems for learning support in more convention contexts of European education (high-school and undergraduate engineering education, as well as design and human-computer-interaction contexts).

The PELARS project goes through three phases. First, we look at the practices of teachers and teaching institutions. How do the best STEM teachers do what they do? What are the specific values embedded in, and value of, the "demonstration", the "practical", the "laboratory" both in terms of institutional emphasis (how teachers are supported) as well as individual learning processes? Secondly, the PELARS convenes over a period of iterative design phases, explicitly derived from the constraints and inspiring practices of

real teachers, where propositional design prototypes and experiments are tested and assayed with teaching and learning communities. This phase includes designers and design researchers who create laptop-electronics, and interactive kits for learning, as well as mobile learning and physical environment (furniture and interior design) designers, pedagogy experts and curriculum development partners, see Figure 6.1. Finally, the results of iterative design through the second phase will bring us to a trial of PELARS systems in real classrooms, workshop environments and educational milieu.

6.2 Background

The roots of project-based learning extends back almost a century to John Dewey's approach that argues for "laboratory schools" in which students are engaged with the process of enquiry in their learning activities. It is important to define the concept and explain its main features. Project-based learning is a form of situated learning, in which students engage in real-world activities that are similar to the activities that professionals engage in [87]. Project-based learning activities that support learners' participation in open-ended tasks are one of the most commonly used teaching approaches for improving 21st century skills [12], and they emphasise the engagement of learners in projects that are personally meaningful and they encompass driving questions, investigations, and collaboration [86]. However, the hands-on and open-ended nature of project-based learning creates challenges for tracking the learning process. One of the key challenges faced in project-based work is the support of the group work and ensuring that students succeed in the planned learning outcomes [19, 87].

Current research in multi modal learning analytics (MMLA) focuses on better understanding the complexity of learning through the advances of high-frequency multimodal data capture, signal processing, and machine learning techniques [114, 127]. MMLA offers an opportunity to capture different insights about learning in project-based learning tasks in which students have the opportunity to generate unique artifacts like computer programs, robots and small-groups collaboration to solve open-ended tasks [18, 16]. MMLA builds upon multimodal human interaction, educational data mining and many other fields that include learning sciences and cognitive sciences to capture the complexity of learning through data intensive approaches [168, 140].

In terms of the focus on purposes and context, there is an emerging body of work with MMLA to capture small group work on project-based learning

that has grown mainly out of the work of Blikstein and Worsley investigating engineering students'design activities [16, 25, 113]. Recently, within this research domain, Blikstein et al. [17] explored multimodal techniques for capturing code snapshots to investigate students learning computer programming as well as video and gesture tracking for engineering tasks; Worsley et al. [170] presented different approaches for data classification that included points about how these techniques have a significant impact on the relation of research and learning theories. Both of these initial approaches provided the means for other researchers to begin to explore MMLA with small groups of students across different subjects. In addition, notable data sets from the MMLA grand challenges workshop Ochoa et al. [113], presented the Math Data and Oral Presentation Quality Data Corpora that has enabled the community to analyse and discuss the different requirements and results within this field. Moreover, Ochoa and colleagues' work [112] used existing multimedia processing technologies to produce a set of features for accurate predictions of experts in groups of students solving math problems, which illustrated the benefits of MMLA to support students' learning in these contexts. Similarly, Chen et al. [25] expanded from the Oral Presentation Quality Data corpus to further examine the feasibility of using multimodal technologies for the assessment of public speaking skills; and Grover and colleagues [62] have explored how to develop computational models of social learning environments. In their work Grover and colleagues managed to classify the quality of collaboration from body movement and gestures of pair programmers working together with acceptable accuracy rates. Although most of the existing MMLA research approaches focus on learners' data, Prieto et al. [122] and Martinez-Maldonado et al. [99] have focused their research efforts on how MMLA can support teaching actions and orchestration in the classroom.

On the other hand, in terms of the technical focus, in order to make sense of complex data streams coming from multiple data sources, MMLA researchers employ various computational techniques. These approaches include logistic regressions, different feature reduction algorithms, and statistical models to investigate MMLA to identify features and predict student performances. These approaches all have advantages and disadvantages depending on the main research question and the purposes of data analysis and have potential to provide insights how to proceed with a multimodal data-set. Regardless of which computational approach is taken on board, it is clear to us drawing from the literature that MMLA has a role to play to

support education in project-based learning approaches and it has the potential to provide new means for gathering insights for complex open-ended learning activities which otherwise are extremely challenging to monitor and support with existing traditional standardised evaluation approaches. Lastly, small group work where students create unique solutions to open-end and complex problems provide challenges that span the technical, user experience, and new ways to support education are required for the research to contribute to practice.

6.3 Architecture

The PELARS Learning Analytic System (LAS) is based on a client-server architecture in which a number of remote clients acquire data during student's learning sessions and send the acquired information to a remote server. Here we refer to a single remote server but, due to the nature of the involved processing, the remote server can be easily realized as a cloud service. The LAS contains a subsystem called Learning Environment (LE) that corresponds to the location where the students are working on a project.

In this location the following elements have been identified: the furniture in which the LAS will be integrated; the Arduino Kit for the experiments; a Collector, which collects all the information gathered by the LAS and the Sensors, analyses the scene and extracts relevant information about objects and actions. The final results are available for the teachers through a Web Interface. The LE comprises a series of elements that contribute to the learning experience while, at the same time, collect information about the activity of the students. The sources of information for the LAS are constituted by sensors embedded in the furniture. In the following sections the details of the different blocks will be described, together with their interconnections.

Arduino Programming IDE The programming IDE, Figure 6.2, is the tool used by the students for interfacing with the code and developing the projects. PELARS designed, developed and produced a novel learning kit called Talkoo using a novel communication protocol called ESLOV, based on the Arduino platform, which enables working with modular hardware components through a visual programming interface. The base for the modular Talkoo kit is a hub that connects any combination of the other 12 modules with a computer. The amount of modules created for the initial Talkoo kit is easily expandable. We chose a series of boards that are very

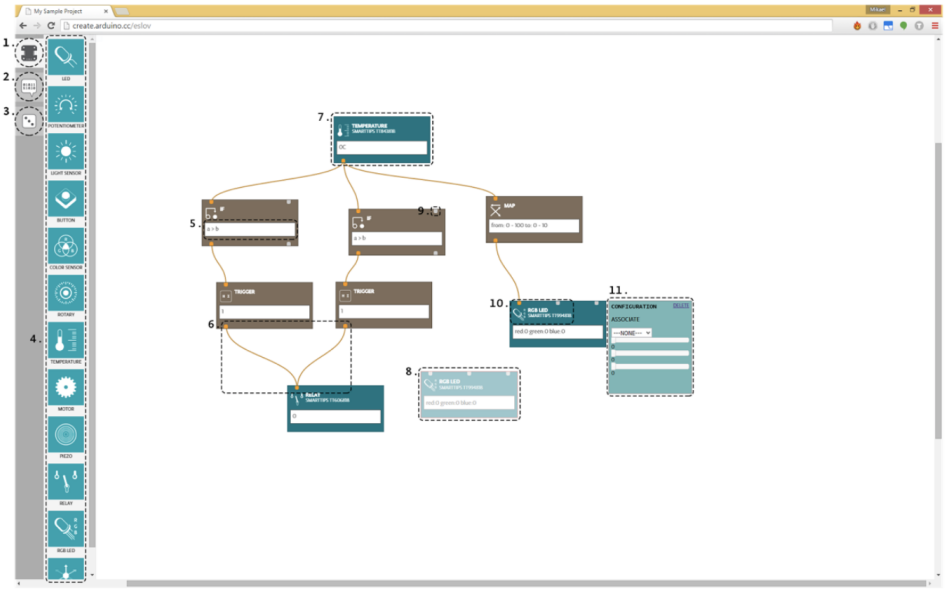


FIGURE 6.2: Image representing the Arduino Programming IDE.

commonly used when prototyping: relay, piezo buzzer, RGB LED, rotary encoder, potentiometer, button, light sensor, color sensor, temperature sensor, dc motor control, servo motor control and 6 axis EMU, see Figure 6.3. We created the firmware needed for each one of the modules as well as the hub in a way that allows for users to include as many units of the same kind (up to the theoretical limit of 127 units, limitation imposed because of the underlying protocol used: I2C). The processors chosen for the hub and the modules are different, while still being both from the same brand and family (ATMEL, ATMega). The capabilities of the processors allowed for the basic functionality required for the prototype: I2C communication, sufficient processing power to process the ESLOV protocol, memory to host the needed libraries and analog/digital pins to interface with sensors and actuators. All modules are re-programmable microcontroller boards. The main feature of this prototype is the possibility of addressing each one of the modules and negotiating all of the addresses when hot-plugging the modules on the fly.

The connection between modules at a logical level is controlled through the Talkoo Visual Programming Language (VPL). This software is a tool where users design programs through the connection of graphical elements: visual blocks and connection lines instead of text-based code. Each hardware module (excluding the hub) has a graphical representation on the screen, which, upon connection to the hub or to another module connected to the

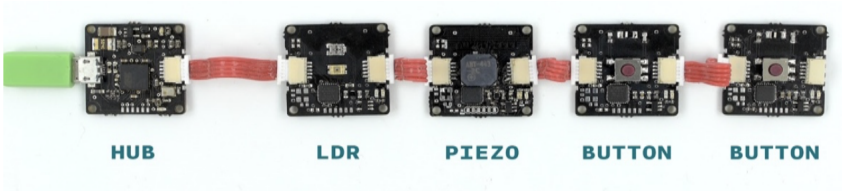


FIGURE 6.3: Talkoo components example.

hub, will automatically show up on the IDE's screen and make itself available for the user. Each block has inlets (inputs) and outlets (outputs) when needed. Users can literally draw connections between the outputs and inputs from different modules. There are also a series of logical blocks that will allow users to filter and manipulate data within the IDE and send it to other programs or over the Internet to a server. The VPL design served the purpose of giving users the smallest learning curve to get creative with this technology following the PELARS project requirements, type of students/users and use scenarios.

Desk and vision sensors Due to the exploratory nature of the project we opted for a two-camera setup configuration with a frontal Logitech c920 Webcam and a top-down Kinect v2 RGB-D camera. After calibrating the pose of these two cameras, it is possible to relate objects and motions with respect to the student's table. Single camera configuration could be considered in the future, but it would require larger field of view. The other, and more useful focus, is the one of the students that were tracked both in terms of hand motions and head motions. These two indicators can be connected to the measure of student's collaboration as they emerge from the Collaborative Problem Solving (CPS) framework developed in PELARS. For facilitating the tracking, PELARS adopted fiducial markers at student wrist that provide precise positioning at the cost of issues of occlusion. Further image processing could be possible for extracting specific interaction gestures and phases of student interactions.

We captured over time also the audio level at the table to understand how much learners are communicating. To do this we used the microphone incorporated in the used webcam and computed the power spectral density of the input signal.

While we researched and prototyped concepts for passive or ambient tracking of student behavior and patterns, at the same time we also wanted to give students tools to actively input their own perception of the current state of work. We wanted to know, not just what the students are doing, but

also, how they feel about what they are doing. Furthermore, our contextual research emphasized that it is often difficult for teachers to be constantly aware of each student's current status in their work because there are simply too many students in the classroom to orchestrate. We wanted to give students a way to communicate to their teachers how their project is going without the teacher needing to constantly check in with each group or individual. To this end, we designed a simple set of buttons that would allow a student to input their current status: the Sentiment Feedback Box. The Sentiment Feedback Box is a physical, internet-enabled box with two buttons. The left button is for inputting a positive sentiment and is represented with a light bulb icon on the front of the physical button. The right button serves to input a negative sentiment and is represented with a storm-cloud icon. The concept of positive sentiment is flexible and is intended to be used to denote a moment of success, a happy feeling or a bright idea. The concept of negative sentiment is similarly flexible and is associated with frustration, difficulty, or even failure. The data obtained through interaction with the Sentiment Feedback Box is constantly collected and aggregated by the Learning Analytics System, adding a richer, learner-driven perspective to the session.

Mobile Annotation System We developed a mobile annotation system which allowed students to take photographs, record video, and report via a form and free text their plan, progress, and reflective thoughts.

Collector The Collector is a piece of software responsible of acquiring the information produced by the different sensors and of elaborating the data in order to extract the Learning Metrics. The collector is a stand-alone application which collects and processes information from the various sensors (Vision and Desk) and the Arduino IDE. All the information extracted by the Collector will be sent to the Server deferred, or in real-time. It is written in C++ and consists in a series of threads and queues producing and consuming resources.

Server The Server is a dedicated machine hosting a Web Server and a database. Inside the Web Server three Applications will be executed: the Acquisition App that receives data from the Collector and stores into the Database, the LA Core App that performs computations over the acquired data and a Web App that provides the front-end to the user. The Web App provides mainly two types of services: the Administration of the LAS and

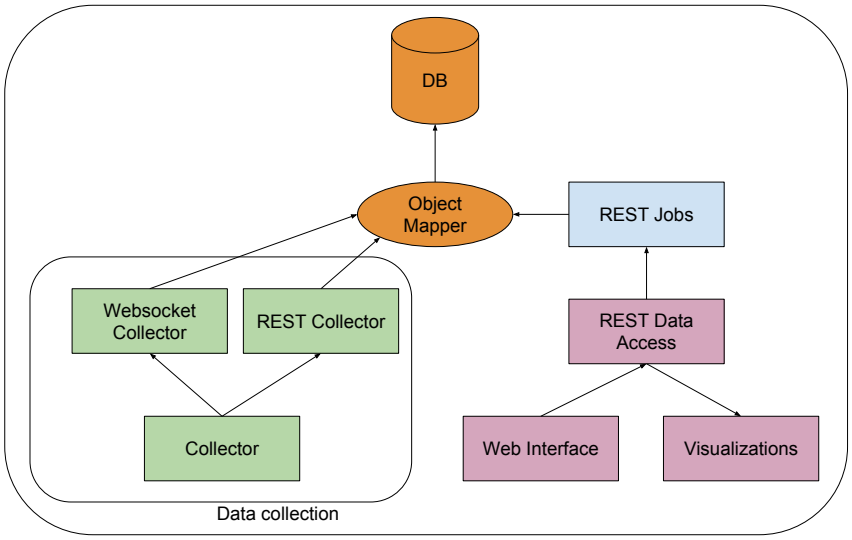


FIGURE 6.4: The general architecture of the PELARS Server.

the Visualization of the learning traces in the form of dashboard or custom visualizations. Figure 6.4 shows a general overview of the PELARS Server.

WebApp The Web Server is exporting a web interface, accessible from any computer, for Teachers that will be able to manage the collection of data from learning sessions and visualization of data. The Visualization is based on two concepts: the Dashboard that allows the users to visualize the key elements about a group of students, and a Traces Visualization that presents the information collected for a given user along.

6.4 Low Level Data Acquisition

Low-level data acquisition deals with basic face recognition and hand tracking as a way to assess whether the students are interacting with the system, looking one to another o handling objects present in the PELARS desk. Figure 6.5 shows an overview of the PELARS data acquisition architecture.

Face detection Faces are captured using the Logitech C920 Webcam. The webcam operates through gstreamer to be able to use the h.264 protocol which allows a very low latency on the processing of full HD images.

We exploited OpenFace, explained in 5.2, in order to get more detailed and meaningful feature to pass as input to the machine learning algorithms

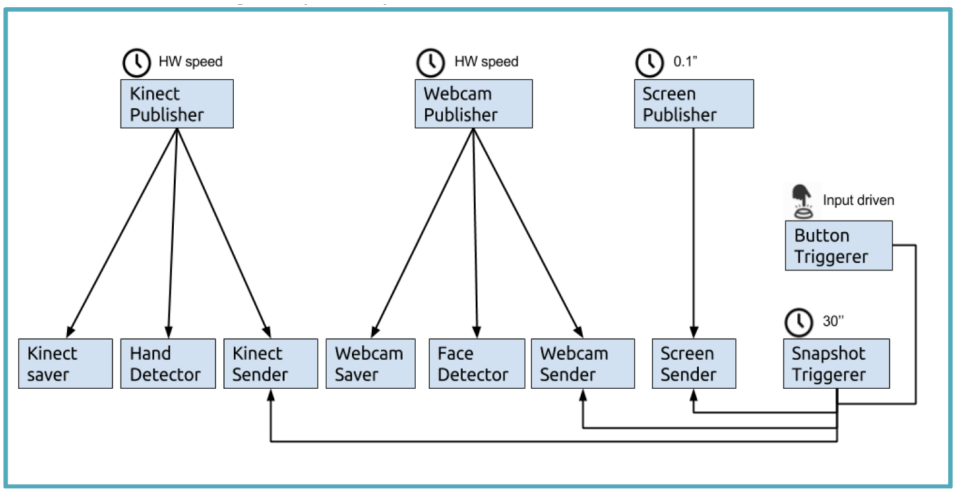


FIGURE 6.5: General overview of the PELARS data acquisition architecture.

developed for behavior analysis. The results are, for each detected face, the 3D head pose, the coordinates of facial feature points, e.g. chin tip, nose tip, lip corners etc. and 3D face model fitted to the face. The most relevant features we extracted using this software are, for each face: the pose, gaze direction and eye closure along with the 2D positions of the key points of the face, Figure 6.6.

The output of the face detection system for each image consists in an array of JSON objects with the following structure:

- 2Dfeatures: Facial feature points (2D coordinates). The 2D feature point coordinates are normalised to image size so that the lower left corner of the image has coordinates 0,0 and upper right corner 1,1. The feature points are identified according to the MPEG-4 standard (with extension for additional points), so each feature point is identified by its group and index. For example, the tip of the chin belongs to group 2 and its index is 1, so this point is identified as point 2.1.

- Eyeclosure_left: Boolean indicating whether the left eye is closed or open

- Eyeclosure_right: Boolean indicating whether the right eye is closed or open

- gaze_direction: Global gaze direction, taking into account both head pose and eye rotation. This is the current estimated gaze direction relative to the camera axis. Direction is expressed with three values determining the rotations around the three axes x, y and z, i.e. pitch,

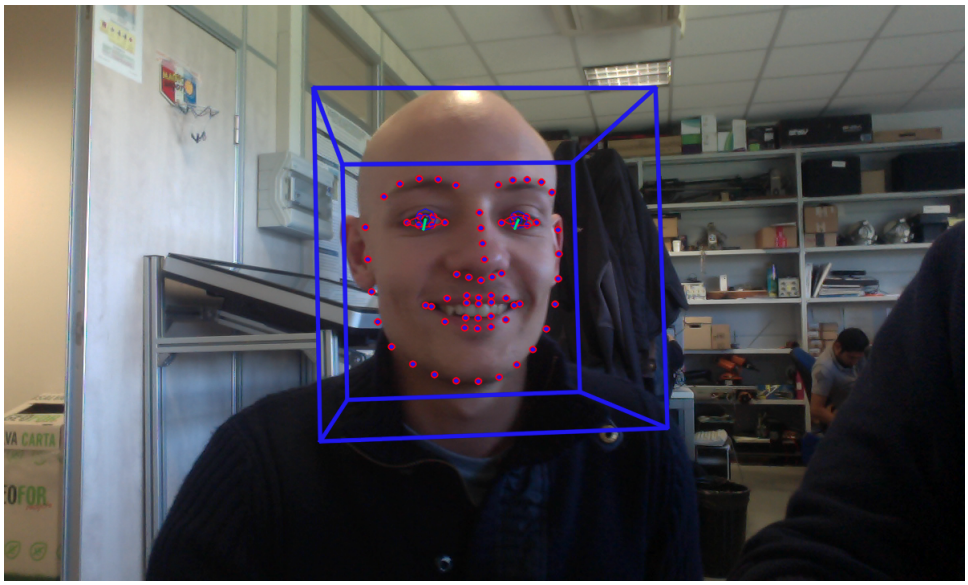


FIGURE 6.6: Detected face position and gaze estimation using OpenFace.

yaw and roll. Values (0, 0, 0) correspond to the gaze direction parallel to the camera axis. Positive values for pitch correspond to gaze turning down. Positive values for yaw correspond to gaze turning right in the input image. Positive values for roll correspond to face rolling to the left in the input image. The values are in radians.

- Pose: Composed by translation and rotation of the head from the camera. Translation is expressed with three coordinates x, y, z. The coordinate system is such that when looking towards the camera, the direction of x is to the left, y is up, and z points towards the viewer - see Figure 6.6. The global origin (0,0,0) is placed at the camera. The reference point on the head is in the center between the eyes. The returned coordinates are in meters. Rotation is instead expressed as a quaternion, obtained from the euler angles returned by the original algorithm.

The camera is set in the LAS in such a way that when faces are detected they are facing towards the screen. This allows to detect how many persons are using the Arduino IDE at the same time. The Collector also computes the distance of each face from the camera in order to avoid capturing faces of people which are further away than a certain threshold which can be set. To compute the face distance we assumed a fixed mean size for faces and then used the intrinsic parameters of the camera to compute the distance. Strong changes in face sizes could influence the system, but this has not been investigated.

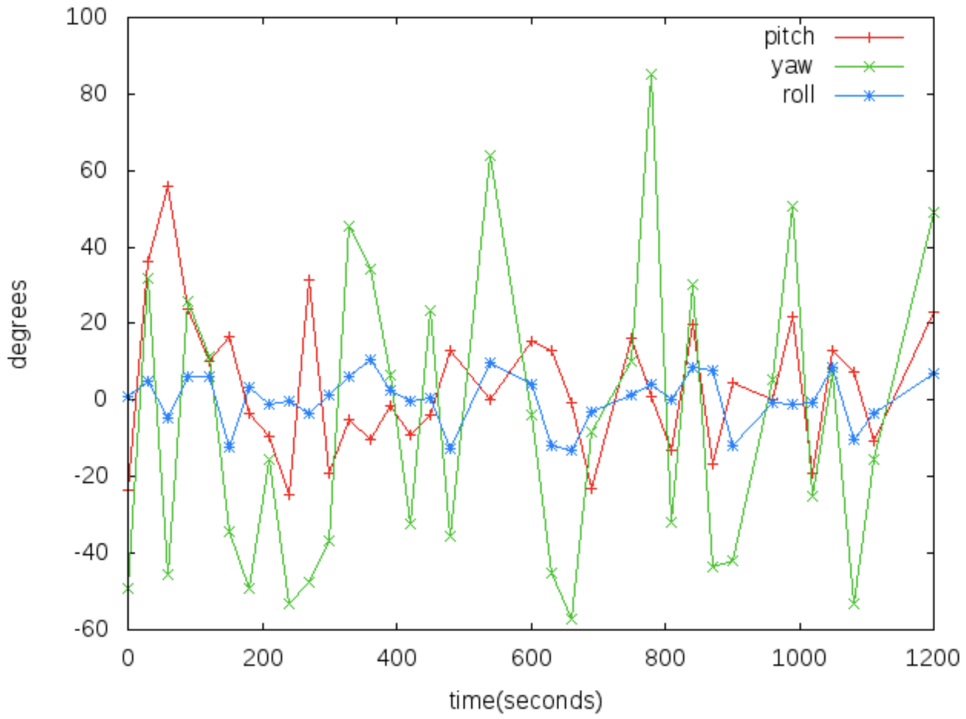


FIGURE 6.7: Angle of the head motions as corresponding to the snapshots taken by the system.

Figure 6.7 shows the gaze orientation of a single subject, expressed in euler angles, during a test session. Data is extracted from the snapshots taken during the session. Note that if all the three values are 0 then the subject is looking straight towards the camera. It is also important to say that, differently from head rotation [105], gaze directions represent where the eyes are pointing to, regardless of the head pose. Therefore, it is a powerful estimation of a student's level of engagement with the screen.

Hand tracking Hands are captured using a fiducial marker provided by the Aruco library [104]. Detection is done using partially the GPU to reduce the computational load on it. The library identifies the marker's unique id which allows movement tracking. All markers wore by users are unique in order to allow the correct evaluation of the captured data. Each hand position is referred to a base reference system positioned on the table, see Figure 6.8. The markers are attached to each side of the wristbands. This way we can track hands in almost all positions, as long as the markers are seen by the camera. For each marker, the relative 3D position with respect to the table is computed. This allows to relate faces and hands since they are positioned in the same reference frame (table). The system has to be calibrated initially to set the base reference system using the marker with id 0, which has to be used only for this purpose.

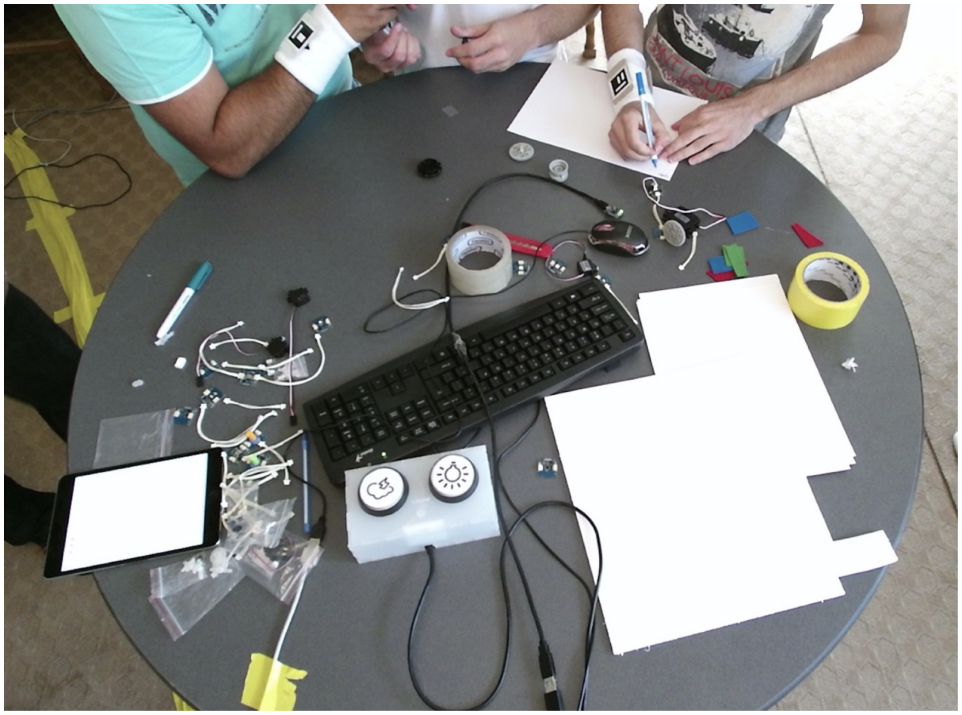


FIGURE 6.8: PELARS desk as seen from the rgb camera.

Object Recognition and Annotation The goal of the PELARS object recognition software is to track the location of objects as they are manipulated by the students. This can give us information about how students make use of the space, or how they distribute tasks among themselves: builder, programmer and documenter. 2D Objects can be captured from a video stream using a custom C++ tool that we developed. The tool is based on the BOLD descriptor (Binary Online Learned Descriptor) [7] which is used to recognize and track textureless objects. The tool works as follows: the user identifies in the first frame of the video the object, which has to be tracked. It is possible to associate a name to each object. The program segments the tables using a given mask taken as input parameter and then extracts the objects computing a unique descriptor for each object. The program identifies in each frame the different objects, and stores the position in a separate file as a JSON message. The program generates a new video stream in which each object is identified as a colored dot. To make the system more stable, the position of the object is passed through a low pass filter that stabilizes the center of the identified object. This creates a trade off between position precision and system reactiveness. The user can set the filter sample length as input parameter to adjust this value. The final output of the tool consists of a file containing Json messages that can be parsed to obtain for each frame

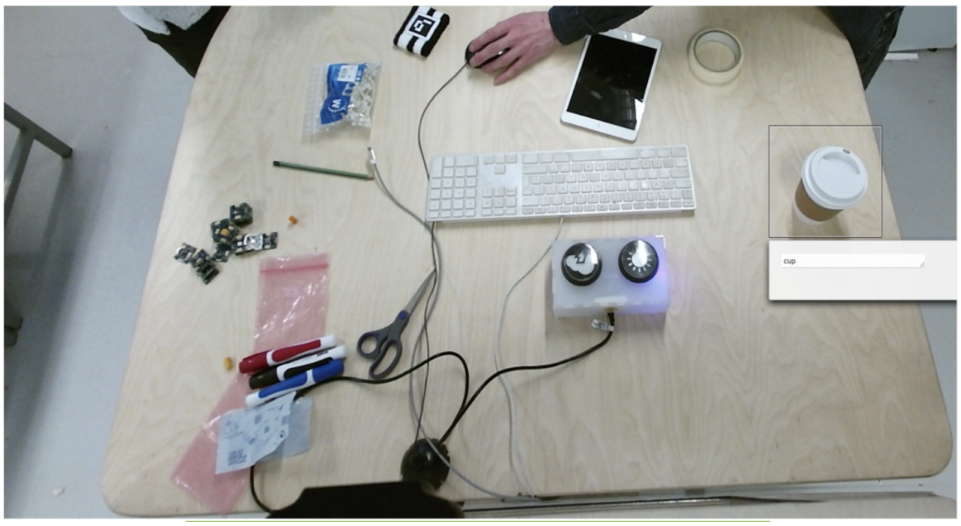


FIGURE 6.9: Example of the interface for the object tracking task.

the identified object and a video showing the center of the recognized object with colored dots, see Figure 6.9.

We did not use object recognition in the following experiments given the high variability of objects present on the table. Students were building different projects and using different tools for it, so it was not possible to find common useful objects to track.

6.5 ML Activity Recognition

Supervised machine learning approaches have been employed for associating the measured student actions with the resulting scores by the experts. In the following we use Deep Learning techniques and traditional Supervised Learning techniques to evaluate the outcome of projects and the possibility of inferring the current working phase.

6.5.1 Datataset Acquisition

The analysed data has been acquired in 3 sequential educational interventions with 18 engineering students at an European University (average age 20 years old, 17 men and 1 woman). The students were divided into 6 groups made up of 3 students. Each student group used the PELARS system over 3 days completing one open-ended design tasks for each session. First, the students were introduced to the system with a workshop to familiarise them with it, and then their first task was to prototype an interactive toy. The second task was the prototyping of a color sorter machine, and in the third task the students have been asked to build an autonomous automobile.

Each of these design sessions ranged from 60 to 80 minutes. As can be seen, each of the tasks introduced a more complex design concept to be solved with respect to the previous ones. Students were asked to perform an initial phase of planning, followed by execution/building and finally a documentation/reflection phase. During the activity the students had to document their planning, building and reflecting phase through the mobile annotation tool, see Figure 6.3. The research observers used the mobile tool to divide the students work flow into the planning, building and reflecting phases.

6.5.2 Initial Project Classification

To grade the students' design projects, a scoring scheme was developed that combined different approaches for collaborative problem solving (CPS) in small groups as well as bringing the design thinking principles. We started with the seminal work done with engineering students [4] that was initially adopted by [169] for multimodal learning analytics. From these initial frameworks, we began to develop a framework for CPS (blinded) that we could apply to the PELARS context. We used a version of our CPS framework with the mobile system and an agreed set of codes for on-fly observations to initially grading the student's projects. From the initial score of the students' work, the team of researchers reviewed the students' work collected in the LAS, which included snapshots of the students' plan, video of solutions and learners text input. The 18 sessions were graded with these criteria, where 50% of the grade was the expert's opinion based on the documentation collected by students, 25% was how the students planned and delivered the artifact and the remaining 25% was the student's own self-assessment of the quality of their projects. The resulting scores were categorised in three classes: poor, ok and good. This classification of the sessions was used as the reference point for the previous machine learning based classification work (blinded) in which the nature of this evaluation allowed only to reliably classify the works in two classes: good and bad.

6.5.3 Improved Project Classification

Based on the issues present in the previous scoring each of the sessions has been re-evaluated and re-scored by experts looking at videos, documentation (from the mobile tools) and final project outcome (the artifact). The aim was to generate a more rich scoring that reflected the learning practices for engineering courses. The new scoring has been based on 5 different aspects expressed in a scale from 1 to 5 and are shown in Figure 6.1:

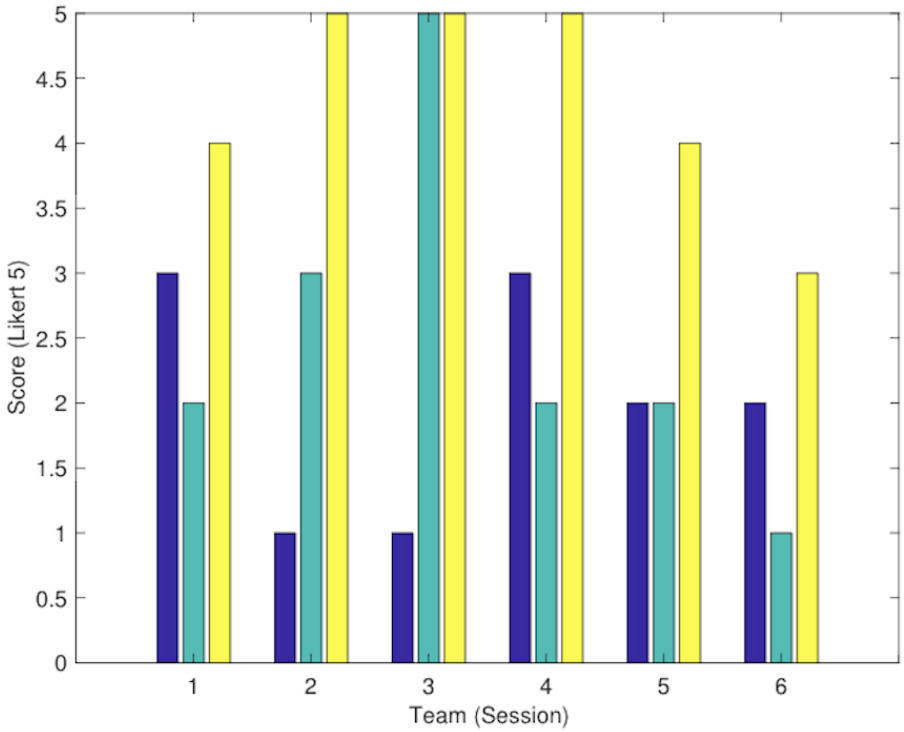


FIGURE 6.10: Quality of solution scores (QuaOS) of each team during the three sessions.

- Level of Clarity [Loc] (5=very clear, 3=legible, 1=not understandable)
- Independent Thinking [InTh] (5=independent, 3=based off instruction, 1=same as instruction)
- Corresponds with plan [CorPI] (5=Fully, 3=partially, 1=not at all)
- Does it Work? [DoWo] (5=fully, 3=partially, 1=not working)
- Quality of solution [QuaOS] (5=great, 3=mediocre, 1=poor)

6.5.4 Data Pre-processing

Data was collected at variable data rates (around 2Hz), yet it was not synchronised. For this reason, we needed a processing stage that aggregates indicators from the different variables in windows of same duration. The aggregation was performed based, for most of the variables, on counting for most of the variables. However, only for the distance/proximity features we employed averaging. Considering the fact that, students' sessions were different in terms of their lengths due to the open-ended nature of the project-based learning activities, we employed zero padding for sessions that were too short. For the investigation presented in this work, we tested

TABLE 6.1: Table of the 18 session scores organized by team. The five scores expressed in a 5 level Likert-type are reported

Team	Session	Loc	InTh	CorPI	DoWo	QuaOS
A	1	5	2	5	4	3
A	2	1	1	5	5	5
A	3	5	3	5	4	5
B	1	2	3	3	3	2
B	2	1	3	3	1	1
B	3	2	4	1	3	2
C	1	1	4	3	5	4
C	2	2	1	5	5	5
C	3	5	3	2	2	2
D	1	4	5	1	1	1
D	2	5	3	4	4	4
D	3	5	4	3	3	3
E	1	4	4	4	3	3
E	2	2	1	3	3	3
E	3	2	2	3	4	2
F	1	2	5	3	5	5
F	2	3	5	2	1	2
F	3	1	3	2	1	1

window sizes of 10, 20 and 30 minutes, and we also tested the case of one single window for the whole learning activity.

6.6 Method

A supervised machine learning approach has been employed for associating the measured students' actions with the resulting scores by the experts. In particular we have performed a two stage approach with different techniques. One assessment is based on large data quantities and uses DL for regressing the 5 scores by the experts. The second, based on traditional machine learning, deals with the simpler 3-levels assessment of the sessions and tries to address the problem of explaining the causes of the outcome depending on measured features and phases. Table 6.2 shows a synthetic view of the two tasks together with the inputs, outputs and details about the algorithms as discussed in the rest of this section.

6.6.1 Deep Learning approach

The input data is a set of timeseries that have different rates and partial synchronization. In this work we decided to use a windowing approach with dense network for compensating such difference, leaving the use of recurrent neural network techniques for future work. Given a session of duration T seconds we split it into non-overlapping windows of length L seconds (120, 240 and 360) obtaining $\lceil T/L \rceil$ windows. For a given input we

TABLE 6.2: Machine Learning Tasks performed over Data

Method	Deep Learning	Traditional
Task	Regression	Classification
Input	18 variables	9 variables per-window
Output	6 scores over 5 levels	1 score with 3 levels
Metrics	Regression Score	Classifier Accuracy
Windowing	120,240 and 360 seconds	10,20,30,90 minutes
Phase Exclusion	Reflection	Reflection
Method	Multiple layers	NB, LR, SVML, SVMR

compute an aggregated statistics for each window (averaging or summation). The following aggregated statistics (18 values in total) have been employed:

- Total number of faces looking toward the screen FLS
- Total number of connected Arduino components IDEC
- Mean distance between faces DBF
- Mean distance between hands DBH
- Mean hand movement speed HMS
- Mean audio level AUD
- Software blocks used IDEVSW
- Variety of hardware IDEVHW
- Number of interconnections between blocks IDEX

The total input consists of a series of 18 dimensional vectors consisting of the metrics indicated above and depending on the time chosen to window the data.Three different window sizes have been tested: 120s, 240s and 360s.

Deep learning has been tested to check the feasibility of non-linear regression on the input data gathered from the sensors. For the purpose of this experiment regression has been used since the output values can be a set of continuous values. The network has been implemented using a Python library for deep neural networks called *Keras* [30]. This high-level library allows to abstract the use of the GPU optimized processing libraries *Tensorflow* [1] and *Theano* [14].

Given the input data, we tested a fully connected network that was trained to fit a function that has an 18 dimensional domain and a 6 dimensional co-domain. Several additional DNN parameters have been tuned to obtain the best possible solution along with the window size for the input data creation. These parameters include:

- Dropout
- Regularization
- Epochs
- Layers

Input data is randomly split, as usual, in training and test data, with an additional minor split of the training data into training and validation. In these experiments 20% of the sessions are removed as test sessions, leaving 80% for training. Of this 80% another 20% has been used as validation set during the training phase. It is important to notice that complete sessions have been left out for testing and not just random inputs (windows) since they are usually correlated and could alter the final results if used.

The results of the net are evaluated using a mean squared error distance between the predicted value vector and the true value vector obtained in the test data set. A mean squared error has also been computed for each of the six output values along with the variance in order to understand if any of the output values had a different behavior. Three different NN architectures have been tested, growing from one to three fully connected layers of size 1024, 512 and 256. The best obtained net was created using the following parameters:

- Dropout 0.5
- No regularization
- 100 Epochs
- 3 Dense Layers of size 1024, 512, 256
- 240s Window size

The network structure can be see in Figure 6.11.

6.6.2 Traditional Approaches

The supervised approach we used is based on a supervised classification task that matches the observers' scores. The purpose of this approach is to identify the data features that can support different score classifications that have been evaluated by human observers (experienced teachers) as poor, ok, and good. Among the different families of classifiers available, we tested various parametric ones namely Naive Bayesian (NB), Logistic Regression

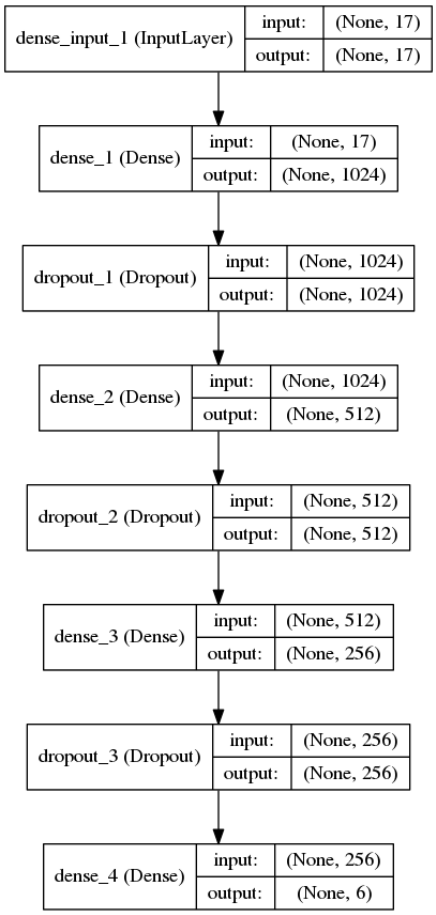


FIGURE 6.11: Neural Network structure of the model which obtained the best results

(LR) and Support Vector Machines with linear (SVML) and Gaussian kernel (SVMR). We avoided the non-parametric ones (Nearest neighbours) or decision trees with the purpose of reducing the overfitting effect. In particular, NB is a simple classifier that employs a strong assumption about features, a condition that holds valid for most of the variables we employed in our investigation except for the ones related to the Arduino IDE. We decided not to use the ensemble of classifiers [85], as we would like to study the model behind these classifications as much as performing the classification itself.

We used cross-validation (k=4) for understanding the effect of different parameters such as window size and the inclusion of different phases. Due to the small sample size (18 sessions from 18 engineering students working in 6 groups of 3 students) and the high variance of the data we avoided the leave-one-out scheme. The data acquired from the PELARS LAS was exported and then processed in Python using the sklearn [119] toolkit that provides state-of-the-art machine learning techniques integrated with a common interface. The test of the classifiers was performed by varying the window size, the score (binary or original 3-level), the inclusion of the different phases (planning, building, and reflecting) and, most importantly, the effect of features identified and described above (FLS, DBL, DBH, HMS, IDEC, IDEVHW, IDEVSW, IDEX, AUD)

6.7 Results

6.7.1 Deep Learning Results

The overall results for the different network structures are illustrated in Table 6.6, 6.3, 6.4 and 6.5 show the mean and variance for the error between expected output and predicted value. We then compared 120s, 240s and 360s window sizes. These sizes where chosen arbitrarily given a first rough estimation of the input data. The 240s NN achieves a mean squared error of 0.13 as shown in Table 6.4 across the improved classification of the student's outcomes. We then investigated the different features by removing them individually. In general, the results get worse as expected, see Table 6.7. This result illustrated that this feature of distance between faces is a substantial input for project-based work in the PELARS context. Additionally, the results show that the smallest window performs worse than the others, see Table 6.3. The network achieving the best results is shown in Figure 6.11 and is using a window size of 240s.

TABLE 6.3: Results for the 120s window, 0.242 overall

120s Window	Loc	InTh	CorPi	DoWo	QuaOS	OG
Mean	0.182	0.238	0.166	0.197	0.155	0.228
Var	0.074	0.112	0.069	0.076	0.061	0.099

TABLE 6.4: Results for the 240s window, 0.129 overall

240s Window	Loc	InTh	CorPi	DoWo	QuaOS	OG
Mean	0.086	0.175	0.150	0.175	0.154	0.084
Var	0.074	0.056	0.084	0.092	0.062	0.048

TABLE 6.5: Results for the 360s window, 0.193 overall

360s Window	Loc	InTh	CorPi	DoWo	QuaOS	OG
Mean	0.213	0.077	0.237	0.147	0.196	0.181
Var	0.097	0.006	0.083	0.063	0.071	0.057

TABLE 6.6: Best network results for the different network configurations

1024	0.186 with 360s
1024, 512	0.174 with 360s
1024, 512, 256	0.129 with 240s

TABLE 6.7: Best error scores after removing isolated features

Removed Feature	Best Result
No features removed	0.129
All faces data	0.21
All Arduino data	0.21
DBF	0.15
DBH	0.21
HMS	0.19
AUD	0.18
Hand pos	0.21
Arduino comp	0.19

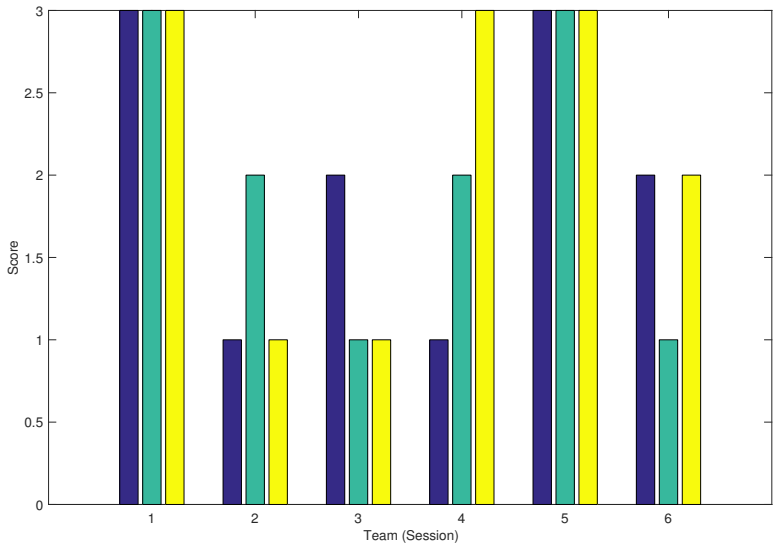


FIGURE 6.12: Resulting grades of the projects output developed by the tested groups of students.

6.7.2 Supervised Learning Results

6.7.2.1 Phases

Although, we had a small sample size of 18 sessions, the total amount of data generated from these sessions was rich and big due to the multimodal nature of our investigation. The project-based learning activities lasted within the range of 33 minutes to 75 minutes (median 63 min$\pm$13), with a total activity time of 17 hours and 10 minutes. Each project-based learning activity's project outcome was graded based on the criteria described earlier, and different patterns along the three sessions were observed. Figure 6.12 shows the different grades of the group's outcomes. The presented values are poor=1, ok=2, good=3 and the colors represent the different created student projects over the three sessions.

The design phases annotated by the observer (planning, building, and reflecting) varied broadly among the sessions as well as among the groups. The mean scores for the time spent on these phases among the sessions are planning (11min$\pm$10min), building (41min$\pm$16min) and reflection (4min$\pm$7min). Figure 6.13 shows the duration of each session and the timing of the phases for different groups of students.

6.7.2.2 Scoring

The three-level scoring we initially identified using human observation (poor, ok, good) posed difficulties to the classification activity, and we needed to move to a binary version in which we aggregated ok graded groups with good graded groups. For example, NB and SVM classifiers score 0.8 and

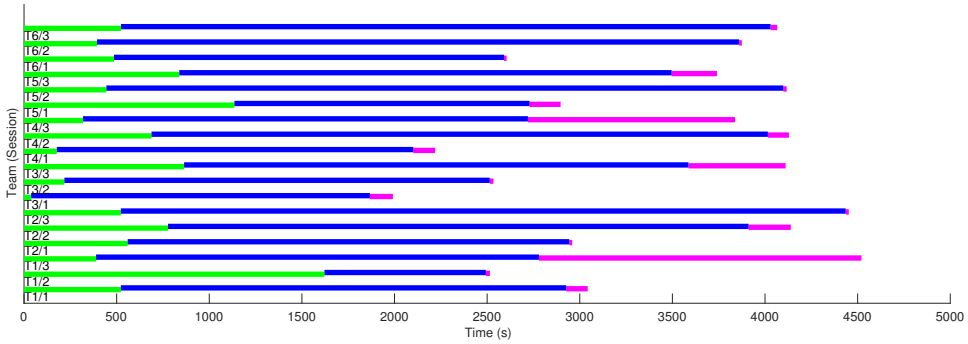


FIGURE 6.13: Distribution of phases among session of the 6 teams. Each session is split in the three phases, first plan, then build and finally reflect

TABLE 6.8: Effect of phases in the inclusion of the classifier. P=plan, W=work, R=reflect. Values are accuracy percentages.

	PWR	**PW**	**W**	**WR**
NB	0.8	0.8	0.6	0.75
SVML	0.6	0.75	0.75	0.8
SVMR	0.75	0.75	0.75	0.75
LR	0.6	0.75	0.5	0.6

0.75 respectively with a window of 30min and binary classification, however this value decreases to 0.5 for both of the classifiers when we use a three-way classification. This situation is clearly not ideal, however in order to achieve adequate results, we took this binary approach which still has great value to be able to identify project-based learning groups who perform poorly from others. Alternatively, it can be used to identify those group performances that are considered good in a binary fashion. This example shows the limitations of traditional ML approaches in respect to DL ones.

6.7.2.3 Effect of Phase

The selection of the phases used to train the classifier have a strong effect of the capacity to recognise the classifiers. For example, with a 30min window and binary classification, the exclusion of reflection (PW) phase in student activities provided stronger results across the different classifiers, while the exclusion of both planning and reflection reduced the classification power. See Table 6.8 for the details.

In order to provide the most reliable results and use the strongest classification power, we focus our results on data collected from the planning and working stages of the student activities excluding the reflecting stages.

As can be seen in Table 6.8, across the different tests of the classifiers NB is performing the best, followed by the SVML and SVMR. LR has the worst

results among the traditional classifiers. SVMR is the most consistent among different phases, discriminating among all always with an accuracy of 75%.

Having established the window size as 30 mins, grade classifications as poor vs. ok plus good scored projects, learning activity stages as planning and building phases, and the statistical methods we will use as NB and SVML, we now present the results of our analysis on the effects of the multimodal learning analytics features. We start from the full set of features with a given selection of the other parameters mentioned above and we proceed removing features, as a form of model selection.

Regarding the effects of the multimodal learning analytics features on predicting students' group performances in open-ended project-based learning, below results are found:

- IDEC (Arduino IDE) removal does not effect the results of the classifiers,
- Removal of all faces and hands duration has very little effect on the classifiers,
- Distance measures DHB and DBL **alone** are capable of predicting the results with a high accuracy (0.75) across classifiers,
- The audio level feature AUD **alone** is currently a **strong** feature for classification (1.0 with Naive Bayes) with time windows 5min, 10min and 30min and binary scoring.

Interestingly the logistic regression is capable of an optimal result (1.0) when considering IDEX, IDEVHW, IDEVSW and DBL, which are the main IDE features, except component counts and the distance between learners (DBL). One of the main limitations of our approach is the scoring of the sessions that is limited to a binary classification with respect to a richer 3-level human scoring.

6.8 Discussion

6.8.1 Traditional Approach

In the linear regression approach, we focused on identifying the different phases of work in relation to accuracy in predicting the group's artifact quality. We found that the planning and building stages of students learning activities are better predictors of their artifact quality than the reflection stage (in the intervention the reflection phase signalled the end of making

artefacts and coding to documenting with a mobile device the work). Then, we investigated the certain features of the MMLA, in order to determine which features can predict the student's artefact quality with higher accuracy. Our results show that the distance between hands and distance between learners are key features to predict student's performances in project-based learning activities. In our case, they highly correlate with the quality of the student's artefacts in project-based learning. These results are aligned with existing MMLA research findings that show the potential of hand motion and speed, and the location of the learners to predict student success at various learning outcomes [17, 113, 62].

There are three main aspects of PBL: students are asking driving questions, doing investigations to answer these questions, and collaborate together to solve these questions [86]. It is important that MMLA research aims to support these three main aspects of Project Based Learning (PBL). The results presented here show that the value of the distance between students' hands and distance between students are well aligned with the argument that closer students may fruitfully collaborate, which is an important aspect of PBL.

The other features of MMLA such as Hand Motion Speed (HMS) and Faces Looking at the Screen (FLS) did not perform very well to predict students' artefact quality across the classifiers. While the Arduino IDE the Number of Active Blocks (IDEC), the Variety of Hardware (IDEVHW) and Software Blocks used (IDEVSW) and the number of interconnections between blocks as a Measure of Complexity (IDEX) were able to predict students' outcomes, they were only marginal across the classifiers. Furthermore, the audio signal level (AUD) appears to be a promising feature to predict performance, however more investigation is needed for using this feature in combination with others.

6.8.2 Deep Learning Approach

The DNN results are more promising and show the feasibility of this method as an efficient approach for MMLA. In our investigation with this approach, we obtained a mean squared error of 0.129 with a window of size 240s as shown in Table 6.4. One important result that emerged from our results and is worth to notice is how the smallest window performs worse than the others, see Table 6.3. This is possibly due to the low information amount in that time window. The 240s interval performs the best, while the 360s

interval gives no performance gain as can be seen in Table 6.5. This suggests that the information gain from 240s to 360s is negligible for our purposes.

It is possible to see that (see table 6.7) by removing a single feature, in general results get worse except partially in the case of the distance between faces. This shows that this is a very strong input feature. It is also important to notice that the network learned some higher level features, which do not consist of a single input, given that by removing any single input we can not achieve the optimal results that we achieved using them all.

All results show a reasonably low variance evidencing the stability of the results, which is a positive sign in terms of the learned features. The fact that strong features have been trained is possibly due to the 0.5 dropout value, which "encourages" the network to find high level, strong features discarding the low level, weak features. Regularization gave no significant boost of the results, but this is probably due to the relatively "small" amount of training data, avoiding partially the problem of over-fitting. This parameter should become more relevant when more data will be added to the training set. A future step could consist in removing pairs or triplets of features to understand the relationship and importance of the input features further and make the factors on the learning process more visible. We aim to further investigate these in our immediate future work.

6.9 Conclusion

We showed that traditional ML techniques are able to solve some recognition tasks as for example inferring the phase of the students' work. Using the same instruments to predict the overall project quality gave bad results and we needed to switch to a binary classification, Section 6.7.2.2. This shows the limitations of traditional ML techniques. DNNs instead are able to predict correctly the complex outcomes of the projects without limitations as shown in Section 6.7.1. Distances between student's hands and faces, while they are working on projects, is a strong predictor of student's artefact quality. All this indicates the possibility of activity recognition in student collaborative projects, answering **RQ1**, **RQ3**, **RQ4**. This shows also that new and promising approaches such as DNNs, and more traditional regression approaches in some situations, can be used to classify MMLA data depending on the research questions and contexts being investigated, answering **RQ2**. Although, it is traditionally notoriously challenging to provide evidence about the robust and objective evaluations of project-based learning activities, techniques and types of data we presented here can be the first step

towards effective implementation and evaluation of project-based learning at a scale.

Chapter 7

Human State Evaluation

> "We're going to become caretakers for the robots. That's what the next generation of work is going to be."
>
> Gray Scott, [94]

Human internal state can be potentially provided by physiological signals which are difficult to interpret, partially due to technical limitations (e.g., availability of sensors, different sources of information), and their inter subject variability [152]. In the context of robotic systems we cannot rely on complex biological recordings, such as electromyography, complex laboratory setups, such as marker-based motion capture systems or sensorized hallways or even wearable sensors, which can obstruct the user during his everyday life activities. For these reasons the user internal state must be indirectly inferred from the available sensor sources, which usually consist in RGB-D cameras, range sensors and sometimes tactile sensors [148]. This task is non trivial since we often have to deal with sensors which are noisy and sensitive to light conditions or occlusions. For this reason these parameters are often assumed constant during the human machine interaction. To overcome these problems we try to evaluated the fatigue of MCI patients using ML techniques with depth images captured from a Kinect v1 sensor. This will answer **RQ2** and **RQ5** as part of the RAMCIP project presented in Section 1.3.

In the following we show that this it is actually possible to detect the internal state of a person with non invasive ML techniques, allowing assistive robots to infer the patients state in order to adapt their behavior dynamically. Figure 7.1 shows an image of the Ramcip Robot.

7.1 RAMCIP Project

The project RAMCIP is an EU Horizon 2020 funded project, under the grant agreement no: 643433. The project started on the 1st of January 2015 and will last for more than 36 months.

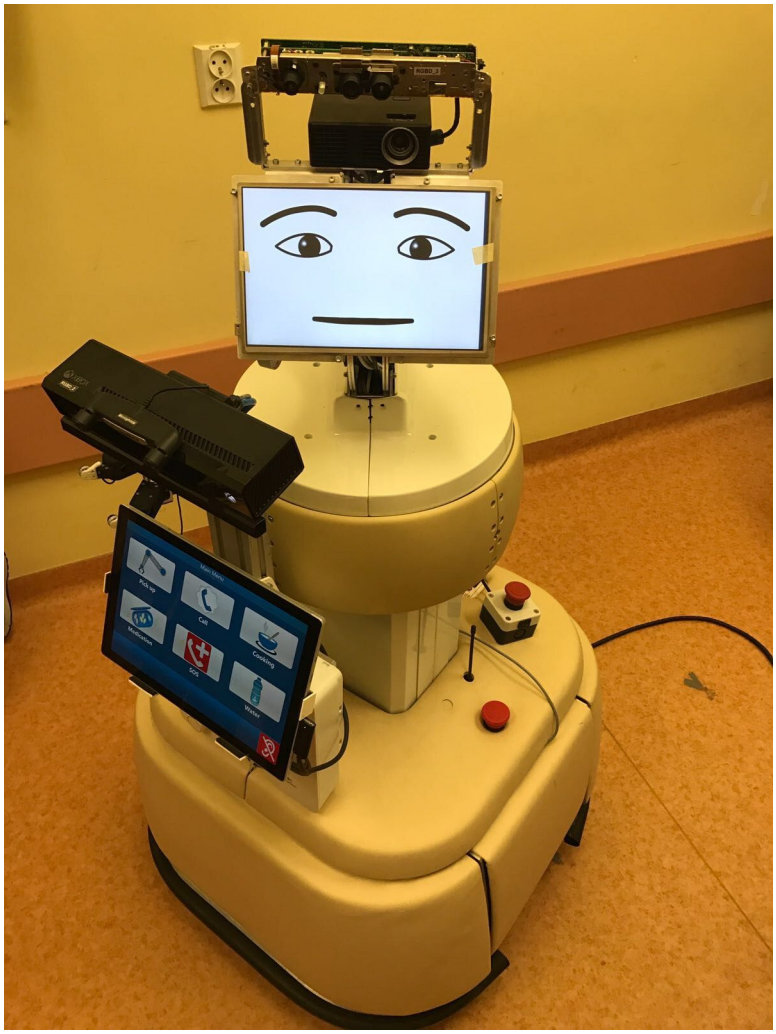


FIGURE 7.1: The Ramcip Robot.[1].

[1] http://www.fundacioace.com/en/ramcip-segunda-reunion-anual/

RAMCIP aims to research and develop real robotic solutions for assistive robotics for the elderly and those suffering from MCI and dementia. This is a key step for developing a wide range of assistive technologies. We adopt existing technologies from the robotics community, fuse those with user-centred design activities and practical validation, with the aim to create a step-change in robotics for assisted living.

According to the RAMCIP vision, future service robots for the assisted living environments of MCI and Alzheimer's Disease (AD) persons should be capable of providing safe, proactive and discreet assistance in a series of significant aspects of the user's daily life, ranging from food preparation, eating and dressing activities to managing the home and keeping it safe, both for the user and other persons, e.g. grandchildren, while at the same time, the robot should assist the user to maintain positive affect and also exercise cognitive and physical skills. It should be underlined that although for the latter, the focus of assisted living robot companions has so far been in initiating and managing specific physical or cognitive training interventions. The RAMCIP project foresees future robots to have this capacity embedded in their daily behaviour; i.e. providing such exercise subtly, by modifying the way they assist.

Working toward addressing the question of how a service robot can realize the above, the RAMCIP project will research and develop a service robot that will first of all have advanced high-level cognitive functions. These functions will be driven by thorough modeling and monitoring of the home environment, the user and other co-located persons, allowing the robot to take optimal decision regarding when and how to provide assistance, in a proactive and discreet way. Assistance provision will also be driven by enabling the robot to understand user commands and affect, through multimodal, adaptive and emphatic HR communication channels, see Figure 7.2

7.2 Objective

In order for RAMCIP to successfully realise its vision, several prerequisites are set in the form of major scientific and technological objectives throughout the duration of the project. The overall project success will be defined by the effectiveness and efficiency of the appropriate synthesis, as well as the individual quality of the specific achievements. The main objectives of the RAMCIP project are:

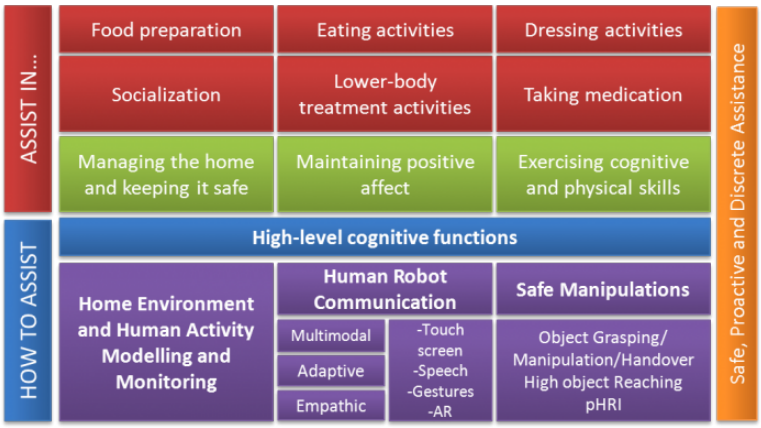


FIGURE 7.2: An overview of the objectives of the RAMCIP project.[1].

[1] http://www.ramcip-project.eu/ramcip/content/ramcip-vision

- **Objective 1**: To develop a service robot that will be capable of robustly understanding actions, complex activities and behaviour of multiple people in the user's home.
- **Objective 2**: To develop a service robot that will provide proactive, discreet and optimal assistance to the user.
- **Objective 3**: Establishment of advanced communication channels between the user and the robot.
- **Objective 4**: Establishment of advanced physical interaction between the robot and the home environment.
- **Objective 5**: Establishment of assistance activities involving physical interaction between the robot and the user.
- **Objective 6**: To validate RAMCIP project results in real-life scenarios. Evaluation of the RAMCIP robot prototype will take place in two pilot sites, in Poland and in Spain, through trials that will involve MCI and AD patients.

The majority of these objectives needs to evaluate the current state of the patient in order for the assistive robot to decide which actions to perform. To do this we acquired data from a set of patients affected by MCI and trained a DNN to discriminate fatigue and not fatigue by observing the patients walking patter.

7.3 Data Acquisition

The current study was approved by the ethics committee of the Lublin Medical University (Protocol Number KA-0245/2A/2016) and performed in accordance with the ethical standards laid down by the Declaration of Helsinki. All subjects provided written informed consent prior to the participation.

7.3.1 Subjects

A group of 20 elderly people (10 Male, 10 Female aged 71 ± 10.03) participated in the experiment. Subjects were suffering from memory impairments with different level of severity. The subjectively reported memory impairments were verified with Mini Mental State Examination Weighted Sum Score (MMSE WS) [51]. Participants assessed with values in the range 30-27 are considered as cognitively intact, between 26 and 23 as suffering from MCI, while lower values indicate the presence of dementia. All participants, either subjectively, reported increasing memory problems (age-assotiated memory impairments), or has been confirmed with such test. Table 7.1 reports the statistics of the subjects indicating age, sex, MMSE WS and the self-assessed level of physical fatigue reached during the experiment.

7.3.2 Sensors and Protocol

The experiment consisted in acquiring data while the subjects were walking on a flat ground in two states: fatigued and non fatigued. Each trial was composed of three phases.

During the first phase (*pre-fatigue*) the subjects walked back and forth straight, for up to 3 minutes, on approximately a 4 meter distance. The second phase of the experiment was based on the recommendation of the 6 minute walk test [47], which is a well known and standardized test for fatigue induction. The test consists in walking on a flat ground, at a self-preferred speed, for 6 minutes. The 6-minutes walking phase can be extended to multiple 6 minutes long rounds. The test has been previously successfully used among different groups of healthy and diseased participants [142, 98, 95]. Due to the specific health conditions of the participants, the initial 6 minutes intervals have been treated as an indicator, with the possibility of reducing the fatiguing time interval. For safety reasons, the aforementioned test has been chosen in order to minimize the potentiality occurrence of hazardous events. The evaluation of the fatigued state was self-assessed by the participant and assessed by a physician by measuring

Subject	Age	Sex	MMSE WS	Tiredness
1	78	M	23	7
2	75	F	27	5
3	64	M	27	0
4	71	F	19	1
5	88	F	12	6
6	68	F	19	3
7	80	M	30	0
8	60	F	27	6
9	67	F	28	8
10	86	F	26	7
11	69	M	25	1
12	69	M	14	1
13	74	M	26	5
14	57	M	17	8
15	89	M	19	7
16	85	F	24	1
17	60	M	24	6
18	60	M	26	4
19	63	F	27	8
20	64	F	25	3

TABLE 7.1: Age, sex and Mini Mental State Examination Weighted Sum score, and the self-assessed level of Tiredness (1-10) for every participant at the end of the trial.

the heart rate and the blood pressure of the subject and by checking the state of the patient with a sphygmomanometer. In the third phase (*post-fatigue*) the subject walked again back and forth for up to 3 minutes as in phase one. Participants were allowed to interrupt the experiment at any time.

Table 7.2 reports the duration of the three phases for every subject. The content of the table highlights the high degree of intraclass variability between the subjects. This is mainly due to the different age-related clinical conditions (typically cardiovascular and pulmonary diseases). Moreover, the MCI condition is often reflected in changes in balance and walking patterns [5] deviating from the regularity that characterizes the healthy behavior. This increased irregularity can partially mask the purely fatigue-driven changes making the classification task more challenging.

In particular, it is possible to group the behavior of the subjects in three categories: (1) full pre-phase followed by fatiguing (12 subjects: 2-5,7,9,11,13-14,17-20), (2) full pre-phase with fatigue at the end (5 subjects: 1,6,8,15,16), (3) interrupted pre-phase due to fatigue (2 subjects: 4,10). For the learning approach discussed later in the paper we have considered all the pre-phases

Id	Sess. pre (s)	Fatiguing phase (s)	Sess. post (s)
1	180	-	40
2	180	360	180
3	180	720	180
4	85	-	33
5	180	300	90
6	180	-	120
7	180	260	120
8	180	-	180
9	180	90	180
10	165	-	90
11	180	360	180
12	240	360	180
13	180	360	140
14	180	180	180
15	180	-	180
16	195	-	180
17	180	360	180
18	180	360	180
19	180	360	180
20	180	160	180

TABLE 7.2: Durations of the three phases for all the subjects.

as non fatigued situation without using the information of fatigued situation at the end of the pre-phase.

The RGB-D sensor employed for the data capture is a Kinect v1 camera running at 30Hz with a 640x480 resolution. We were constrained in the use of this camera given that it is was part of the specifications of the Robot developed in the RAMCIP project. The RGB-D camera was positioned at the beginning of the path pointing in the direction where the subject walked, with a minimum distance of 1 meter and an angle of approximately 20°.

The OpenNI2 library [33] was employed to process data coming from the Kinect sensor. To extract the full body kinematics in terms of the 3D positions of 15 body markers we used the Nite2 skeleton tracker by OpenNI which was presented in Section 3.3. The acquisition system is capable of tracking the human skeleton at 30Hz, which is the maximum camera acquisition speed.

We used the torso, shoulders and lower limb markers (feet, ankle and knee) to extract features which were used as input for the machine learning system. The ROS (Robot Operating System) [125] was employed to acquire and archive the data.

Marker Name	symbol
Head	$\mathbf{m}_{HE}$
Neck	$\mathbf{m}_{NE}$
Torso	$\mathbf{m}_{TO}$
Left Shoulder	$\mathbf{m}_{SL}$
Left Elbow	$\mathbf{m}_{EL}$
Left Hand	$\mathbf{m}_{DL}$
Right Shoulder	$\mathbf{m}_{SR}$
Right Elbow	$\mathbf{m}_{ER}$
Right Hand	$\mathbf{m}_{DR}$
Left Hip	$\mathbf{m}_{HL}$
Left Knee	$\mathbf{m}_{KL}$
Left Foot	$\mathbf{m}_{FL}$
Right Hip	$\mathbf{m}_{HR}$
Right Knee	$\mathbf{m}_{KR}$
Right Foot	$\mathbf{m}_{FR}$

TABLE 7.3: Markers provided by the skeleton tracker and the relative terminology used in this work.

7.3.3 Data Pre-processing and Features Extraction

We filtered the data provided by the OpenNI2 software at 3Hz by using a low-pass Butterworth filter [21] of 3rd order to reduce noise effects. After the filtering step, we computed a fixed reference frame on the estimated participants center of mass (CoM) for each skeleton sample. The body markers were expressed relatively to this new frame in order to make the trajectories independent from the Kinect camera position w.r.t the participant [54].

The OpenNI2 skeleton tracker provides always all body markers, even when they are not visible, and infers the position of the occluded ones. Each marker position is provided with a confidence level label: 0 for not found, 0.5 for inferred and 1 for tracked. During the data capturing phase, the confidence level of the relevant markers was almost always 1, except for the feet for which the tracker reported inferred state when the distance of the subject was far from the camera (5m) or, in few cases, when the subject was close to the camera (2m). The distribution of these values is shown in Figure 7.3.

In order to obtain a stable estimation of the CoM we employed a moving window approach. For each of the selected samples, the CoM was estimated as the mean of the centroid of the left shoulder, right shoulder, left hip and right hip markers over the whole window. We selected only the samples for which all these markers had a high confidence value for a time interval of

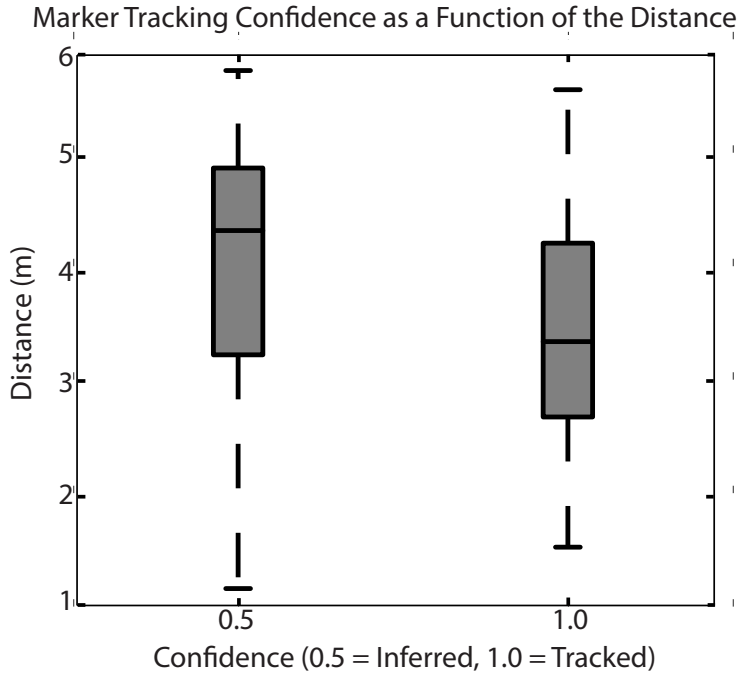


FIGURE 7.3: Confidence level of the tracker as a function of the distance from the camera.

size w_{CoM}, centered around the sample. We picked a value of 21 samples for w_{COM} after evaluating the stability of the CoM position.

Let $\mathbf{c}_K^C$ (where K subscript and C superscript indicates the Kinect and the CoM reference frames respectively) be such centroid. The estimation of the CoM frame, T_K^C, with respect to the Kinect frame has been geometrically computed from the torso, hip and shoulder markers. By computing the median along each component of the windowed velocity of the CoM position over consecutive samples, we obtained a vector whose direction is an estimation of the walking direction (WD) and whose norm is an estimation of the walking speed. Once the T_K^C transformation matrix has been estimated, we computed the markers coordinates in the CoM reference frame by multiplying each marker for the matrix $T_C^K = (T_K^C)^{-1}$:

$$\mathbf{m}_{xx}^C = T_C^K \mathbf{m}_{xx}^K \tag{7.1}$$

We selected only the trajectories of six markers as significant for our ML task. The selected markers, called *interest markers* hereafter, are: right foot, left foot, right ankle, left ankle, right knee and left knee:

$$\left\{\mathbf{m}_{FR}^C, \mathbf{m}_{FL}^C, \mathbf{m}_{AR}^C, \mathbf{m}_{AL}^C, \mathbf{m}_{KR}^C, \mathbf{m}_{KL}^C\right\} \tag{7.2}$$

as shown in Table 7.3. Note that the superscript C indicates that we express the coordinates (x, y, z) in the CoM reference frame.

An illustration of the CoM estimation is illustrated in figure 7.4. The trajectory of a marker, in the CoM frame, is represented (black). The selected samples, corresponding to a stable computation of the CoM, are highlighted in red, green and blue for x, y, and z coordinates respectively.

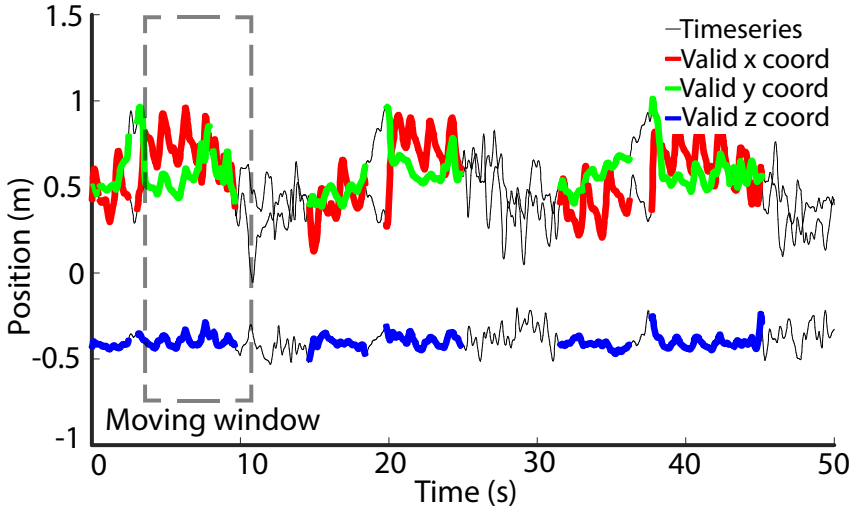


FIGURE 7.4: Time series of right foot x, y and z coordinates in the CoM reference frame during pre-fatigue task of subject one. For the whole x, y and z coordinated trajectories (black) the valid CoM estimation is highlighted in red, green and blue.

The resulting valid data percentages for both phases of each participant are shown in Table 7.4. The data selection process reduced the available data for the analysis to an average of 50% per subject in both phases, and minimum to 32% for some subjects, independently from the fact that they completed or not the session.

The windowing approach for the CoM estimation decomposes the time series of samples in chunks of contiguous samples. These are used as input for the ML task discussed in the following section.

7.4 Deep Learning Method

The objective of the ML task is to classify input gait data as fatigued and non fatigued. We used DNNs for this task given their wide use and the promising results detailed in Chapter 5.

To find the optimal deep neural network configuration for this task we explored several parameters based on the classification accuracy of the

Id	Valid Data % pre	Valid Data % post
1	46.6	48.1
2	49	55.8
3	53.2	42.3
4	48.8	45.1
5	51.6	37.4
6	42.7	40
7	55.6	57.5
8	38.2	32.8
9	55	53
10	44	34
11	46.5	38.3
12	58.6	60.4
13	73.8	77.1
14	55	46.4
15	70.5	57.5
16	46.6	34.2
17	51.4	46.7
18	35	48.6
19	44.5	48.4
20	62.1	58.8

TABLE 7.4: Dataset composition

network. We used the *Keras*[1] library to train our model. The training has been performed on a machine with 2 Intel Xeon E52650 processors running at 2.0GHz with 10 cores each, 64 GB of RAM and a Nvidia k20m GPU with 5GB of dedicated memory.

7.4.1 Data Preparation

The *interest markers* features discussed in Section 7.3 corresponded overall to a 18 dimensional space.

The data structure, after the pre-processing, consisted in a list of subjects containing each 2 different data fields, which are:

- a list of chunks of pre-fatigued data. Each chunk is a matrix of one row for each sample and the 18 columns of the aforementioned features.
- A list of chunks of post-fatigued data with the same structure as pre-fatigued.

Due to chunks having a variable number of rows, we employed a windowing strategy (with window size and data stride criteria) for decomposing each chunk into windows of fixed size. The effective input of the DNN is the

[1] https://keras.io/

flattened version of a matrix of size window size times 18. A window size of 60 samples (2 seconds at 30 Hz) and a stride of 15 samples (0.5 seconds at 30 Hz) were used. Usually bigger windows sizes may result in a higher recognition accuracy, since a longer observation of the subject usually results in a better prediction. On the other hand, we need to consider that, given a fixed window size, the robot has to observe the subject and track its skeleton reliably until a valid window of at least window size is recorded. Once this window has been recorded, the robot can predict the status of the user. The window size increases linearly with the latency that the robotic assistant will have in the prediction time. This could be an issue if a quick result is needed, since the system needs to accumulate valid skeleton data until window size samples are recorded. We found a 60 samples window to be a good trade-off value.

The following pre-processing has been applied to the input data:

1. data has been normalized in order to have zero mean and unitary variance.

2. The data have been split randomly into training and testing sets using 80% of the points for training and 20% for testing.

3. Training data points have been split into training and validation using an 80%-20% split.

4. For each input vector, an output label has been created. The labels consist of 2-dimensional on hot encoded vectors, representing the fatigued and non fatigued classes. All the output labels have been stacked into a final label matrix which was also split into training and testing accordingly to the input data.

7.4.2 Exploration of Parameters

The DNN has been trained testing several different parameters. The parameters were explored using grid search on a log scale with a one-fold validation approach. We preferred this approach over cross-validation because in the majority of the cases a single big enough validation set represents a good trade-off reducing the computational load compared to cross-validation.

- **Window size and data stride**. The two parameters are used during the data pre-processing to decide the temporal dimension of the windows and the temporal overlapping between consecutive windows (data stride). We performed a grid search increasing the window size from 1

second (30 samples) to 5 seconds (150 samples). At the same time we spanned the data stride from 5 samples up to the size of the window, increasing by 0.5 seconds (15 samples) at every step.

- **Number of hidden layers**. We tested several different hidden layer configurations. In the first architecture, the number of layers ranged from 1 to 9 with the number of neurons decreasing with the NN depth by a factor of 2. The second architecture had a hourglass structure, in which the number of neurons in every layer initially decreases with the depth and then increases again. This resulted in a DNN thick at the ends and thin in the middle [92]. In general, we chose to use fully connected layers, but several other architectures could be of interest, like RNNs.

- **Dropout**. We tested four different configurations, no dropout, 10%, 30% and 50%.

- **Learning Rate**. Learning rate determines the size of the steps we take to reach a minimum (usually local) during the gradient descent optimization. A too small learning rate leads to a slow convergence, while a too high learning rate can interfere with the convergence of the optimization problem causing the loss function to oscillate around a minimum or diverge.

- **Regularization**. L2 regularization is another form of regularization. It consists in adding a term dependent upon the squared magnitude of the net weights to the loss function, thus penalizing the weights magnitude itself.

As an additional exploration we also studied the effects of different training epochs and batch sizes. A low number of epochs could cause the optimization process to terminate too early without reaching the optimal loss while a too high number could lead to over-fitting the input data, loosing generalization. We found a good value to be 300. Thus, it was used for all the further tests. The batch size was chosen equals to 512 samples. The parameters exploration phases was performed using Adam as optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varepsilon = 1 \times 10^{-8}$ and null decay. All test have been carried out on the aforementioned machine using an automated script which tested the different configurations and saved for each one a configuration file and the trained DNN, along with the input data and the evaluation of the network on the total input set in order to gain statistics for each subject.

Hyper-parameters	Value
Dropout	0.0
Learning Rate	9×10^{-3}
Regularization	2×10^{-5}

TABLE 7.5: Optimal hyper-parameters for the NN.

7.4.3 Results

The proposed approach achieved a best accuracy of 78.3% on the test set using a NN with 5 fully connected layers. The input layer dimension depends on the window size and the number of feature, resulting in a total dimension of 1080. The optimal hyper-parameters are summer up in Table 7.5.

This configuration is optimal for a windows size of 2 seconds (60 samples) and stride of 15 samples.

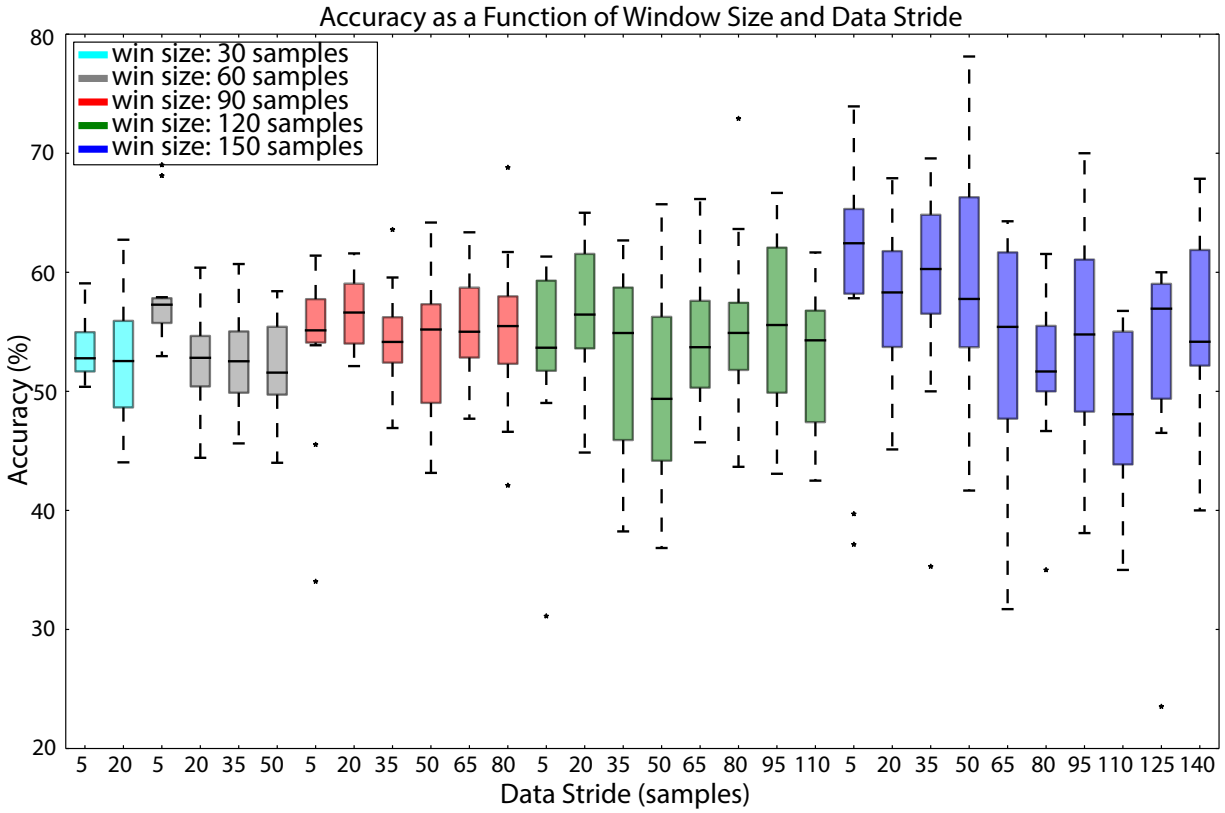


FIGURE 7.5: Accuracy as a function of window size and data stride. For every window size (30 samples: cyan, 60 samples: gray, 90 samples: red, 120 samples: green, 150 samples: blue) we explored different data stride starting from 5 samples to window size, with increments of 15 samples. The optimal values resulted with a window size of 150 samples and a data stride of approximately between one tenth and a third of the window size. Median, first and third quartiles are shown, whiskers show the 1.4 interquartile range values.

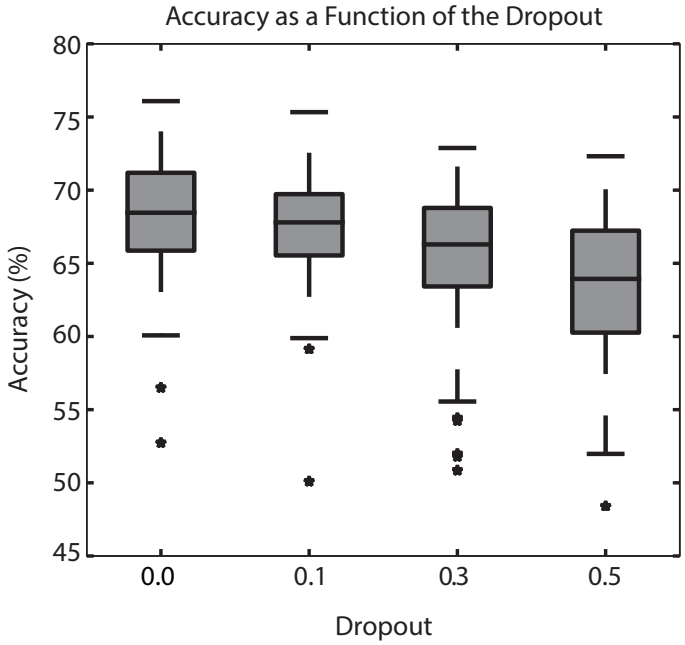


FIGURE 7.6: Accuracy as a function of the dropout. Median, first and third quartiles are shown, whiskers show the 1.4 interquartile range values.

Figure 7.5 shows the accuracy achieved by the NNs as a function of the window size and data stride. No significant difference in accuracy was found when using the two different NN architectures detailed in Section 7.4.2.

Figure 7.6 shows the accuracy achieved by the NNs as a function of the dropout. We found that for the task a null or a small dropout achieve a greater accuracy than a more aggressive approach.

Figure 7.7 shows the accuracy achieved by the NNs as a function of the learning rate. We found that the accuracy initially increases with the learning rate but it starts to decrease when the learning rate is too high.

Figure 7.8 shows the accuracy achieved by the NNs as a function of the regularization. Again, we found that a too aggressive regularization approach leads to a worse classification performance of the NN.

For each session of each subject, we computed the major prediction value (fatigued/not-fatigued) and used this to create a confusion matrix. The confusion matrices of the optimal NN for the training set and for the test set are shown respectively in Figure 7.9 (a) and (b) together with their F1 Scores.

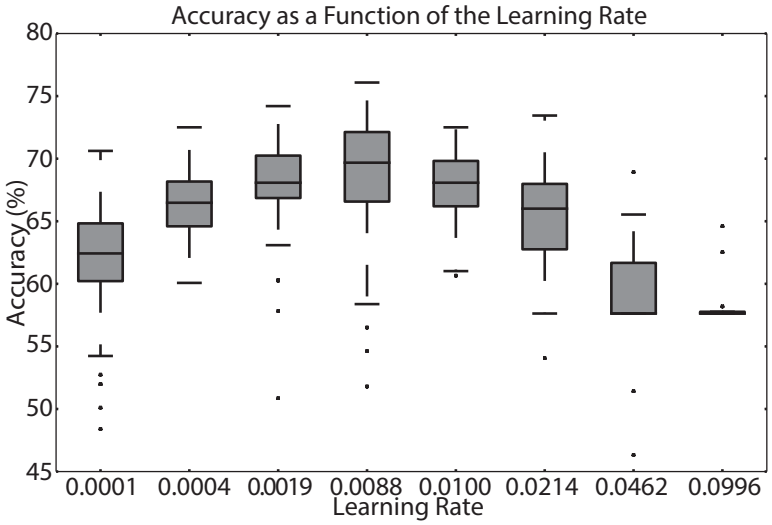


FIGURE 7.7: Accuracy as a function of the learning rate. Median, first and third quartiles are shown, whiskers show the 1.4 interquartile range values.

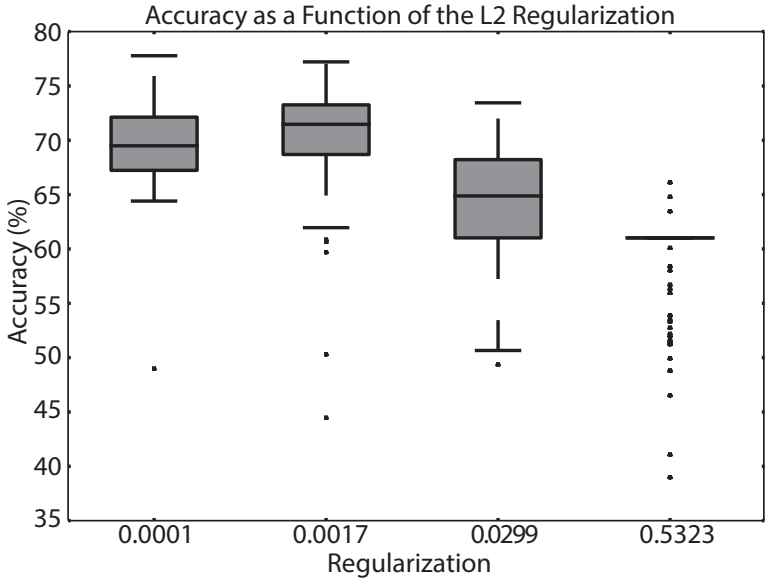


FIGURE 7.8: Accuracy as a function of the L2 regularization. Median, first and third quartiles are shown, whiskers show the 1.4 interquartile range values.

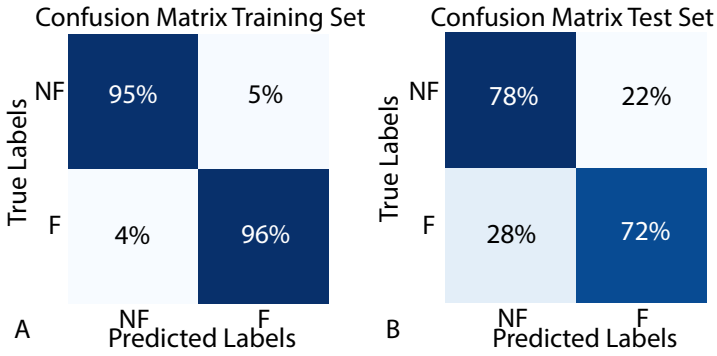


FIGURE 7.9: Confusion matrices for the training set with F1 Score 0.95 (A) and for the test set with F1 Score 0.76 (B). The classification outputs are non fatigued (NF) and fatigued (F).

7.5 Classic ML Method

We further assessed the performances of our approach, comparing the accuracy achieved by our model with two standard ML approaches: the decision tree (DT) [96] and Support Vector Machine (SVM) [171]. For the SVM we used a radial basis function (RBF) kernel and we performed a grid search approach to find the optimal C and γ parameters. The first parameter influences the soft margin cost function, while the second parameter is the inverse of the standard deviation of the RBF kernel, thus defining the influence of a training example. We found as optimal solutions $C = 7.84$ and $\gamma = 0.001$. The comparison of the classification accuracy on the test set between our approach, the DT and SVM achieved an accuracy of 63.4% and 75% respectively. The comparison between the ROC curves of the three approaches (SVM, DT and DNN) are shown in fig. 7.10.

7.6 Discussion

As detailed in section 7.3.2 some subjects did not perform the fatiguing phase of the protocol since they felt already fatigued at the end of the pre-fatigue trial. For these subjects, it is legitimate to assume that the onset of physical fatigue may have been occurred at some point during the first phase of the protocol. Figure 7.11 shows the classification results for three subjects (1, 4, 15). The dashed box represents the fatiguing portion of the trial; on the left we show the pre-fatigue phase (blue) and on the right the post-fatigue one (red). The blue and red vertical lines represent the end of the two phases respectively. The three subjects shown did not perform the fatiguing protocol. While for subject 4 the DNN shows an unsure classification behavior, for subject 1 and 15 it starts to clearly classify a fatigued behavior at the end of

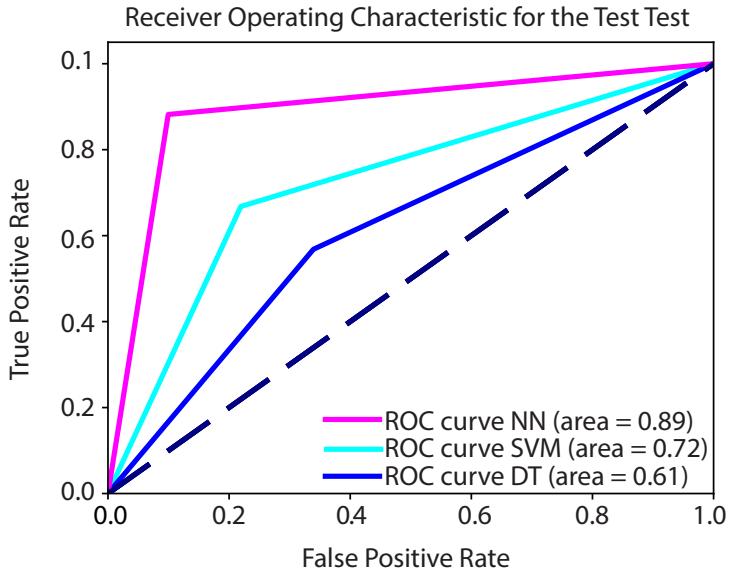


FIGURE 7.10: ROC curves for the test set for the three approaches SVM, NN and DT. The proposed approach exhibits a greater area under the curve, compared to the other classifiers.

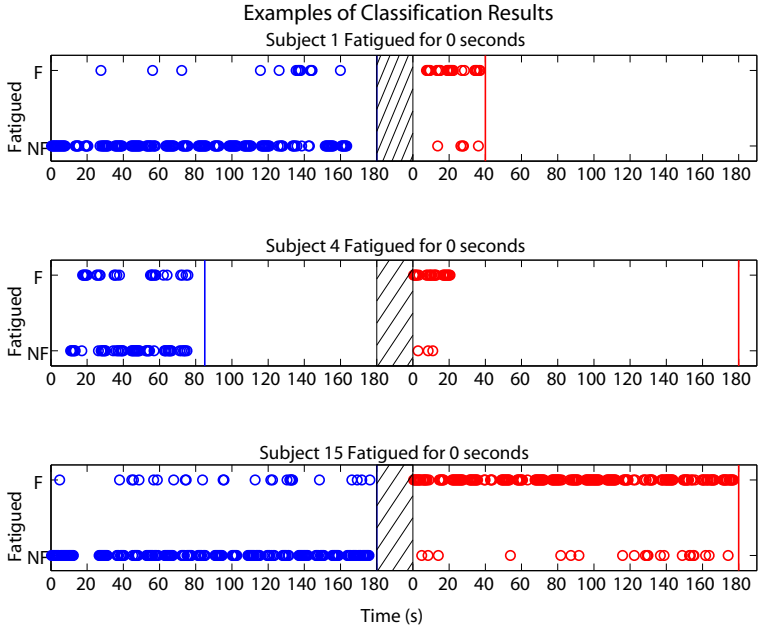


FIGURE 7.11: Classification results, fatigued (F) and non fatigued (NF), for three subjects: 1 (top), 5 (center), 15 (bottom), outputted by the trained NN. The dashed box represents the fatiguing portion of the trial; left represents the pre-fatigue phase classification (blue), right represents the post-fatigue classification (red). The blue and red vertical lines represent respectively the end of the two phases.

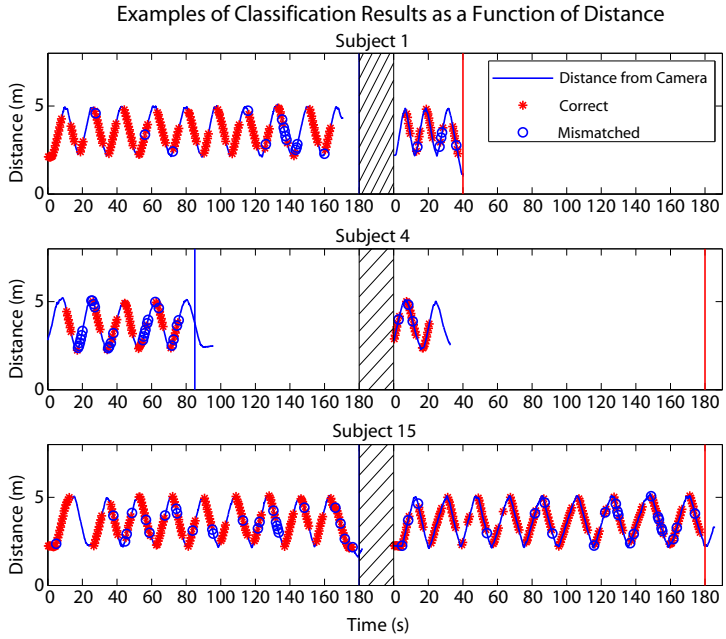


FIGURE 7.12: Representation of the correctness of the recognition as a function of the distance from the camera. Note that the sampling of the points along time is affected by the windowing and confidence level of the tracking.

the pre-fatigue phase. Despite the fact that this behavior was considered wrong, we could argue that the net is performing its classification task correctly, while the provided labels were wrong. These considerations not only further validate our approach, but may also introduce new possible ideas for future developments and investigations. We could, for example, train a new NN using as labels the classification results of the presented network for the uncertain subjects and validate it with the data from a new experiment.

7.6.1 Effect of Distance from Camera

As discussed in Section 7.3, there is an relationship between the distance from camera and the stability of the tracking system in particular relating the pose estimation of the feet. We verified a posteriori if there is also such an effect of the DNN classification capability, but overall there was not such an effect. Anyway for better understanding how distance and walking pattern can affect specific classifications we report in Figure 7.12 the distance of the subject from the camera with the results of the classification for subjects 1, 4 and 15.

7.6.2 Effect of Network Depth

The mean prediction time of our DNN on an Nvidia GeForce 960 is 2ms. Given that this video-card is not a high end card, we can easily assume that a better one could have even lower prediction times. This time is really low compared to the latency of the system (2s) and does not effect negatively the ability of the robotic assistant to detect promptly the fatigue level of the patients. In principle, it could be possible to train several nets for different window sizes in order to use increasingly longer windows, when available, to have an increasing accuracy. Performance can be increased even more by using ASICS as described in Chapter 4.

7.7 Conclusion

With this work we show that it is possible to infer human state using non invasive sensors and ML techniques. In particular we analysed the Detection and Classification process, which is depicted was depicted in Figure 1.1. We do this by inferring fatigue in MCI patients analysing their gait pattern as an example case of human state evaluation. Given that the patients have minor disabilities, the data is very noisy and unbalanced, making the task even harder.

To analyse the gait of the MCI patients we track their lower limb body pose using an ML technique based on RFs as explained in Section 5.3, showing that this system (Detection) is reliable enough to give us good results in the Classification step. This answers **RQ5**, leaving open the possibility to use more advanced sensors as the Kinect v2.

We tested DL techniques and classical ML approaches showing that DL gives the best results in Classification; we did this to answers **RQ2**. It is important to notice that ML are giving good results because the input data is very noisy and this techniques are able to filter it out finding strong features with high order non linear functions. Another advantage of ML techniques is that it is very hard to find hand crafted features which are good fatigue estimators, while in this case there is no need of such.

Chapter 8

Environment Recognition

To answer **RQ1**, **RQ2** and **RQ3** we tried to develop a system capable of augmenting the environment to allow navigation for autonomous systems. VALUE (Large Scale Voting-based Automatic Labelling for Urban Environments) was developed with the general idea of localising in 3D space any type of content (traffic signs, traffic lights, shops etc...) . In the specific case we tried to identify automatically in 3D traffic lights in maps of urban environments. Given that this system relies on the ability of triangulating precisely a 3D location from multiple 2D images with detections, we investigated the quality of several triangulation techniques to answer **RQ6**.

8.1 VALUE System

> "Some Google employees have their self-driving vehicles take them to work. These car robots don't look like something from "The Jetsons"; the driverless features on these cars are a bunch of sensors, wires, and software. This technology "works"."
>
> Tyler Cowen, [32]

The next generation of self-driving cars will likely operate more robustly by using maps of their environment [79]. These maps allow the robots to have strong priors on their environments to improve perception [8], and have metric and semantic components for localisation and planning (*top-left* and *top-right* in Figure 8.1) [111]. For self-driving cars in urban environments, these semantic maps typically contain static objects such as road signs, traffic lights, road markings, etc. It is common to manually label these [146], but on the city-scale this becomes prohibitively expensive, and furthermore it needs to be laboriously relabelled as the urban landscape inevitably changes. A system that can automatically generate such labels for entire cities without the need for hand-labeling would be very valuable to overcome this issue. In this work we present such a system.

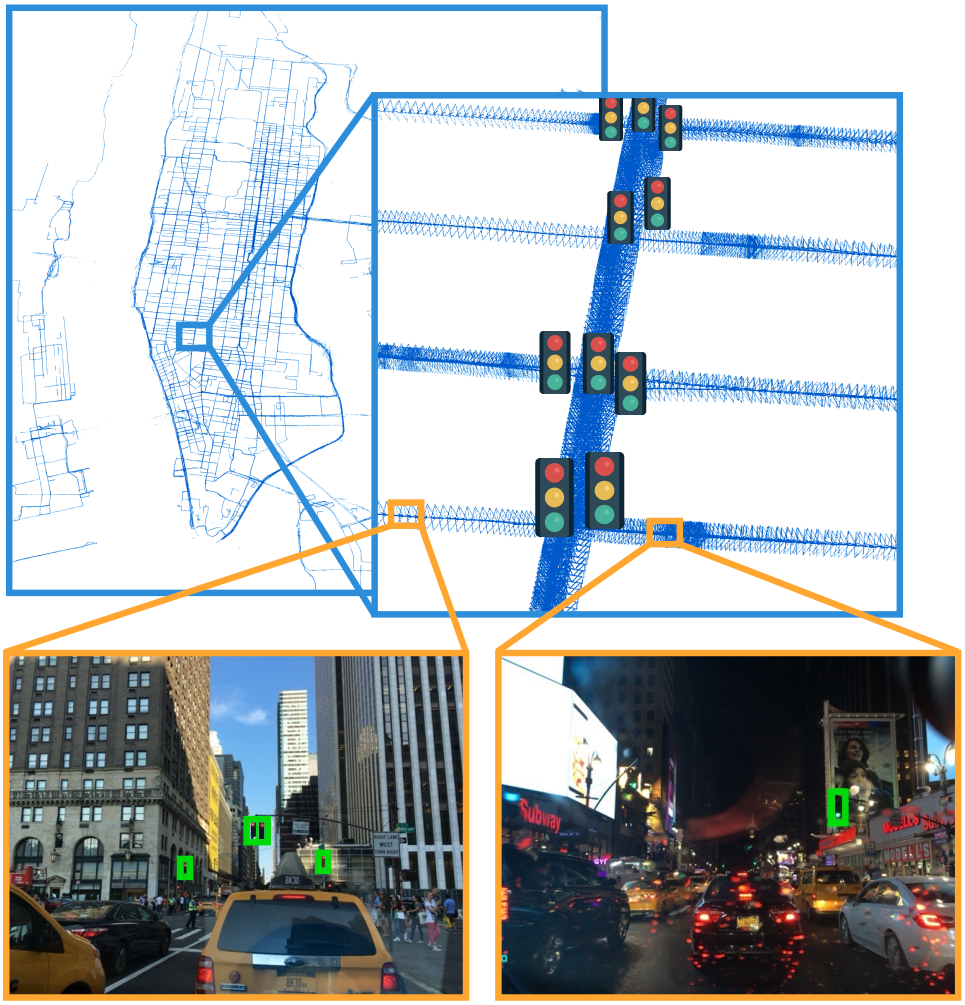


FIGURE 8.1: We use VALUE to automatically find the 3D locations of all traffic lights in Manhattan, such that they can be used for autonomous driving. Zero images were manually labeled to make this semantic map.

While the idea of using machine learning for automatic content detection has been explored before, the use of large-scale accurate maps together with machine learning opens new possibilities. Our work is inspired by the possibility to merge large amounts of noisily labelled but accurately localised data from a particular location. With this, we compute accurate, denoised estimations of the semantic information of a superior quality.

Our contribution is a novel and simple system that

1. takes images, their accurate 3D position in a large-scale environment and returns the 3D positions of static object in the environment

2. improves both object detection rate and 3D object position accuracy as areas are revisited, making it scale and work better with very large data sets

3. deliberately avoids the need for visually matching objects in-between the images, a problem that prevents most similar systems from working on objects that (a) inherently look similar to each other (like traffic lights) and (b) can appear very differently based on time of day, lighting, weather conditions, season, etc.

We evaluate our system on two new large datasets taken from San Francisco and New York City, in total comprising almost 0.4M images over 40 km^2 from different times and weather conditions over a period of several weeks to robustly recover position of the traffic lights in the environment. We demonstrate that significantly superior results can be obtained using even only mediocre and noisy 2D detection algorithms, if enough data are provided.

A number of works explore robust detection of the static 3D objects in the environment.

The basic component in a vision-based systems is an accurate 2D detection of the object in a single image or video. Recently, this approach has been dominated by DL techniques [81, 34, 118]. In this work we decided to adopt a full CNN as described by Huang et al. [72] to detect traffic lights in a large collection of pictures.

Given two detections of the same object coming from a stereo camera, it is possible to determine the 3D position of the object by triangulation [65, 91]. These approaches are viable, for example, for the online detection of relative positions of objects around the vehicle

Similarly, if the position of multiple cameras observing the same object are known, a multi-view triangulation approach can be used [64]. This has an advantage over live detection since a significantly higher number of potential views from different cameras can be collected over extended periods of time, resulting into potentially significantly higher performance. We take this approach in our work when we use large collection of 2D observations to produce accurate, denoised 3D output.

A common problem in applying such approaches relies in the need to accurately localise a set of sensors in the area. This problem can be addressed by using a highly-precise GPS system. Unfortunately, accuracy of GPS in dense urban environments is limited due to low sky visibility. An alternative approach in these environments consists in using a map-based localisation. Here state-of-the-art structure-from-motion systems have demonstrated ability to construct large-scale map of the environments [82]. We follow this

approach and construct a large-scale map of the city to localise accurately the positions of all the sensors.

The most closely related work to ours is [48] where they use, similarly to us, high-accuracy localisation of camera-equipped vehicles to map positions of traffic lights in the environment. Our work is novel for the use of a robust triangulation method operating jointly on the set of all the data from a particular location, resulting in improved performance as the amount of data increases.

8.2 Triangulation of Contents

There are several options for multiple view triangulations; we opted for the recent 3DTMA [41] approach due to the smaller re-projection error and speed. Here we present a test case in which we compared the different approaches. The variants tested are: Direct Linear Transformation (DLT), (un) constrained Non-Linear minimization (NLU, NLC), weighted least squares (WLSQ) minimization [1] and 3DTMA. The DLT is the classic formulation used for solving a linear system involving homogeneous coordinates. The non-linear formulation uses Matlab fminunc/fminbnd with the re-projection error as the objective function. The constrained version imposes that the solutions are placed on the semispace centered in the view origin and containing the frustum. The weighted least square minimization has a formulation similar to DLT but uses the distance of each view from the estimated point to weight the importance of the view in the solution. WLSQ is an iterative algorithm that provides good results also with 1 iteration.

TABLE 8.1: Result of the evaluation among the algorithms. The first two columns report the average and maximum triangulated error respectively, the third is the re-projection error in pixels and the fourth the running time of the algorithm.

	Avg Err. (m)	**Max Err. (m)**	**R-Err. (px)**	**Duration (s)**
DLT	4.61	147	2.21e+03	0.000894
NLC	17.1	31.1	64.1	0.329
NLU	40.4	51.8	5.34e+04	0.0234
WLSQ	2.11e-10	4.91e-09	8.48e-10	0.000679
3DTMA	2.53e-06	1.16e-05	3.58e-06	0.000489

The comparison is performed using a realistic configuration of 28 views taken from the 25k dataset spanning a space of 18 by 23 and 15 meters. We choose a set of randomly generated points around a real content location:

[1] used in OpenMVG

100 points have been sampled with variance 50m and then filtered removing the ones outside the image plate of the views. The views have different an average distance 68 ± 79 meters from the chosen point. The use of a known point set allows to evaluate triangulation without the uncertainty of detection. Table 8.1 shows the results across the triangulation algorithms and Figure 8.2 shows the configuration of the views and the target point. We decided to use 3DTMA given the possibility of using also speed as an additional parameter for future work.

The comparison is performed using a realistic configuration of 28 views taken from the 25k dataset spanning a space of 18 by 23 and 15 meters. We picked a set of images containing several traffic lights and choose a set of randomly generated points around a real content location: 100 points have been sampled with variance 50m and then filtered removing the ones outside the image plate of the views. Figure 8.3 whows an example of triangulated traffic lights. Table 8.1 shows the results across the triangulation algorithms and Figure 8.2 shows the configuration of the views and the target traffic light used in Figure 8.3. We decided to use 3DTMA given the possibility of using also speed as an additional parameter for future work.

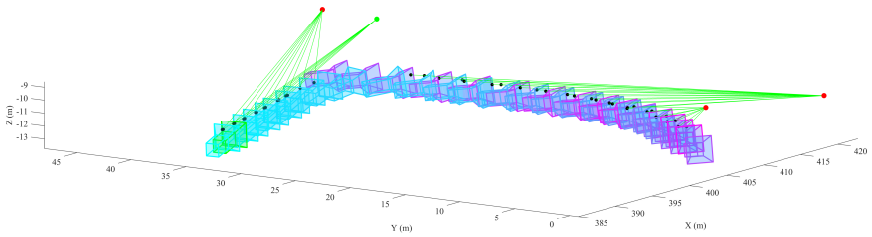


FIGURE 8.2: The 3D representation of a cluster of views with 4 contents shown as red dots. Each content is connected with the voters using a green line. The green view and the green content correspond to the one shown in the figure 8.3.

8.3 Method

Our system takes a large set of 2D images I_i, with associated camera-intrinsics parameters q_i and 6 degrees-of-freedom poses $P_i \in \mathbb{SE}(3)$, and produces a set of 3D positions of objects $L_i \in \mathbb{R}^3$. Figure 8.4 shows a general overview of the system. In our work the images are captured from our mapping fleet traversing various cities, and the poses are calculated using a large-scale structure-from-motion (SFM) pipeline [84], but in general there is no restriction on the source of the poses as long as they are accurate and globally consistent.

FIGURE 8.3: Image of the highlighted view and content of figure 8.2 showing the content projected over the detected position. The re-projection difference is 0.5 pixels. A detected content is a content which has been detected by the ML system, while a reconstructed content is a content which passed the voting scheme and has an associated 3D position.

The process then consists of two steps: (1) applying a noisy 2D detector to each image I_i resulting in a set of object detections $Z_i \subset \mathbb{R}^2$, followed by (2) estimating their final 3D positions L by a simple voting-based triangulation algorithm.

8.3.1 2D Object Detection

We generate 2D object detections in the images using an off-the-shelf CNN trained to predict bounding boxes for traffic lights [72]. These detections are usually noisy and suffer from many false positives and false negatives. In Section 8.4 we show that our system compensates for these noisy detections if shown a large amount of data. One alternative to using a detector is to use hand-annotated labels from, for example, *Amazon Mechanical Turk* [20], which however have also been shown to suffer from label noise [53].

8.3.2 Robust Voting-based Triangulation

The output of the previous step is a large set of 2D detections. Importantly, the 2D detection step cannot tell you which detections can be associated with which physical 3D traffic light D_i, and any feature descriptors that it might produce to associate them would be useless under the appearance changes

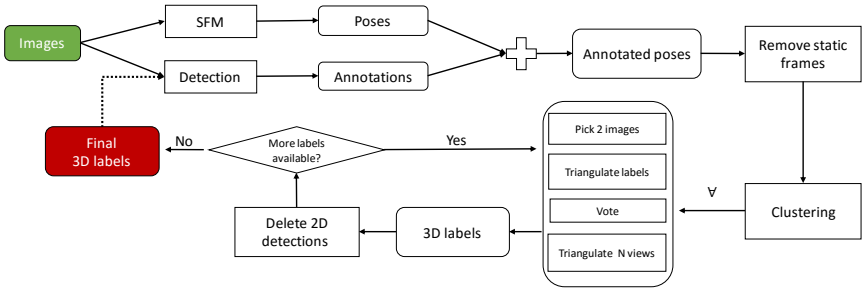


FIGURE 8.4: The overview of the system. The final output consists of a list of 3D detections. These can be passed back into the input of the system to obtain better results.

that we see in outdoor environments. This is true for any set of objects that look similar (traffic lights are a good example). The only difference between them is their position in 3D space. Without this association, classical algorithms for multi-view triangulation can therefore not be directly used. Instead, we use a robust voting-based triangulation algorithm to jointly determine these 2D associations and the position of the traffic lights in 3D space.

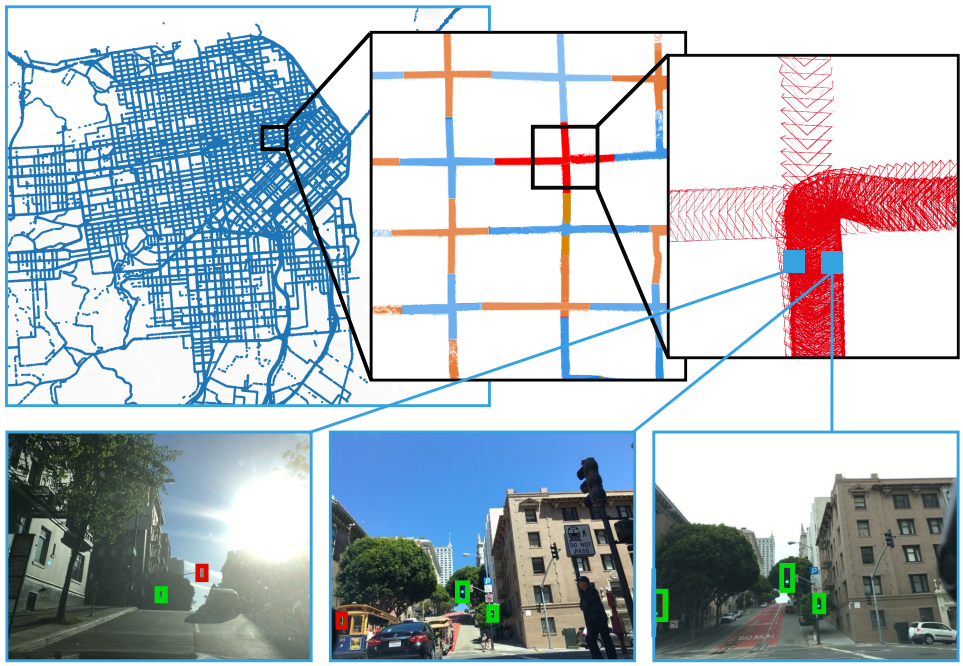


FIGURE 8.5: An example of a cluster in the map consisting of several traversals in differing conditions. *Top*: Close-up of the dataset clustering. *Bottom row*: example frames that are from different traversals belonging to the highlighted cluster and associated traffic lights.

.

For each pair of detections (z_a, z_b) (where a and b are indices into 2D detections) from two different images I_i, I_j we create a 3D hypothesis h_{ab}

Algorithm 1 Robust Voting-Based Triangulation

Input:	$\mathbf{I}$	*set of images*
	$\mathbf{Q}$	*camera intrinsics*
	$\mathbf{P}$	$\mathbb{SE}(3)$ *camera poses*
	d_{max}	*maximum reprojection error*
	α	*minimum ratio of inliers*
Output:	$\mathbf{L}$	*3D positions of objects*

Detect objects in 2D images:

1. **for** $I_i \in \mathbf{I}$
2. $\quad Z_i \leftarrow \textit{detect}(I_i)$
3. $\mathbf{Z} \leftarrow \cup_i Z_i$
4. $\mathbf{L} \leftarrow \emptyset$
5. **for** $(z_a, z_b) \in \mathbf{Z}^2$

   Compute 3D position of the object:

6. $\quad l_{ab} \leftarrow \textit{triangulate}(\{z_a, z_b\})$

   Compute inliers for computed 3D position:

7. $\quad S_{ab} \leftarrow \{z_k | \forall z_k \in \mathbf{Z} : \pi(l_{ij}, p_k, q_k) - z_k < d_{max}\}$

   Find the hypothesis with most votes:

8. $a, b \leftarrow \arg\max_{a,b} |S_{ab}|$
9. **if** $|S_{ab}| \geq \alpha \cdot mean(|S|)$
10. $\quad L \leftarrow L \cup \textit{triangulate}(S_{ab})$
11. $\quad Z \leftarrow Z - S_{ab}$
12. $\quad$ **goto** 5
13. **return L**

under the assumption that these two detections correspond to the same physical 3D traffic light generating in total $\mathcal{O}(N^2)$ hypotheses where N is the total number of detected traffic lights. The 3D position l^* of each hypothesis can be determined by K-view triangulation (in this case $K = 2$), where we minimise the sum of the reprojection errors:

$$l^* = \arg\min_{l} \sum_{k \in K} \left(\pi(l, p_k, q_k) - z_k\right)^2, \tag{8.1}$$

where K is {a,b} in this case, π is the projection of the 3D point l into the camera at position p_k with intrinsics q_k. We consider a hypothesis viable if it satisfies the following:

1. *triangulation constraint*: the point is triangulated in front of each camera,
2. *rays intersect in 3D space*: the reprojection error is smaller than d_{max},
3. *the projection is stable*: the angle between the optical axes is larger than θ_{min},
4. *distance to camera*: the distance from the traffic light to either camera is less than r_{max}.

Optionally, additional constraints reflecting prior information about the location of a traffic lights can be used to further restrict the hypothesis space.

For each hypothesis h_{ab} we compute the set of consistent inliers S_{ab}. This set consists of all the 2D detections that observe a traffic light at the same location, which is computed by projecting the 3D position l^* into each image and verifying whether the projected position is less than d_{max} to any 2D detection. Next, we remove the hypothesis with the maximum number of votes and also remove the detections that voted for it (inlier detections). This process is repeated until no hypothesis with at least $\alpha \cdot M$ inliers is found, where M is the average number of inliers per hypothesis and α is a tunable parameter over the confidence. This creates a set of confirmed hypotheses. An important theoretical property of this schema is that in the case of noisy but unbiased 2D detector and a uniform distribution of the data, it converges to the correct solution as the amount of data increases. This is due to noisy detections forming hypotheses with small numbers of votes, and correct detections gathering consistent votes over time. As the amount of data increases, these two metrics begin to separate, and α is the threshold on their ratio.

Finally, for every hypothesis we refine its 3D position by optimising the reprojection error from (8.1) over all the hypothesis detections. This entire algorithm is presented in Algorithm 1.

8.3.3 Large-Scale System

The above method works well for small-scale scenario, but does not scale well to large, city-scale settings we are interested in due to its potential $\mathcal{O}(N^4)$ complexity where N is the number of detected traffic lights [2]. Instead, we resort to a distribution schema based on splitting the data set to clusters, running Algorithm1 for each cluster independently, and then merging the results.

We employ a simple clustering schema where we keep identifying the closest images to a detected traffic light until a cluster of size N_{max} is created, at which point we remove it from the data set and continue the process until it terminates. An illustration of these clusters is shown in Figure 8.5.

After each cluster is triangulated using Algorithm 1 it might be the case that the same traffic light is triangulated in two different clusters. To resolve this issue we merge all pairs of traffic lights closer than $1m$, producing the final set of labels L.

[2] A slightly better complexity of $\mathcal{O}(N^3)$ can be achieved by reusing the computation of the inliers in each iteration.

	San Francisco	*New York City*
# images	12048	360207
# detections	17198	547689
# clusters	172	3941
# traffic lights	183	1732
# detectable traffic lights	167	1906
mean # views / cluster	70.05	91.4
mean # detections / frame	1.65	2.84
# images with 0 detection	1587	44483
# images with 1 detections	6231	145608
# images with 2 detections	2389	125924
# images with 3 detections	1346	32487
# images with 4 detections	369	7754
# images with 5 detections	98	2700
# images with 6 detections	17	802
# images with 7 detections	9	283
# images with 8 detections	0	80
# images with 9 detections	1	51
# images with 10+ detections	1	35

TABLE 8.2: Per-dataset statistics of 2D detection and clustering.

	San Francisco	*New York City*
true positives	156	1560
false positives	4	84
false negatives	11	172
duplicates	14	56
mean reprojection error	2.94	3.24

TABLE 8.3: The results of the method on two datasets.

8.4 Experiments

We evaluate the presented system on two large-scale data sets from San Francisco and New York City that we collected using a dedicated fleet of mapping vehicles. We demonstrate that the presented system scales to the size of cities, and that as the amount of data increases it generates increasingly accurate results both in terms of successfully recovered traffic lights and their 3D positions, despite using a very inaccurate off-the-shelf 2D detector.

8.4.1 Data Sets

The San Francisco and New York City data sets have been created by capturing images using a fleet of vehicles. The vehicles traversed most of the roads

multiple times, in both directions, at varying times of day and weather conditions. During this time they captured images at regular intervals. Example images are shown in Figs. 8.1 and 8.5. Each of these images has associated ground-truth 2D labels of traffic lights with label noise estimated at 5%.

We resize each image to 640×480 pixels and use a large-scale, distributed, structure-from-motion pipeline [82, 175] running on multiple computers to calculate the 3D positions P of the images.

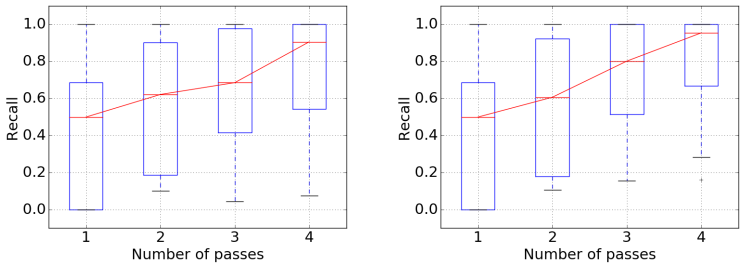
Each data set covers an area with a certain number of physical traffic lights. Not all of them are recoverable, i.e. their 3D positions cannot be accurately determined. We consider a traffic light recoverable if it has been observed from at least two different viewpoints under angle difference at least θ_{min}. In reality, as the amount of data increases, almost all the traffic lights eventually become recoverable. The sizes of these data sets together with their RMSE results are shown in Tab. 8.2. We also present the amount of traffic lights present in our dataset along with the number of detectable traffic lights.

8.4.2 2D Detection

We use a simple, convolutional neural network architecture to detect traffic lights in 2D images. Firstly, we use a binary segmentation network [72] to compute the probability that each pixel is part of a traffic light. We then use a simple thresholding schema to compute connected components of pixels representing traffic lights, and fit bounding boxes.

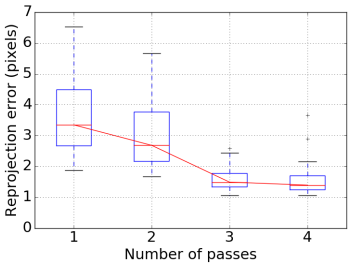
We train this network using the Bosch Data Set [11]. We split the data set into a training set of size 5,093 and a testing set of size 8,334, covering in total 24,242 traffic lights detections. For training we use two Nvidia P5000 cards until convergence. The network has been used to classify all the images in our datasets.

In this work we purposely did not fine-tune the detector for either the San Francisco or New York City data sets. There are significant differences between the training and testing data: the Bosch data are from a suburban area, while our data are urban; the cameras are different; and the training data contain mostly small traffic lights while our evaluation data contain traffic lights of all sizes. While the learned classifier achieves 90% recall on the Bosch test data, it becomes a relatively noisy detector on our datasets with a recall of 85% and an average Intersection-Over-Union (IOU) of 0.45.

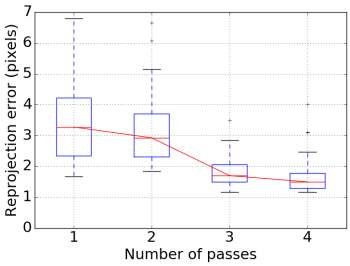


(A) New York City

(B) San Francisco



(C) New York City



(D) San Francisco

FIGURE 8.6: Performance of the system as a function of number of passes through a location. As the amount of data increases both recall and 3D localisation accuracy (as measured by negative reprojection error) increase.

8.4.3 Results

We run the described clustering method from Section 8.3 on all the data. Figure 8.5 shows the results of clustering in the San Francisco data set. Any clusters can contains images from multiple passes of the mapping vehicles through the area, as shown in Table 8.4.

Table 8.3 shows the statistics for the triangulation step. The presented method is able to recover at least 90% of all the recoverable traffic lights in both data sets, while suffering from only about 10% of false positive detections. The average reprojection error of the triangulated traffic lights for two datasets is 2.94 and 3.24 pixels for San Francisco and New York respectively. Note that reprojection error incorporates both the error in the triangulated 3D object and the underlying map accuracy. As discussed in Section 8.3.3, during the triangulation some of the traffic lights might be detected in two or more different clusters and must be unified in the merge step. These form only a small fraction of all the traffic lights.

Running the 2D detector over all 372k images took 30 hours. Clustering was performed in 1.1 minutes for San Francisco and 19 minutes for New York City, while the final triangulation took 1.1 and 3.1 minutes respectively.

Of primary importance is the performance of the system as the number of data increases. We characterise this by an increasing number of passes

# passes	# clusters San Francisco	New York City
1	47	476
2	43	304
3	9	229
4	6	121
5	1	49
6	4	27
7	6	0
8	6	0
9	1	0

TABLE 8.4: Statistics of the number of passes per cluster. As the mapping fleet traverses the environment each place is visited several times. As discussed in the text more passes through the environment result into higher performance of the system.

(A) (B) (C)

FIGURE 8.7: Failure modes of the presented method. When not enough data from a particular location are provided both (a) false negative detections due to the undetected objects or (b) temporarily consistent wrong detections can manifest. The largest challenge is (c) consistent and repeated detections of a objects that look similar to a traffic light over a period of time.

through an area. For this experiment we took a random subset of 25 clusters with at least 5 passes and computed the statistics of the number of recovered traffic lights and their reprojection error. The results are shown in Figure 8.6(a-b). Note that the recall starts off poorly because not all traffic lights in a cluster are detectable in a single pass, if for instance they are angled orthogonally to the direction of travel and increases with the number of passes through an area. With increasing passes we are more likely to detect the traffic light and have enough views to accurately localise it in 3D. We can see that the likelihood of detecting a traffic light does increase with number of passes. Note that the number of false positives for this random subset of 25 clusters is zero.

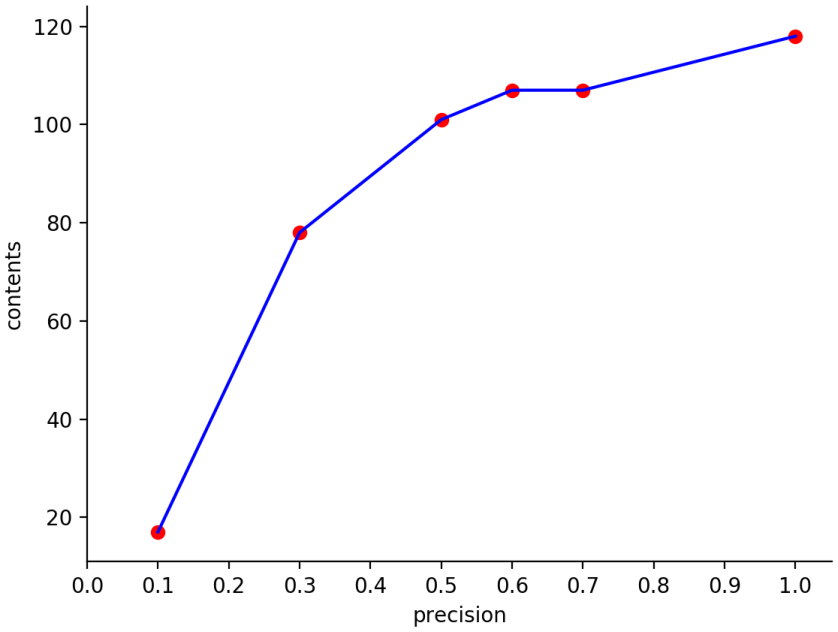


FIGURE 8.8: Number of detected contents in respect of a precision parameter. The precision value is multiplied by the distance threshold when voting for a content.

Figures 8.6(c-d) show that the 3D localisation accuracy also increases with number of passes. We measure this by taking the 3D object position estimated using up to N passes and project it into the images from a *leave-one-out* pass, measuring the reprojection error between where that 3D object is projected into the image and where it truly appears in the image. The statistics in Figure 8.6 show that the system converges in both recoverability of the traffic lights and their 3D position with more data. Figure 8.8 shows an intuitive result, which consists in the fact that there is a trade off between number of detected contents and reprojection error. This can be tuned using a value called reprojection factor α. Given a content C_i in 3D, its bounding box with center B_i in an image, a corner of the bounding box K_i and its 2D reprojection R_i in an image, the system cast a positive vote if

$$dist(R_i, B_i) \leq \alpha * dist(B_i, K_i). \tag{8.2}$$

In other words this means that the reprojected points falls inside the circle which has center B_i and radius (B_i, K_i). Lowering α lowers the size of the radius requiring a more precise reprojection in order to cast a positive vote.

Finally, we show some failure cases. The method converges under the assumption that the underlying detector is unbiased, but might produce

incorrect results when this assumption is broken or when only a small amount of data are provided. Figure 8.7 shows some of these cases. While the method is prone to generate a number of false negative detections when not enough data are provided, the largest problem are the false positive detections created by consistent, incorrect detections. While some of them tend to appear only in a single pass (such as series of reflections on a deck of the car) and thus can be resolved with more data, the incorrect detections of traffic-light-like objects which are repeatedly and consistently observed over periods of time posses a serious challenge to be addressed in the future.

8.5 Conclusion

We have demonstrated a simple and robust system for finding the 3D positions of static objects in complex city environments. We leverage a reliable image pose source and a large quantity of image data to overcome the common challenges of noisy 2D labels. The resulting accurate 3D object positions are borne out of a voting-based triangulation system that solves the data association problem that poses a particularly difficult challenge when the desired objects are similar in physical construction and yet appear vastly different in images as a result of strong variations in lighting and weather.

The system is specifically designed in such a way to be parallelisable, and therefore efficiently process very large image sets. We have evaluated our system on city-scale data sets comprising almost 0.4M images, and have shown that despite the very noisy input detections, the system output increases in 3D positional accuracy and recall with more data. This has been used to answers **RQ1**.

To answer **RQ2** we had a look at the state of the art solutions and decided to use CNNs given that all the most promising results in this research area have been developed using this technique. We showed in Section 8.4.3 that we are able to localize almost all traffic lights even if the used detections are not optimal given that we trained our detection system on a dataset which is very different from the evaluation one. This proves also that a robust voting scheme can cope with a bad detections which is a fundamental assumption for an AS.

To estimate the precision of the system we tested several different triangulation techniques in Section 8.2. We used a fixed test case to evaluate precision and convergence speed of several algorithms showing that we can evaluated with enough precision the position of an object in space given multiple views. Obviously the required precision depends on the task which

the AS has to perform; for example, in the case of autonomous cars, knowing the position of a traffic light within a few centimeters might be enough. All this has been used to answers **RQ3** and **RQ6**.

Chapter 9

Conclusions

In this thesis we analysed how Machine Learning techniques can be used in combination with Autonomous Systems to interact with humans and the environment. In particular, we analysed how Autonomous systems can detect and classify activities, environments and humans states. A general overview of the different techniques and application domains in depicted in Figure 1.1. In the following we will resume the obtained results and contributions which have been obtained in this work as reported in Table 1.1.

The first contribution of this work **C1** consists in the development of an autonomous system capable of recognizing activities. In particular we developed the PELARS system to identify learning activities performed during projectual sessions by students on a sensorised table. Such system is composed of a set of sensors that acquire information about the students, an annotation system based on mobile phones and tables to record images, videos and text, and a collection and visualization system that stores and processes the data in order to provide learning statistics to students, teachers and researchers. The sensors were used to acquire the position of faces, hands and objects on the PELARS table in order to measure interaction between students. All the data is processed locally on a desktop PC, which is used also as interface for the students to the Arduino programming toolkit. The acquisition code is written in C++ using concurrent threads for the acquisition and Cuda code for the processing on GPU of input signals. All data is processed and sent to a remote server that exposes a set of Websocket endpoints and some REST APIs for the data storage, manipulation and visualization. The server has been written in Java and uses a MySql database to store data.

The system has been evaluated with two different approaches: in the first experiment we evaluated the possibility of inferring the working phase from the acquired data using classical ML techniques, while in the second experiment we evaluated the possibility of inferring projectual outcomes

using DL techniques. Working phases and projectual outcomes have been annotated by teachers and researchers during testing sessions. 18 engineering students at an European university were involved in the experiment as detailed in Section 6.5.1.

The results showed in both cases the possibility of inferring working phases and projectual outcomes from the data acquired at the sensorised table. This allows teachers to possibly focus more on student in difficulty, propose automatic help from the system for students in need and it provides an easy annotation and visualization tool for both students and teachers for post analysis of the work.

The second contribution of this work **C2** consists in the development of an autonomous system capable of evaluating the state of a person. We developed this system in the context of the RAMCIP project, which aims at creating an autonomous robot to help elderly people with MCI. The goal of this work is to estimate fatigue in patients by looking at gait patterns using non invasive techniques. The developed system is composed of a Kinect camera, which acquired RGB-D frames at 30Hz and a Desktop pc for the acquisition and processing of the data. In particular we acquired the human skeleton using a skeleton tracker based on Random Forests. The experiment dataset has been acquired by firstly letting patients walk in front of the camera in a non fatigued state, then applying a fatiguing protocol and finally having the patients walk again in front of the camera in a fatigued state; in total 20 people have been recorded.

Given the extreme variability of the input data and the lack of strong fatigue indicators from gait analysis, we opted for a DL method using as input the position of the lower limb joints. Data had to be pre-filtered partially to ease the training of the system given that often it was not possible to track for enough time with high confidence the skeleton of the patient.

Results show that we were able to infer correctly the fatigue state of patients with an accuracy of 78.3% observing the patients for 2s. This makes it possible for autonomous robots to adapt their behavior to the state of the person interacting with them. Also the latency of the system is crucial for such applications given that people move around and it might be hard to track them for a long duration with high confidence. Given the high variability of subjects it might be strongly possible that additional data could produce a better detection network increasing the classification accuracy. Also a better sensor could contribute to increase the overall performance of the system, like for example the Kinect v2.

The last two contributions of this thesis **C3, C4** consist in the development an autonomous system capable of detecting and triangulating contents in large scale 3D maps. A content can be anything that can be detected in 2D images by a detector. In our case we tested the system by looking for traffic lights since they are widely used in the autonomous driving sector. The system is detecting contents, but most importantly triangulating them by associating correctly the same content in multiple images using a voting scheme. A 3D map consists in a set of images along with relative poses.

To test the system we developed a 3D map of San Francisco and New York reaching up to a size of 400k frames. We used a fully convolutional neural network to detect the traffic lights in the images and then, after associating them correctly in the different views, triangulated them. We evaluated the system and were able to correctly detect and triangulate over 90% of the available traffic lights. Most importantly, we showed that the system tends to get better performances over time when new data arrives (each street can have multiple passes during different times of the day and different weather conditions) and this is a fundamental assumption given that the system can have many false positives in the detection phase.

To triangulate the detected traffic lights we tested several approaches, in order to find the most precise and fastest algorithm. This was done given that we could not find a large comparison of state of the art triangulation systems, leading to the constribution **C4**. We decided to use 3DTMA given the results presented in Section 8.2, but several other strong candidate algorithms are available.

We did not spend time on the optimization of our detection system that was trained on a dataset which is very different than the one used to build the maps. This has been done since we wanted to test the robustness of the system to noise. We proved the robustness of the system and will test also how much performance is gained from improving the detector in a future work. It is also important to notice that in principle we could use directly the final output of the system to do hard negative mining to improve automatically the quality of the detector.

All these results are fundamental in the field of autonomous driving given that the environment changes rapidly and that weather conditions might influence strongly a detection system. Our system shows a way of developing an automatic system for generic detection which can be boot strapped with a small initial dataset and then improves over time when more data is available.

All these contributions represent an attempt to propose possible new strategies to use ML techniques to enhance the capabilities of autonomous systems. These new techniques are almost always performing better than classical approaches and can push the state of the art in robotic applications. Given that autonomous systems are getting pervasive in daily life, it is becoming of fundamental importance to enhance the interaction quality between them and humans, making it as natural as possible. This will probably become the next big challenge.

Appendix

Appendix A

List of Publications

Conference

- Giacomo Dabisias, Emanuele Ruffaldi, Hugo Grimmet, and Ondruska Peter. "VALUE: Large Scale Voting-based Automatic Labelling for Urban Environments". In: (2018) **Accepted at ICRA2018**

- Emanuele Ruffaldi, Giacomo Dabisias, Lorenzo Landolfi, and Daniel Spikol. "Data collection and processing for a multimodal learning analytic system". In: *SAI Computing Conference (SAI), 2016*. IEEE. 2016, pp. 858–863

- Emanuele Ruffaldi, Filippo Brizzi, Giacomo Dabisias, and Giorgio Buttazzo. "SOMA: an OpenMP toolchain for multicore partitioning". In: *Proceedings of the 31st Annual ACM Symposium on Applied Computing*. ACM. 2016, pp. 1231–1237

- Giacomo Dabisias, Daniel Spikol, and Emanuele Ruffaldi. "A Learning Analytics Framework for Practice-Based Learning". In: (2015)

- Paolo Tripicchio, Massimo Satler, Giacomo Dabisias, Emanuele Ruffaldi, and Carlo Alberto Avizzano. "Towards smart farming and sustainable agriculture with drones". In: *Intelligent Environments (IE), 2015 International Conference on*. IEEE. 2015, pp. 140–143

Journal

- Giacomo Dabisias, Lorenzo Peppoloni, Alessandro Graziano, Justyna Gerłowska, Konrad Rejdak, and Emanuele Ruffaldi. "Deep learning based automated fatigue detection in MCI subjects". In: *Transaction on Human-Machine Systems, 2017*. IEEE. 2017, xxx–yyy **Ready to be submitted**

- Daniel Spikol, Emanuele Ruffaldi, Giacomo Dabisias, and Mutlu Cukurova. "Supervised Machine Learning in Multimodal Learning Analytics for Estimating Success inProject-based Learning". In: *JCAL, 2017*. IEEE. 2018, xxx–yyy **Submitted**

Bibliography

[1] Martín Abadi et al. "Tensorflow: Large-scale machine learning on heterogeneous distributed systems". In: *arXiv preprint arXiv:1603.04467* (2016).

[2] W Abrah&atildeo et al. "A comparison of Haar-like, LBP and HOG approaches to concrete and asphalt runway detection in high resolution imagery". In: ().

[3] Brandon Amos, Bartosz Ludwiczuk, and Mahadev Satyanarayanan. *OpenFace: A general-purpose face recognition library with mobile applications*. Tech. rep. CMU-CS-16-118, CMU School of Computer Science, 2016.

[4] Cynthia J. Atman et al. "Engineering Design Processes: A Comparison of Students and Expert Practitioners". In: *J. Eng. Educ.* 96.4 (2007), pp. 359–379. ISSN: 10694730. DOI: 10.1002/j.2168-9830.2007.tb00945.x. URL: http://doi.wiley.com/10.1002/j.2168-9830.2007.tb00945.x (visited on 10/15/2016).

[5] Lindsay Bahureksa et al. "The impact of mild cognitive impairment on gait and balance: a systematic review and meta-analysis of studies using instrumented assessment". In: *Gerontology* 63.1 (2017), pp. 67–83.

[6] Dana H Ballard. "Generalizing the Hough transform to detect arbitrary shapes". In: *Pattern recognition* 13.2 (1981), pp. 111–122.

[7] Vassileios Balntas, Lilian Tang, and Krystian Mikolajczyk. "Bold-binary online learned descriptor for efficient image matching". In: *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*. IEEE. 2015, pp. 2367–2375.

[8] Dan Barnes, Will Maddern, and Ingmar Posner. "Exploiting 3D semantic scene priors for online traffic light interpretation". In: *Intelligent Vehicles Symposium (IV), 2015 IEEE*. IEEE. 2015, pp. 573–578.

[9] Luis C Basaca-Preciado et al. "Optical 3D laser measurement system for navigation of autonomous mobile robot". In: *Optics and Lasers in Engineering* 54 (2014), pp. 159–169.

[10] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. "Surf: Speeded up robust features". In: *European conference on computer vision*. Springer. 2006, pp. 404–417.

[11] Karsten Behrendt and Libor Novak. "A Deep Learning Approach to Traffic Lights: Detection, Tracking, and Classification". In: *ICRA*. IEEE. 2017, pp. 1370–1377.

[12] Stephanie Bell. "Project-Based Learning for the 21st Century: Skills for the Future". In: *The Clearing House: A Journal of Educational Strategies, Issues and Ideas* (2010).

[13] Yoshua Bengio, Nicolas Boulanger-Lewandowski, and Razvan Pascanu. "Advances in optimizing recurrent networks". In: *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE. 2013, pp. 8624–8628.

[14] James Bergstra et al. "Theano: A CPU and GPU math compiler in Python". In: *9th Python in Science Conf*. 2010, pp. 1–7.

[15] Reinhard Blickhan. "The spring-mass model for running and hopping". In: *Journal of biomechanics* 22.11-12 (1989), pp. 1217–1227.

[16] Paulo Blikstein. "Multimodal learning analytics". In: *Proceedings of the Third International Conference on Learning Analytics and Knowledge - LAK '13*. Ed. by Dan Suthers and Katrien Verbert. New York, New York, USA: ACM Press, 2013, p. 102. DOI: 10.1145/2460296.2460316. URL: http://dl.acm.org/citation.cfm?doid=2460296.2460316 (visited on 10/13/2016).

[17] Paulo Blikstein. "Using learning analytics to assess students' behavior in open-ended programming tasks". In: *Proceedings of the 1st International Conference on Learning Analytics and Knowledge - LAK '11*. New York, New York, USA: ACM Press, 2011, p. 110. DOI: 10.1145/2090116.2090132. URL: http://dl.acm.org/citation.cfm?doid=2090116.2090132 (visited on 10/14/2016).

[18] Paulo Blikstein and Marcelo Worsley. "Multimodal Learning Analytics and Education Data Mining: using computational technologies to measure complex learning tasks". In: *Journal of Learning Analytics* (2016). URL: http://epress.lib.uts.edu.au/journals/index.php/JLA/article/view/4383/5596 (visited on 10/12/2016).

[19] Phyllis C. Blumenfeld et al. "Motivating Project-Based Learning: Sustaining the Doing, Supporting the Learning". In: *Educ Psychol* 26.3-4 (1991), pp. 369–398. ISSN: 0046-1520. DOI: 10.1080/00461520.1991.9653139. URL: http://www.tandfonline.com/doi/abs/10.1080/00461520.1991.9653139 (visited on 10/13/2016).

[20] Michael Buhrmester, Tracy Kwang, and Samuel D Gosling. "Amazon's Mechanical Turk: A new source of inexpensive, yet high-quality, data?" In: *Perspectives on psychological science* 6.1 (2011), pp. 3–5.

[21] Stephen Butterworth. "On the theory of filter amplifiers". In: *Wireless Engineer* 7.6 (1930), pp. 536–541.

[22] Michael Calonder et al. "Brief: Binary robust independent elementary features". In: *European conference on computer vision*. Springer. 2010, pp. 778–792.

[23] Zhe Cao et al. "Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields". In: *CVPR*. 2017.

[24] Hui Chen and Bir Bhanu. "3D free-form object recognition in range images using local surface patches". In: *Pattern Recognition Letters* 28.10 (2007), pp. 1252–1262.

[25] Lei Chen et al. "Towards Automated Assessment of Public Speaking Skills Using Multimodal Cues". In: *Proceedings of the 16th International Conference on Multimodal Interaction - ICMI '14*. New York, New York, USA: ACM Press, 2014, pp. 200–203. ISBN: 9781450328852. DOI: 10.1145/2663204.2663265. URL: http://dl.acm.org/citation.cfm?doid=2663204.2663265 (visited on 10/15/2016).

[26] Liang-Chieh Chen, George Papandreou, et al. "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs". In: *arXiv preprint arXiv:1606.00915* (2016).

[27] Trishul M Chilimbi et al. "Project Adam: Building an Efficient and Scalable Deep Learning Training System." In: *OSDI*. Vol. 14. 2014, pp. 571–582.

[28] Minsik Cho et al. "PowerAI DDL". In: *arXiv preprint arXiv:1708.02188* (2017).

[29] Changhyun Choi and Henrik I Christensen. "3D pose estimation of daily objects using an RGB-D camera". In: *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE. 2012, pp. 3342–3349.

[30] François Chollet. *Keras*. 2015.

[31] John Cohen. *Human robots in myth and science*. AS Barnes, 1967.

[32] T. Cowen. *Modern Principles of Economics*. Worth Publishers, 2009.

[33] Leandro Cruz, Djalma Lucio, and Luiz Velho. "Kinect and rgbd images: Challenges and applications". In: *SIBGRAPI-T*. IEEE. 2012, pp. 36–49.

[34] Gabriela Csurka and Florent Perronnin. "A Simple High Performance Approach to Semantic Segmentation." In: *BMVC*. 2008, pp. 1–10.

[35] Giacomo Dabisias et al. "Deep learning based automated fatigue detection in MCI subjects". In: *Transaction on Human-Machine Systems, 2017*. IEEE. 2017, xxx–yyy.

[36] Giacomo Dabisias et al. "VALUE: Large Scale Voting-based Automatic Labelling for Urban Environments". In: (2018).

[37] Giacomo Dabisias, Daniel Spikol, and Emanuele Ruffaldi. "A Learning Analytics Framework for Practice-Based Learning". In: (2015).

[38] Oscar Déniz et al. "Face recognition using histograms of oriented gradients". In: *Pattern Recognition Letters* 32.12 (2011), pp. 1598–1603.

[39] Konstantinos G Derpanis. "The harris corner detector". In: *York University* (2004).

[40] A. Doyle. *The Adventures of Sherlock Holmes. Adventure 4: "The Boscombe Valley Mystery"*. Lit2Go Edition, 1892.

[41] K. Doğançay. "3D Pseudolinear Target Motion Analysis From Angle Measurements". In: *IEEE Transactions on Signal Processing* 63.6 (2015), pp. 1570–1580. ISSN: 1053-587X. DOI: 10.1109/TSP.2015.2399869. URL: http://ieeexplore.ieee.org/document/7029692/.

[42] Dimiter Driankov and Alessandro Saffiotti. *Fuzzy logic techniques for autonomous vehicle navigation*. Vol. 61. Physica, 2013.

[43] Bertram Drost et al. “Model globally, match locally: Efficient and robust 3D object recognition”. In: *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE. 2010, pp. 998–1005.

[44] Vincent Dumoulin and Francesco Visin. “A guide to convolution arithmetic for deep learning”. In: *arXiv preprint arXiv:1603.07285* (2016).

[45] PP Dutta, A Baruah, A Konwar, et al. “A Technical Review of Lawn Mower Technology”. In: *ADBU Journal of Engineering Technology* 4 (2016).

[46] Andreas Eitel et al. “Multimodal deep learning for robust rgb-d object recognition”. In: *IROS*. IEEE. 2015, pp. 681–687.

[47] Paul L Enright. “The six-minute walk test”. In: *Respiratory care* 48.8 (2003), pp. 783–785.

[48] Nathaniel Fairfield and Chris Urmson. “Traffic light mapping and detection”. In: *ICRA*. IEEE. 2011, pp. 5421–5426.

[49] Michael Firman. “RGBD Datasets: Past, Present and Future”. In: *CVPR Workshop on Large Scale 3D Data: Acquisition, Modelling and Analysis*. 2016.

[50] Robert B Fisher. “Projective ICP and stabilizing architectural augmented reality overlays”. In: *Virtual and Augmented Architecture (VAA'01)*. Springer, 2001, pp. 69–80.

[51] Marshal F Folstein et al. “"Mini-mental state": a practical method for grading the cognitive state of patients for the clinician”. In: *Journal of psychiatric research* 12.3 (1975), pp. 189–198.

[52] Regina Frei et al. “Self-healing and self-repairing technologies”. In: *The International Journal of Advanced Manufacturing Technology* 69.5-8 (2013), pp. 1033–1061.

[53] Benoît Frénay and Michel Verleysen. “Classification in the presence of label noise: a survey”. In: *IEEE transactions on neural networks and learning systems* 25.5 (2014), pp. 845–869.

[54] Moshe Gabel, Ran Gilad-Bachrach, et al. “Full body gait analysis with Kinect”. In: *IEEE EMBC*. 2012, pp. 1964–1967.

[55] Quan Gan et al. “First step toward model-free, anonymous object tracking with recurrent neural networks”. In: *arXiv preprint arXiv:1511.06425* (2015).

[56] Erann Gat et al. “On three-layer architectures”. In: *Artificial intelligence and mobile robots* 195 (1998), p. 210.

[57] Ross Girshick. “Fast r-cnn”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1440–1448.

[58] Ross Girshick et al. “Rich feature hierarchies for accurate object detection and semantic segmentation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 580–587.

[59] Priya Goyal et al. “Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour”. In: *arXiv preprint arXiv:1706.02677* (2017).

[60] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. "Speech recognition with deep recurrent neural networks". In: *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*. IEEE. 2013, pp. 6645–6649.

[61] Daniel Grest, Jan Woetzel, and Reinhard Koch. "Nonlinear body pose estimation from depth images". In: *DAGM-Symposium*. Vol. 5. Springer. 2005, pp. 285–292.

[62] Shuchi Grover et al. "Multimodal analytics to study collaborative problem solving in pair programming". In: *Proceedings of the Sixth International Conference on Learning Analytics & Knowledge - LAK '16*. New York, New York, USA: ACM Press, 2016, pp. 516–517. ISBN: 9781450341905. DOI: 10.1145/2883851.2883877. URL: http://dl.acm.org/citation.cfm?doid=2883851.2883877 (visited on 10/17/2016).

[63] Edwin H Hall. "On a new action of the magnet on electric currents". In: *American Journal of Mathematics* 2.3 (1879), pp. 287–292.

[64] Richard Hartley and Andrew Zisserman. *"Multiple view geometry in computer vision"*. Cambridge university press, 2003. ISBN: 0521540518.

[65] Richard I Hartley and Peter Sturm. "Triangulation". In: *Computer vision and image understanding* 68.2 (1997), pp. 146–157.

[66] Kaiming He et al. "Mask r-cnn". In: *arXiv preprint arXiv:1703.06870* (2017).

[67] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[68] Steve J Heims and Duane W Bailey. "John von Neumann and Norbert Wiener, from Mathematics to the technologies of life and death". In: *American Journal of Physics* 50.4 (1982), pp. 383–383.

[69] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. "A fast learning algorithm for deep belief nets". In: *Neural computation* 18.7 (2006), pp. 1527–1554.

[70] Sepp Hochreiter et al. *Gradient flow in recurrent nets: the difficulty of learning long-term dependencies*. 2001.

[71] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[72] Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. "Densely Connected Convolutional Networks". In: *CoRR* abs/1608.06993 (2016).

[73] Gao Huang et al. "Densely connected convolutional networks". In: *arXiv preprint arXiv:1608.06993* (2016).

[74] Jingwei Huang and David Altamar. "Pose Estimation on Depth Images with Convolutional Neural Network". In: ().

[75] Du Q Huynh. "Metrics for 3D rotations: Comparison and analysis". In: *Journal of Mathematical Imaging and Vision* 35.2 (2009), pp. 155–164.

[76] *Inception*. 2010.

[77] Eldar Insafutdinov et al. “Deepercut: A deeper, stronger, and faster multi-person pose estimation model”. In: *European Conference on Computer Vision*. Springer. 2016, pp. 34–50.

[78] Mircea Horea Ionica and David Gregg. “The movidius myriad architecture’s potential for scientific computing”. In: *IEEE Micro* 35.1 (2015), pp. 6–14.

[79] Joel Janai et al. “Computer Vision for Autonomous Vehicles: Problems, Datasets and State-of-the-Art”. In: *arXiv preprint 1704.05519* (2017).

[80] Kevin Jarrett, Koray Kavukcuoglu, Yann LeCun, et al. “What is the best multi-stage architecture for object recognition?” In: *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE. 2009, pp. 2146–2153.

[81] Vijay John et al. “Traffic light recognition in varying illumination using deep learning and saliency map”. In: *Intelligent Transportation Systems (ITSC)*. IEEE. 2014, pp. 2286–2291.

[82] Bryan Klingner, David Martin, and James Roseborough. “Street view motion-from-structure-from-motion”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2013, pp. 953–960.

[83] Jan Knopp et al. “Hough transform and 3D SURF for robust three dimensional classification”. In: *European Conference on Computer Vision*. Springer. 2010, pp. 589–602.

[84] Jan J Koenderink and Andrea J Van Doorn. “Affine structure from motion”. In: *JOSA A* 8.2 (1991), pp. 377–385.

[85] S Kotsiantis, Kiriakos Patriarcheas, and M Xenos. “A combinational incremental ensemble of classifiers as a technique for predicting students’ performance in distance education”. In: *Knowledge-Based Systems* 23.6 (2010), pp. 529–535.

[86] Joe Krajcik. “Project-Based Science: Engaging Students in Three-Dimensional Learning”. In: *The Science Teacher* 82.1 (2010), pp. 1–25.

[87] Joe Krajcik and patrick Blumenfeld. “Project-based learning”. In: *The Cambridge handbook of the learning sciences*. Ed. by Richard Sawyer. New York, Cambridge, USA: Cambridge University Press, 2006, pp. 317–334.

[88] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[89] Griffin Lacey, Graham W Taylor, and Shawki Areibi. “Deep learning on fpgas: Past, present, and future”. In: *arXiv preprint arXiv:1602.04283* (2016).

[90] Yann LeCun, Fu Jie Huang, and Leon Bottou. “Learning methods for generic object recognition with invariance to pose and lighting”. In: *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*. Vol. 2. IEEE. 2004, pp. II–104.

[91] Hon-Leung Lee. "Critical Points for Two-view Triangulation". In: *arXiv preprint 1608.05512* (2016).

[92] Sidney R Lehky and TJ Sejnowski. "Network model of shape-from-shading: neural function arises from both receptive and projective fields". In: *Nature* 333.6172 (1988), pp. 452–454.

[93] John J Leonard and Alexander Bahr. "Autonomous underwater vehicle navigation". In: *Springer Handbook of Ocean Engineering*. Springer, 2016, pp. 341–358.

[94] *linkedin, howpublished* = http://www.linkedin.com/pulse/every-company-use-blockchain-2027-heres-how-mohit-mamoria/.

[95] Wai-Yan Liu et al. "Reproducibility and Validity of the 6-Minute Walk Test Using the Gait Real-Time Analysis Interactive Lab in Patients with COPD and Healthy Elderly". In: *PloS one* 11.9 (2016), e0162444.

[96] Wei-Yin Loh. "Classification and regression trees". In: *Wiley Int. Reviews: Data Mining and Knowledge Discovery* 1.1 (2011), pp. 14–23.

[97] David G Lowe. "Object recognition from local scale-invariant features". In: *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*. Vol. 2. Ieee. 1999, pp. 1150–1157.

[98] Esther Martín-Ponce et al. "Prognostic value of physical function tests: hand grip strength and six-minute walking test in elderly hospitalized patients". In: *Nature Scientific reports* 4 (2014).

[99] Roberto Martinez-Maldonado et al. "Interactive surfaces and learning analytics: Data, orchestration aspects, pedagogical uses and challenges". In: *Proceedings of the Sixth International Conference on Learning Analytics & Knowledge - LAK '16*. New York, New York, USA: ACM Press, 2016, pp. 124–133. ISBN: 9781450341905. DOI: 10.1145/2883851.2883873. URL: http://dl.acm.org/citation.cfm?doid=2883851.2883873 (visited on 10/15/2016).

[100] Daniel Maturana and Sebastian Scherer. "Voxnet: A 3d convolutional neural network for real-time object recognition". In: *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE. 2015, pp. 922–928.

[101] Volodymyr Mnih et al. "Playing atari with deep reinforcement learning". In: *arXiv preprint arXiv:1312.5602* (2013).

[102] Thomas B Moeslund and Erik Granum. "A survey of computer vision-based human motion capture". In: *Computer vision and image understanding* 81.3 (2001), pp. 231–268.

[103] Marius Muja and David G Lowe. "Scalable nearest neighbor algorithms for high dimensional data". In: *IEEE transactions on pattern analysis and machine intelligence* 36.11 (2014), pp. 2227–2240.

[104] R Munoz-Salinas and S Garrido-Jurado. "Aruco library". In: *URL: http://sourceforge. net/projects/aruco* (2013).

[105] Erik Murphy-Chutorian and Mohan Manubhai Trivedi. "Head pose estimation in computer vision: A survey". In: *IEEE transactions on pattern analysis and machine intelligence* 31.4 (2009), pp. 607–626.

[106] Vinod Nair and Geoffrey E Hinton. "3D object recognition with deep belief nets". In: *Advances in neural information processing systems*. 2009, pp. 1339–1347.

[107] Maryam M Najafabadi et al. "Deep learning applications and challenges in big data analytics". In: *Journal of Big Data* 2.1 (2015), p. 1.

[108] Pablo Negri, Xavier Clady, and Lionel Prevost. "Benchmarking haar and histograms of oriented gradients features applied to vehicle detection." In: *ICINCO-RA (1)*. 2007, pp. 359–364.

[109] Richard A Newcombe et al. "KinectFusion: Real-time dense surface mapping and tracking". In: *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*. IEEE. 2011, pp. 127–136.

[110] *Nova Science Quotes, howpublished* = http://www.pbs.org/wgbh/nova/transcripts/3318_sciencen.html.

[111] Andreas Nüchter and Joachim Hertzberg. "Towards semantic maps for mobile robots". In: *Robotics and Autonomous Systems* 56.11 (2008), pp. 915–926.

[112] X Ochoa et al. "Mla'14: Third multimodal learning analytics workshop and grand challenges". In: *Proceedings of the 16th* (2014). URL: http://dl.acm.org/citation.cfm?id=2668318 (visited on 10/15/2016).

[113] Xavier Ochoa et al. "Expertise estimation based on simple multimodal features". In: *Proceedings of the 15th ACM on International conference on multimodal interaction - ICMI '13*. New York, New York, USA: ACM Press, 2013, pp. 583–590. ISBN: 9781450321297. DOI: 10.1145/2522848.2533789. URL: http://dl.acm.org/citation.cfm?doid=2522848.2533789 (visited on 10/17/2016).

[114] Xavier Ochoa et al. "Multimodal learning analytics data challenges". In: *Proceedings of the Sixth International Conference on Learning Analytics & Knowledge*. ACM. 2016, pp. 498–499.

[115] Charles Otto, Anil Jain, et al. "Clustering millions of faces by identity". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2017).

[116] Seongwook Park et al. "4.6 A1. 93TOPS/W scalable deep learning/inference processor with tetra-parallel MIMD architecture for big-data applications". In: *Solid-State Circuits Conference-(ISSCC), 2015 IEEE International*. IEEE. 2015, pp. 1–3.

[117] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. "On the difficulty of training recurrent neural networks". In: *International Conference on Machine Learning*. 2013, pp. 1310–1318.

[118] Deepak Pathak et al. "Fully convolutional multi-class multiple instance learning". In: *arXiv preprint arXiv:1412.7144* (2014).

[119] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[120] Markus Peters et al. "A reinforcement learning approach to autonomous decision-making in smart electricity markets". In: *Machine learning* 92.1 (2013), pp. 5–39.

[121] Leonid Pishchulin et al. "Deepcut: Joint subset partition and labeling for multi person pose estimation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 4929–4937.

[122] Luis P. Prieto et al. "Teaching analytics: Towards automatic extraction of orchestration graphs using wearable sensors". In: *Proceedings of the Sixth International Conference on Learning Analytics & Knowledge - LAK '16*. New York, New York, USA: ACM Press, 2016, pp. 148–157. ISBN: 9781450341905. DOI: 10.1145/2883851.2883927. URL: http://dl.acm.org/citation.cfm?doid=2883851.2883927 (visited on 10/13/2016).

[123] Charles R Qi et al. "Pointnet: Deep learning on point sets for 3d classification and segmentation". In: *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE* 1.2 (2017), p. 4.

[124] Charles Ruizhongtai Qi et al. "Pointnet++: Deep hierarchical feature learning on point sets in a metric space". In: *Advances in Neural Information Processing Systems*. 2017, pp. 5105–5114.

[125] Morgan Quigley, Ken Conley, et al. "ROS: an open-source Robot Operating System". In: *ICRA workshop*. Vol. 3. 3.2. 2009, p. 5.

[126] Hossein Rahmani, Ajmal Mian, and Mubarak Shah. "Learning a deep model for human action recognition from novel viewpoints". In: *TPAMII* (2017).

[127] Vikram Ramanarayanan et al. "Evaluating speech, face, emotion and body movement time-series features for automated multimodal presentation scoring". In: *Proceedings of the 2015 ACM on International Conference on Multimodal Interaction*. ACM. 2015, pp. 23–30.

[128] Joseph Redmon et al. "You only look once: Unified, real-time object detection". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 779–788.

[129] Shaoqing Ren et al. "Faster R-CNN: Towards real-time object detection with region proposal networks". In: *Advances in neural information processing systems*. 2015, pp. 91–99.

[130] Mario Rojas et al. "Automatic prediction of facial trait judgments: Appearance vs. structural models". In: *PloS one* 6.8 (2011), e23323.

[131] Ethan Rublee et al. "ORB: An efficient alternative to SIFT or SURF". In: *Computer Vision (ICCV), 2011 IEEE international conference on*. IEEE. 2011, pp. 2564–2571.

[132] Emanuele Ruffaldi et al. "SOMA: an OpenMP toolchain for multicore partitioning". In: *Proceedings of the 31st Annual ACM Symposium on Applied Computing*. ACM. 2016, pp. 1231–1237.

[133] Emanuele Ruffaldi et al. "Data collection and processing for a multimodal learning analytic system". In: *SAI Computing Conference (SAI), 2016*. IEEE. 2016, pp. 858–863.

[134] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. "Fast point feature histograms (FPFH) for 3D registration". In: *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE. 2009, pp. 3212–3217.

[135] Samuele Salti, Federico Tombari, and Luigi Di Stefano. "SHOT: Unique signatures of histograms for surface and texture description". In: *Computer Vision and Image Understanding* 125 (2014), pp. 251–264.

[136] Florian Schroff, Dmitry Kalenichenko, and James Philbin. "Facenet: A unified embedding for face recognition and clustering". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 815–823.

[137] Jonghyun Seo. *Automatic vacuum cleaner*. US Patent 9,376,150. 2016.

[138] Amir Shahroudy et al. "Deep multimodal feature analysis for action recognition in RGB+ D videos". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2017).

[139] Jamie Shotton et al. "Real-time human pose recognition in parts from single depth images". In: *Communications of the ACM* 56.1 (2013), pp. 116–124.

[140] George Siemens and Ryan S. J. d. Baker. "Learning analytics and educational data mining: Towards communication and collaboration". In: *Proceedings of the 2nd International Conference on Learning Analytics and Knowledge - LAK '12*. New York, New York, USA: ACM Press, 2012, p. 252. DOI: 10.1145/2330601.2330661. URL: http://dl.acm.org/citation.cfm?doid=2330601.2330661 (visited on 10/14/2016).

[141] Tomas Simon et al. "Hand Keypoint Detection in Single Images using Multiview Bootstrapping". In: *CVPR*. 2017.

[142] Eleanor M Simonsick et al. "Assessing Fatigability in Mobility-Intact Older Adults". In: *Journal of the American Geriatrics Society* 62.2 (2014), pp. 347–351.

[143] Stephen M Smith and J Michael Brady. "SUSAN—a new approach to low level image processing". In: *International journal of computer vision* 23.1 (1997), pp. 45–78.

[144] Daniel Spikol et al. "Supervised Machine Learning in Multimodal Learning Analytics for Estimating Success inProject-based Learning". In: *JCAL, 2017*. IEEE. 2018, xxx–yyy.

[145] Nitish Srivastava et al. "Dropout: a simple way to prevent neural networks from overfitting." In: *Journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[146] Johannes Stallkamp et al. "The German traffic sign recognition benchmark: a multi-class classification competition". In: *Neural Networks (IJCNN), The 2011 International Joint Conference on*. IEEE. 2011, pp. 1453–1460.

[147] Bastian Steder et al. "NARF: 3D range image features for object recognition". In: *Workshop on Defining and Solving Realistic Perception Problems in Personal Robotics at the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. Vol. 44. 2010.

[148] Dimitar H Stefanov, Zeungnam Bien, and Won-Chul Bang. "The smart house for older persons and persons with physical disabilities: structure, technology arrangements, and perspectives". In: *IEEE Trans. Neural Netw. Learn. Syst.* 12.2 (2004), pp. 228–250.

[149] *SupplyChain, howpublished = http://www.supplychaintoday.com/artificial-intelligence-deep-learning-quotes/*.

[150] Christian Szegedy et al. "Going deeper with convolutions". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1–9.

[151] Yaniv Taigman et al. "Deepface: Closing the gap to human-level performance in face verification". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 1701–1708.

[152] Adriana Tapus and Maja J Mataric. "Socially Assistive Robots: The Link between Personality, Empathy, Physiological Signals, and Task Performance." In: *AAAI Symposium: Emotion, Personality, and Social Behavior*. 2008, pp. 133–140.

[153] M. Team. *Makehuman software*. http://www.makehuman.org. 2001–2016.

[154] Federico Tombari, Samuele Salti, and Luigi Di Stefano. "Performance evaluation of 3D keypoint detectors". In: *International Journal of Computer Vision* 102.1-3 (2013), pp. 198–220.

[155] Jonathan J Tompson et al. "Joint training of a convolutional network and a graphical model for human pose estimation". In: *Advances in neural information processing systems*. 2014, pp. 1799–1807.

[156] Michael Toscano. "Department of defense joint robotics program". In: *Proc. SPIE*. Vol. 4715. 1999, p. 97.

[157] Alexander Toshev and Christian Szegedy. "Deeppose: Human pose estimation via deep neural networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 1653–1660.

[158] Paolo Tripicchio et al. "Towards smart farming and sustainable agriculture with drones". In: *Intelligent Environments (IE), 2015 International Conference on*. IEEE. 2015, pp. 140–143.

[159] Edward Tunstel et al. "FIDO rover field trials as rehearsal for the NASA 2003 Mars Exploration Rovers Mission". In: *Automation Congress, 2002 Proceedings of the 5th Biannual World*. Vol. 14. IEEE. 2002, pp. 320–327.

[160] *twitter, howpublished = https://twitter.com/menshumor/status/468481242153750529*.

[161] Paul Viola and Michael Jones. "Rapid object detection using a boosted cascade of simple features". In: *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*. Vol. 1. IEEE. 2001, pp. I–I.

[162] Deepak Geetha Viswanathan. *Features from accelerated segment test (fast)*. 2009.

[163] Dominic Zeng Wang and Ingmar Posner. "Voting for Voting in Online Point Cloud Object Detection." In: *Robotics: Science and Systems*. 2015.

[164] Oliver Wasenmüller and Didier Stricker. "Comparison of kinect v1 and v2 depth images in terms of accuracy and precision". In: *Asian Conference on Computer Vision*. Springer. 2016, pp. 34–45.

[165] Shih-En Wei et al. “Convolutional pose machines”. In: *CVPR*. 2016.

[166] Daniel S Weld. “Recent advances in AI planning”. In: *AI magazine* 20.2 (1999), p. 93.

[167] Norbert Wiener. *Cybernetics or Control and Communication in the Animal and the Machine*. Vol. 25. MIT press, 1961.

[168] Marcelo Worsley. “Multimodal learning analytics”. In: *Proceedings of the 14th ACM international conference on Multimodal interaction - ICMI '12*. New York, New York, USA: ACM Press, 2012, p. 353. ISBN: 9781450314671. DOI: 10.1145/2388676.2388755. URL: http://dl.acm.org/citation.cfm?doid=2388676.2388755 (visited on 10/14/2016).

[169] Marcelo Worsley. “Multimodal Learning Analytics as a Tool for Bridging Learning Theory and Complex Learning Behaviors”. In: *Proceedings of the 2014 ACM workshop on Multimodal Learning Analytics Workshop and Grand Challenge - MLA '14*. New York, New York, USA: ACM Press, 2014, pp. 1–4. ISBN: 9781450304887. DOI: 10.1145/2666633.2666634. URL: http://dl.acm.org/citation.cfm?doid=2666633.2666634 (visited on 10/15/2016).

[170] Marcelo Worsley and Paulo Blikstein. “Analyzing Engineering Design through the Lens of Computation”. In: *Journal of Learning Analytics* 1.2 (2014), pp. 151–186. (Visited on 10/17/2016).

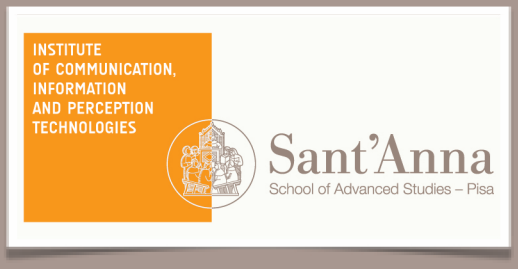
[171] Ting-Fan Wu, Chih-Jen Lin, and Ruby C Weng. “Probability estimates for multi-class classification by pairwise coupling”. In: *Journal of Machine Learning Research* 5.Aug (2004), pp. 975–1005.

[172] Zhirong Wu et al. “3d shapenets: A deep representation for volumetric shapes”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 1912–1920.

[173] Zhengyou Zhang. “Iterative point matching for registration of free-form curves and surfaces”. In: *International journal of computer vision* 13.2 (1994), pp. 119–152.

[174] Yin Zhou and Oncel Tuzel. “VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection”. In: *arXiv preprint arXiv:1711.06396* (2017).

[175] Siyu Zhu et al. “Parallel Structure from Motion from Local Increment to Global Averaging”. In: *ArXiv e-prints* (2017).


INSTITUTE
OF COMMUNICATION,
INFORMATION
AND PERCEPTION
TECHNOLOGIES
Sant'Anna
School of Advanced Studies – Pisa