TEC | Tecnológico de Costa Rica

Instituto Tecnológico de Costa Rica
Escuela de Ingeniería en Computación
Programa de Maestría en Computación

Object Tracking Based on Hierarchical Temporal Memory Classification

Tesis
Presentada en cumplimiento parcial de los requisitos para el el grado de

Magister Scientiæ en Ciencias de la Computación

Autor
Fabián Fallas Moya

Asesor
Francisco J. Torres Rojas, Ph.D.

noviembre 2015

Dedicated to my grandmothers.
Your sacrifice and love, have us here.

Acknowledgements

I would like to thank my adviser, Francisco Torres Rojas, for his guidance during this research; but especially for inspiring me to teaching and researching.

Thanks to the Masters Program for all the support with documents, extensions and requests that I needed.

I would like to thank to the Happy Few Research Group for their feedback, especially to Ricardo Román Brenes who guided me through the exciting world of statistics.

Thanks to my colleagues at UCR (Universidad de Costa Rica) for their support. Thanks to Arnoldo Rodríguez for encouraging me to apply for this academic degree and all his advice. I would like to thank Luis Flores Jiménez for giving me the flexibility to work on this thesis, his support has been a key aspect for me to stay on the academic field.

I am grateful to my family: Diego and Karla, we always encourage to keep going. Also to my sister in law and brother in law, we always wish the best. And especially to my parents, everything I am is because of you, your perseverance for me to continue my studies, has brought me here.

Special thanks to my lovely wife, Saray, who believed in me and gave me all her support during my ups and downs. Also for taking care of our son, who was born during the development of this thesis. A huge part of this thesis... is yours.

And finally, for what I believe, because without it, I could not do it. Totus Tuus.

Abstract

With recently advances in technology (hardware and software) there is an interest of humanity to have machines that behave like humans do. One aspect that researchers have to overcome is how to imitate the cognitive processes of the brain; cognitive processes like visual pattern recognition, speech recognition, space comprehension and so on. This task needs an algorithm that receives *raw* information from the environment, thus a signal processing method is needed to convert the *raw* input into useful information.

Computer Vision is an interesting field of research, because the process of capturing images is simple and the hardware to process these images is available with current technology. A natural cognitive process is *tracking objects*, for example humans (and animals) can focus on an object of their interest and follow it with their eyes; humans do this very easily but it is a very challenging problem for computers.

This research focuses on the field of *video tracking* process using an emerging technology like Hierarchical Temporal Memory (HTM). HTM is a machine learning technique that tries to imitated the Neocortex of the human brain, and then emulate cognitive processes. This research is based on creating a *video tracking* algorithm that tries to imitate the cognitive process of the brain.

Different approaches have been developed to face the video tracking problem, this research was done using HTM network to achieve this purpose.

www.tec.ac.cr

TEC | Tecnológico de Costa Rica
Programa de Maestría en Computación
Teléfono: 2550-2402

APROBACIÓN DE LA TESIS

"Object Tracking Based on Hierarchical Temporal Memory Classification"

TRIBUNAL EXAMINADOR

[illegible]

Dr. Francisco Torres Rojas
Profesor Asesor

[illegible]

M.Sc. Eddy Ramírez Jiménez
Profesor Lector

[illegible]

MSc. Ricardo Román Brenes
Profesor Externo

[illegible]

Dr. Roberto Cortés Morales
Coordinador del Programa de Maestría en Computación

Noviembre, 2015

Acronyms

ANNs	Artificial Neural Networks
CNNs	Convolutional Neural Networks
HTM	Hierarchical Temporal Memory
SVMs	Support Vector Machines
DoE	Design of Experiments
OSR	Occlusion Success Rate
SIFT	Scale-Invariant Feature Transform
OPF	Online Prediction Framework
NuPIC	Numenta Platform for Intelligent Computing
CLA	Cortical Learning Algorithm
SDRs	Sparse Distributed Representations
OPF	Online Prediction Framework
RAM	Random Access Memory
kNN	k-Nearest Neighbor
ANOVA	Analysis of Variance
PyBRAIN	Python-Based Reinforcement Learning, Artificial Intelligence and Neural Network Library

CONTENTS

LIST OF TABLES

LIST OF EQUATIONS

LIST OF FIGURES

CHAPTER 1 INTRODUCTION

“There are many things humans find easy to do that computers are currently unable to do”

Jeff Hawkins

The computing power of a computer is greater than anyone could imagine in the past. The main advantage using computers is their capability to process information faster than a human can do it, in this context the advance in computer vision is remarkable. Computer vision is a field that includes methods for acquiring, processing, analyzing and understanding images (that come from electronic devices) to produce useful information [Wik14].

Capturing video is simple, easy and accessible to any person with different devices. Processing images from these devices have become an important and an interesting task, because both software and hardware available choices. It is normal to think computer vision as a task for robots, and how they can imitate a person. Besides this interesting field (robotics) there are other areas that use computer vision to create applications, for example: video surveillance, communication, interaction with environment, pattern recognition, etc.

An important task in computer vision is to process an input image to "understand" it, and then it can perform some action. So a capability to detect and track objects of interest is needed. The process of estimating over time the location of one or more objects using a camera is referred to as *video tracking* [MC11]. Improvement in technology (software and hardware) has brought the computational power to process this task with good performance.

Another important term is *object of interest* known as *target.* What kind of target does a system have to track? It depends on the needs, if a system needs to track airplanes, it has to specialized in such objects, it has to recognize common patterns of airplanes (edges, colors, shapes) to achieve this goal. The same happens with tracking systems of persons, cars, cells (biology), animals, etc.

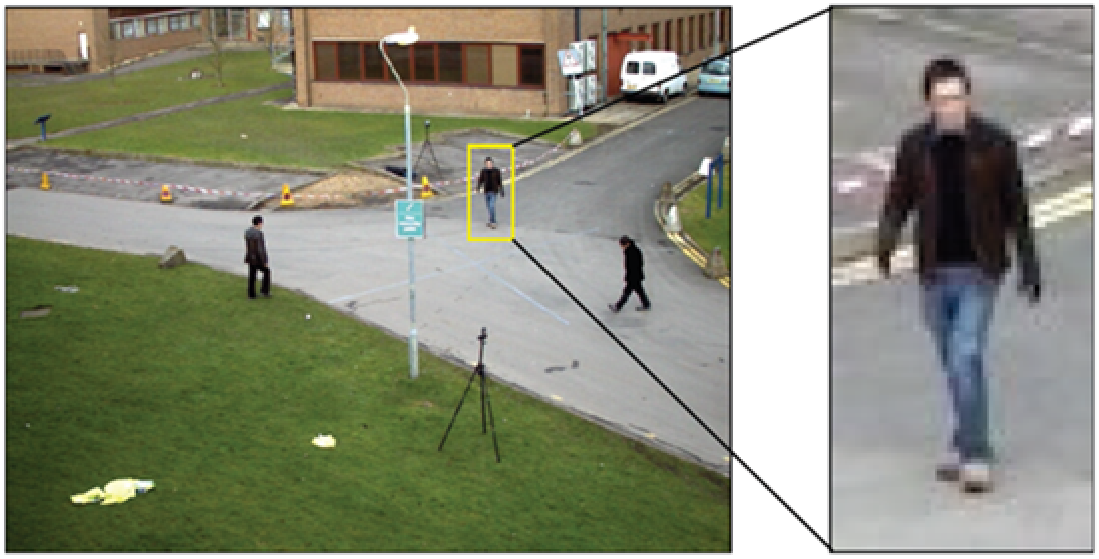


Figure 1.1 Motion capture. Taken from [Com09]

As seen in Figure 1.1, video tracking could be applied to a full body tracking (people). This type of tracking is a common research issue, due to its relevance in robotics and surveillance.

This research proposes a technique to tracking objects (people) using a new algorithm based on an HTM classifier. In the first chapter there is an explanation of *Video Tracking* concepts, main tasks and importance, also a description of HTM techniques and background. Chapter two focuses on the hypothesis, objectives and scope, meanwhile the implementation details are presented in chapter three. Chapter four shows up the experimentation and analysis of results; finally, conclusions and future work are presented on chapter five.

1.1 Video Tracking

As mentioned before, *Video Tracking* is the process of finding the location of one or more moving objects in each frame (image) of a video sequence [SS11], as shown in Figure 1.2. A *target* is the *object of interest* that is tracked by a system; one system can track animals, vehicles, micro-organisms or another specific object, but the tracking of people has an enormous amount of attention (see Section 1.1.1), because it can produce important information in areas like public transport, traffic congestion, tourism, security and business intelligence (markets).

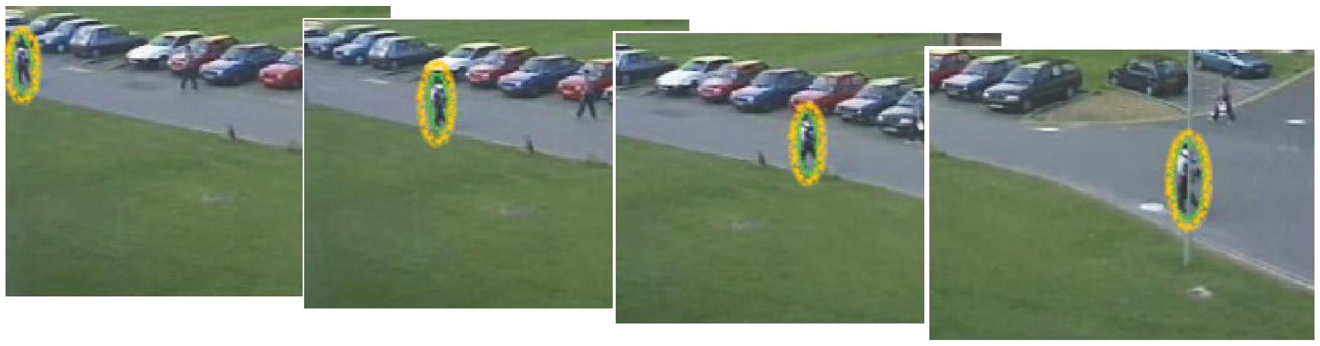


Figure 1.2 Object tracking: Graphical description. Taken from [Com09]

Maggio and Cavallaro [MC11] give us a formal general definition of the video-tracking problem. For a single-target tracking (only one object to track) let $I = \{I_k : k \in \mathbb{N}\}$ be the frames of a video, k represents time, and we have $I_k \in E_I$, where E_I is the space of all possible images. The process of video tracking (only one target at a time) is the estimation of a time series:

$$x = \{x_k : k \in \mathbb{N}\}, \tag{1.1}$$

where x_k is the current "state" of the system (it is the result of processing the image) and $x_k \in E_S$, where E_S is the space of all possibles states of the system. Video tracking implies a time series, that need to map a set of input images (I_k) to a set of states (x_k). As a result, the set of all states x is known as *trajectory* of the target.

If the goal is to concurrently track multiple objects, concatenated all states into a global state is necessary, for this purpose P_k is defined as:

$$P_k = \{x_{k-1}, x_{k-2}, ..., x_{k-m}\}, \tag{1.2}$$

where m is the total of targets in the scene at time k. Zhang *et al.* [ZLN08] use this kind of model, they implemented a system where they have *global data* which is associated to different objects for the tracking process; they associate specific features to each object in the scene, producing an accurate system. Their implementation could face an important problem of video tracking: occlusion, a term than will be explained later on.

1.1.1 Motivation

Many research fields are interested in *Video tracking*, for instance the entertainment industry has a special interest because it needs a tracking system to capture character movements and then mix it with another environment; also video gaming captures user movements to interact with a video game console. In those cases person motion is needed as shown in Figure 1.3, where human movement are tracked.

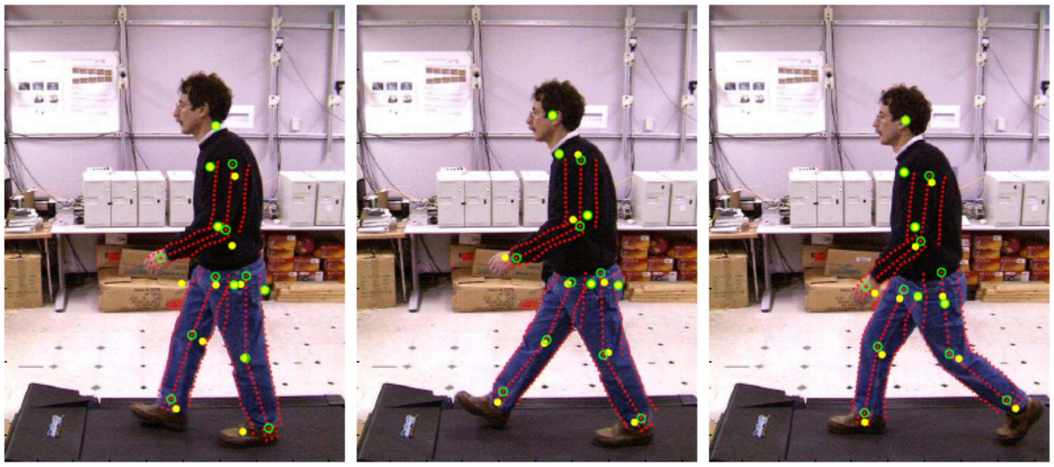


Figure 1.3 Motion capture. Taken from [MHM07]

Figure 1.4 shows another applications interested in *Video Tracking*. The following list explains some important applications:

(a) Biological Research (Figure 1.4a): tracking bacteria for a better comprehension of their behavior.

(b) Medical Applications (Figure 1.4b): motion tracking to understand certain symptoms of a disease.

(c) Surveillance and Business Intelligence (Figure 1.4c): person tracking to analyze costumers preferences or surveillance.

(d) Robotics (Figure 1.4d): human-robot interaction.

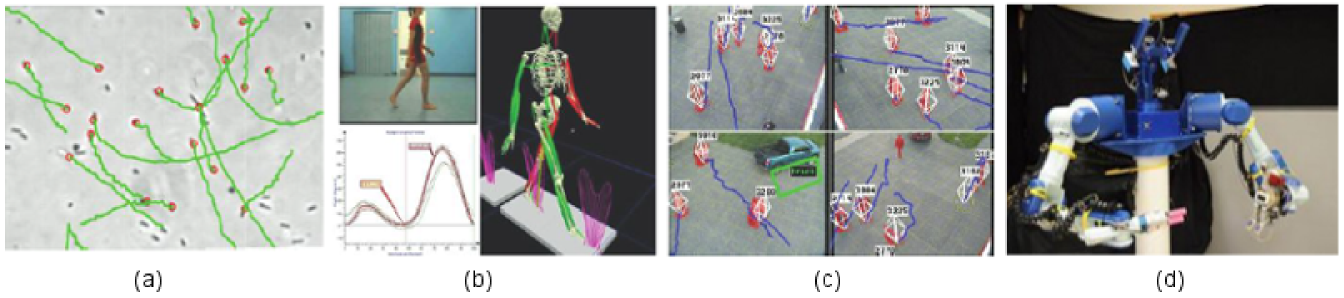


Figure 1.4 Applications of Video Tracking. Taken from [MC11, p. 3]

1.1.2 Challenges

The *input* for any tracking system is a video, which is composed of a sequence of frames. Video has the particularity, that its content could be affected by environmental issues, noise or target shape transformations. As Maggio and Cavallaro [MC11] show, there are some challenges for a video tracking system to overcome.

The first one is *clutter*, which happens when the appearance of any object or the background are similar to the appearance of the target and therefore may interfere with its observation, as shown in Figure 1.5 where objects in background have similar characteristics to the target.

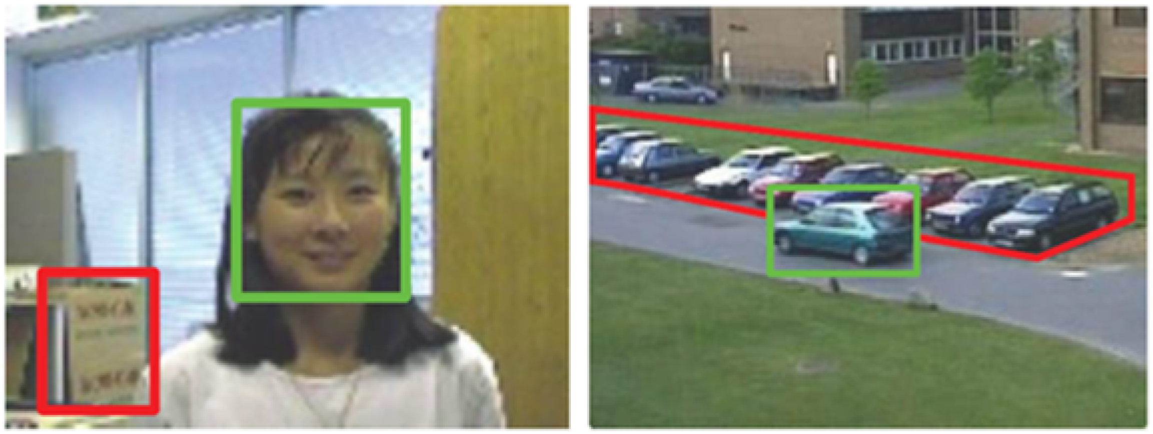


Figure 1.5 Example of clutter: objects in the background red boxes can distract the system because are similar to the target. Left: similar color. Right: similar shape. Taken from [Com01]

Changes in pose also difficult the video tracking, when a target rotates it could have a lot of new features never seen before by the system. Illumination is another factor than can complicate the tracking task, for example when the target moves from an outdoor location to a building, colors may change. Also, when the image signal has certain degree of noise, because of the input device or other external factors.

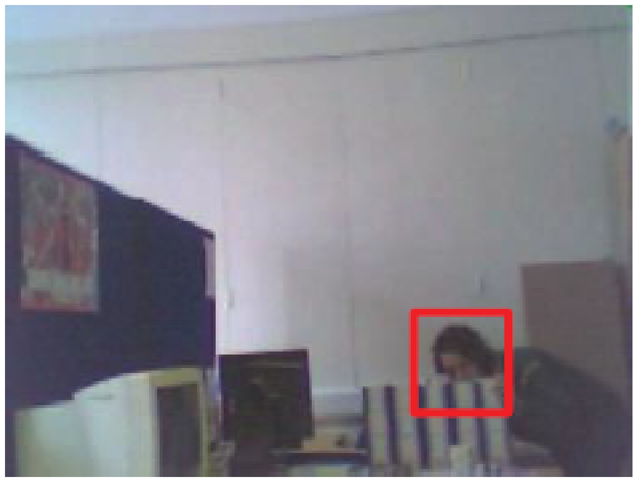


Figure 1.6 Example of occlusion: the target is partially occluded. Taken from [Com01]

Another difficult challenge is occlusion (as shown if Figure 1.6), when a target fails to be observed while partially or totally occluded by other objects. For instance when a person moves behind a car or any other object, there are some frames from the video sequence that lost the target. This last aspect (occlusion) is the main focus of this research, the following Chapter will describe this it.

1.1.3 Tasks

Maggio and Cavallaro [MC11] point out that there are fundamental tasks for the video tracking process. Here is a brief explanation of each task:

Feature extraction

Feature extraction is the first step of the tracking process. This task extracts the relevant information for the system, Figure 1.7 shows a raw image.

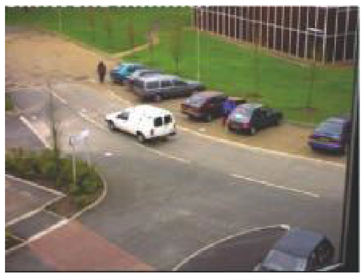


Figure 1.7 A raw image. Taken from [MC11, p. 33]

There are three types of features, that can be extracted:

- **Low-level features**: some techniques for color are used here, also gradient techniques. They analyze every pixel. Figure 1.8a shows an example where gradient features are extracted.
- **Mid-level features**: analyze a subset of pixels. It tries to find structures like: Edges, interesting points, regions, corners or uniform regions. Figure 1.8b shows a border feature extraction results.
- **High-level features**: these techniques extract relevant information, like points and their around or cluster of points. These are the most common techniques.

Figure 1.8c shows an example of high level features, where the white car is the object of interest.

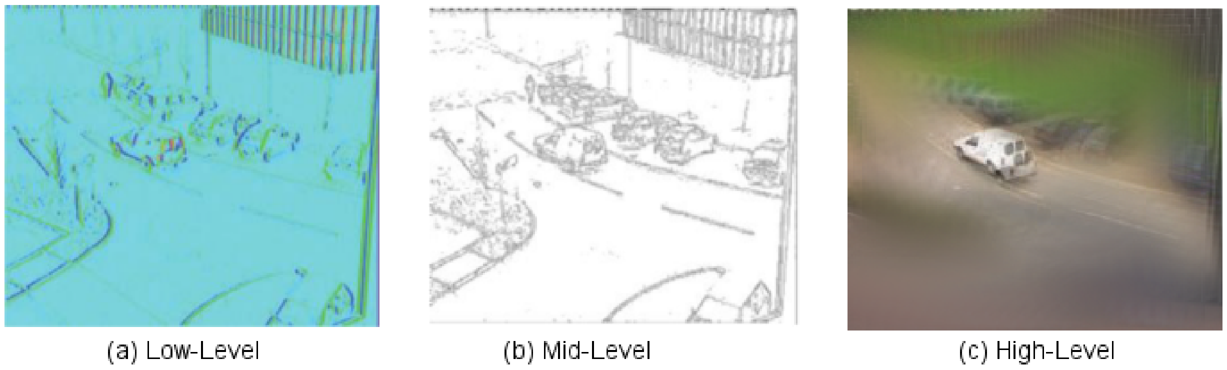


Figure 1.8 Feature extraction types: *a* shows gradient features, *b* shows border features and *c* shows template features. Taken from [MC11, p. 33]

Target Representation

Target representation is how to represent our target with the available information. As shown in Figure 1.9 there are many ways to do a shape representation of the target. Three approximations for this task are: a basic representation, that uses the center of the target for controlling the tracking process. The articulate representation combines areas and it does topological connections. Deformable representation detects multiple regions or points of interest and tracks these parts.

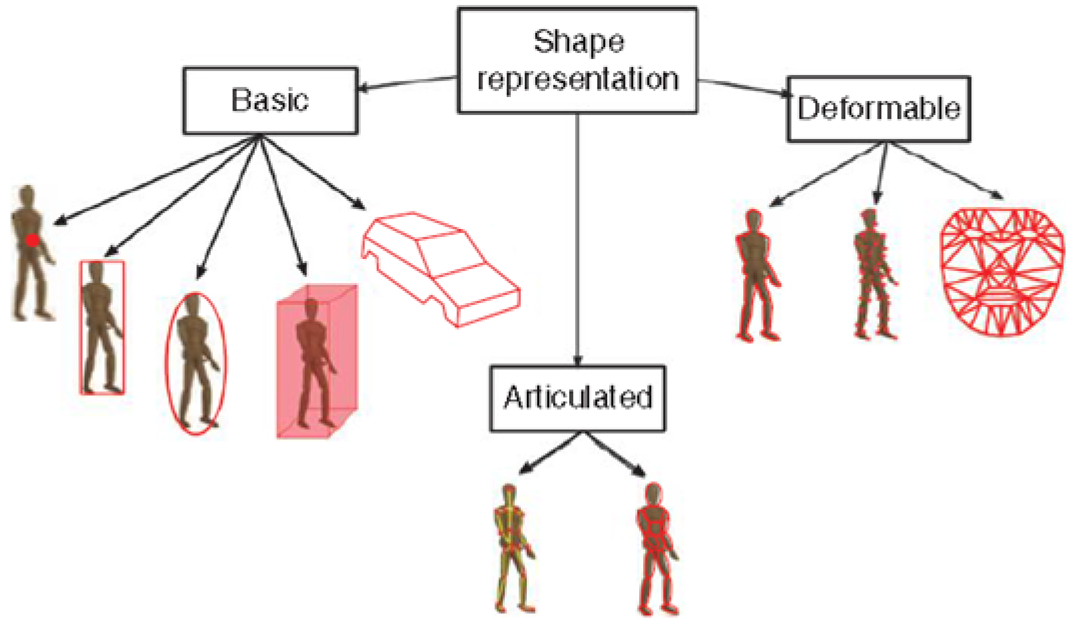
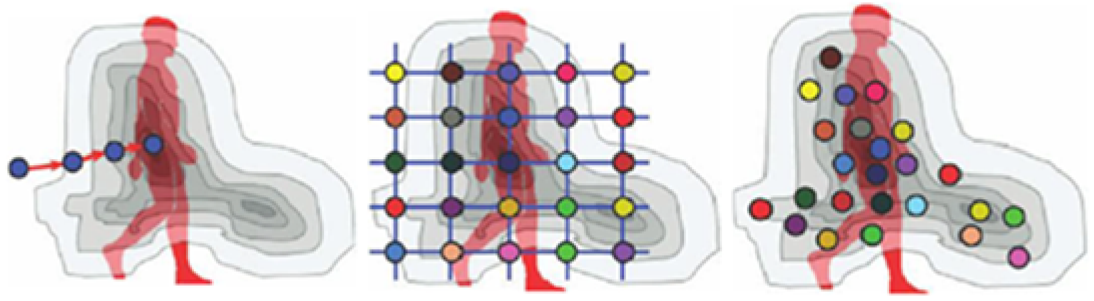


Figure 1.9 Target representation with shape: different shapes representations for video tracking (basic, articulated and deformable). Taken from [MC11, p. 72]

Localization

Localization is the process to localize a target over time, given its initial position. Over each frame the tracker estimates the current state x_k using previous state x_{k-1}.

One first approach is *Single-hypothesis localization* method, where one candidate estimate (single hypothesis) is evaluated at any time. Some examples are *Gradient-based trackers* that use image features to steer the tracking process. Using features extracted from the current state and information from the previous state, this method tries to follow the target. This algorithms also can use color histograms to measure similarity along frames sequences. Figure 1.10a shows a graphical description of a gradient-base tracking, where only one candidate is necessary for the tracking process. Singh *et al.* [SS11] point some well known processes using this technique: Mean-shift tracking, Kanade-Lucas-Tomasi tracker and the Kalman filter.



(a) Gradient **(b)** Grid **(c)** Particle filtering

Figure 1.10 Different localization techniques. Taken from [MC11, Chapter 5]

A second approach to the localization task is *Multiple-hypothesis* method, it generates multiple tracking hypothesis for each frame (Figures 1.10b and 1.10c). It evaluates multiple candidates, in every frame unlikely candidates are pruned and only the most likely candidates are propagated. A *Grid* localization technique (Figure 1.10b) uses deterministic procedures to define a grid and to distribute different candidates (they function as sensors) over the target. One of the most popular technique is *Particle Filtering* (Figure 1.10c), it is a technique that uses a probabilistic approximation (Monte Carlo); where a lot of candidates are distributed along the target, every candidate moves from frame to frame depending on its computation. Particle filtering is a very accurate implementation, and overcomes problems like clutter and occlusion, due to its multiple candidates, they can disappear or recalculate new ones.

1.1.4 Overall

The video tracking process can be defined as a process involving three tasks, as shown in Figure 1.11.

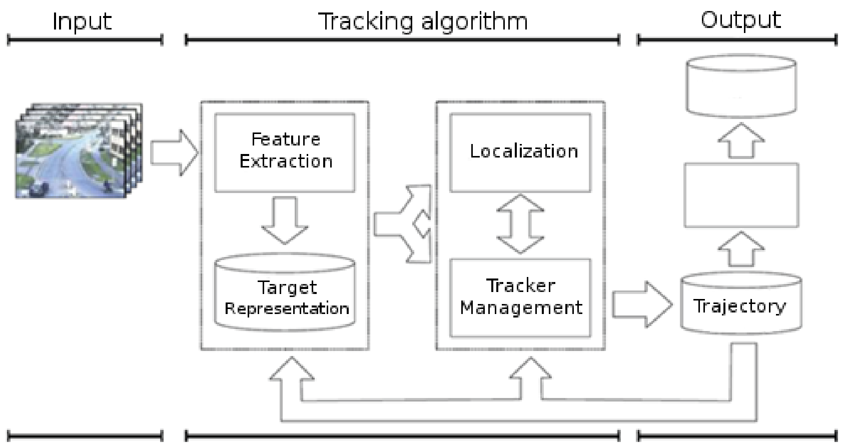


Figure 1.11 A Video Tracking algorithm. Taken from [MC11, p. 6]

The video tracking algorithm needs the development of these tasks: the feature extraction task (Section 1.1.3), target representation (Section 1.1.3) and localization (Section 1.1.3), the latter is a task that need previous states of the tracking object. This thesis proposes a new algorithm structure, where a classification process is implemented to verify if the *target* is present in the frame. Therefor, it connects the feature extraction task to the classifier, and the resulting answer of this classifier will tell us if the *target* is in the frame or not. This is a different approach, the tasks of *target representation* and *localization* were not implemented, because these tasks were out of the scope of this thesis, as explained on Chapter 2. Our algorithm uses an emerging technology for the classification process: HTM networks. This technology is explained in the following section.

1.2 HTM

The human brain has a component called *neocortex.* It performs the processes of visual pattern recognition, understanding the spoken language, the sense of touch and space comprehension [HB05]. Hierarchical Temporal Memory (HTM) is a model of algorithms based on how the neocortex works.

HTM can be understood as a specialization of ANNs, because it uses the neuron[1] concept (in HTM is more common use the term “cell” instead of “neuron”) but with a different implementation and topology. The cells of HTM are arranged in columns, these columns are arranged in regions and these regions are arranged in hierarchies. As well as ANNs, the HTM networks need to be trained with datasets to store a large set of patterns and sequences. It can learn many patterns (depending on the implementation).

1.2.1 Concepts

There are some important concepts to understand HTM. These concepts are important to understand the structure of an HTM network and the relations between its components.

Region

As Hawkins [HB05] explains, the concept of regions comes from biology where regions are sets of cells connected to other regions in the neocortex. The neocortex has six layers of cells, each layer has regions which are interconnected. Inside a layer cells, these are arranged in columns.

[1]Note that the terms *cell* and *neuron* will be used interchangeably in this thesis.

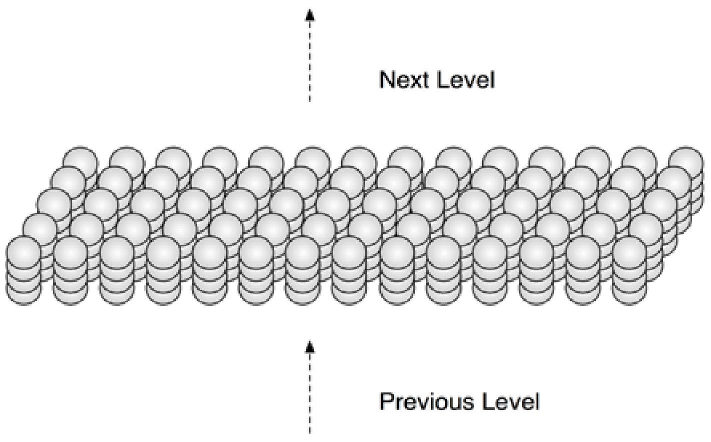


Figure 1.12 HTM region: it has a lot of cells that are arranged into arrays of columns. Each cell has n connections. Taken from [HAD11, p. 11]

These biological concepts exist on HTM, Figure 1.12 shows a graphical architecture of a region. An HTM region consists of a large number of cells, arranged into columns and these ones are connected to other cells. A cell has n connections. This structure was designed to try to imitate the neocortex.

Hierarchy

A hierarchy is a set of regions distributed into levels (hierarchy levels). These concepts are very similar to the idea develop by Fan *et al.* [FXG10], where they use *levels* of ANNs to build feature degrees and then obtained different levels of features, to do a pattern recognition process. In HTM each region represents one level[2] in the hierarchy, lower levels represents *raw* features and as one ascend through the hierarchy complex features are represented. Figure 1.13 shows this idea, where the first levels have *small* features of an object and when combining them to higher levels, more complex features are found and finally it will represent a more complex object.

[2]For our purpose level and region are synonyms.

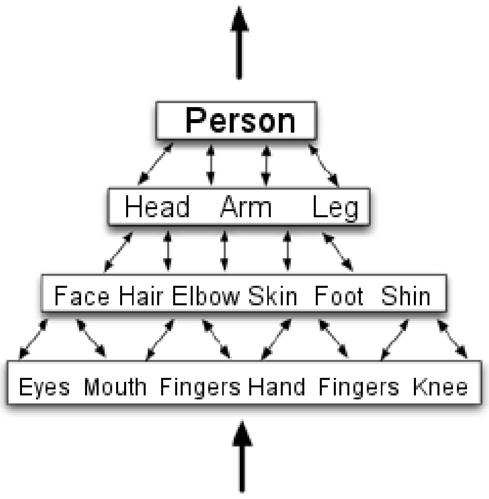

Person
Head Arm Leg
Face Hair Elbow Skin Foot Shin
Eyes Mouth Fingers Hand Fingers Knee

Figure 1.13 HTM hierarchy: a hierarchy comprised of four levels (regions).

It is important to emphasize that every region does not stores features but rather patterns. As shown in Figure 1.12 every region is composed of cells and these cells store our patterns.

As Hawkins *et al.* [HAD11] conclude, the main benefit of HTM and its hierarchical topology is efficiency. It reduces training time (in contrast of ANNs) and memory usage, because the pattern learning process at each level benefits the creation of high level features and this is a beneficial aspect for the classification process. Also the adjustment (feedback) of patterns is done very fast.

Prediction

As Hawkins *et al.* [HAD11] affirm, HTM can perform inference on inputs, it can match these inputs to previous learned spatial and temporal patterns. By matching stored sequences with current input, a region forms a prediction about what inputs will likely arrive next. This is the most robust aspect of HTM: predictions. An HTM region will make predictions based on the input, and the role of **time** is very important. The sequences are important to learn patterns and their order in time is crucial for making predictions. Hawkins *et al.* [HAD11] mentioned some key properties of HTM prediction:

- Prediction is continuous, here the order (role of **time**) is important.
- It occurs in every region at every level of the hierarchy.
- Predictions are context sensitive, they involved previous states and current inputs.
- It can predict multiple steps ahead in time (one, two, five steps).

Creating structures

A full (or complete) HTM implementation, use the following components:

(a) Raw data: it is the data in its simplest form, it can be integers, letters, floating numbers, etc.

(b) Encoders: Encoders turn different data types into data SDRs (see Section 1.2.2).

(c) Spatial Pooler: it receives the data SDRs and chooses a subset of columns (from its region). From data SDRs it creates an SDRs of columns from this region.

(d) Temporal Pooler: it receives the selected columns from the *Spatial Pooler* and it does the connections between cells (synapses), also it updates these connections over time. The most significant cells are known as the prediction cells.

(e) Cortical Learning Algorithm (CLA): it turns the prediction of the *Temporal Pooler* into the predicted value.

One option to build a full (complete) implementation of HTM is to use the OPF, which builds a full structure with all components and their connections, this research used this framework. Another option for implementing HTM, is to use each component in isolation, that is, use the *Spatial Pooler* or the *Temporal Pooler* individually. Our approach also involves this option. Section 3 has a full explanation of the implemented algorithms using these two options. Another option is to use HTM from scratch, doing a manually connection of the components, but this option was dismissed.

1.2.2 Training

The training process begins in the lower regions, once out HTM network has been trained, it switches into inference mode, this produces an output for the next regions to be train. This is process is repeated until the main higher region is reached. Every region learns different patterns through cells, these cells are connected to other cells and they are arranged into columns.

HTM also uses **Sparse Distributed Representations (SDRs)**. It is a characteristic of the neocortex. It can be explained as follows: all cells are interconnected, but only a small percentage of these cells are active at one time of the process. It means that an HTM network receives a big input, but during the training process the first layer only needs a piece of that input, then higher layers need small pieces of that and so on. *Sparse* is referred to this concept, only a small quantity of cells are active at a time. *Distributed* means that the activation of different cells are needed to represent a pattern.

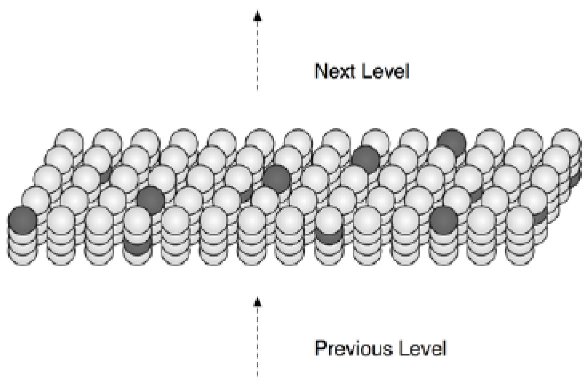


Figure 1.14 Region of a HTM: just a few cells are active at a time, showing how SDRs work. Taken from [HAD11, p. 11]

The cells activation function works with different cells, as shown in Figure 1.14 it is not necessary that cells must to be close to each other, it is not necessary for these cells to be neighbors or belong to the same column.

1.3 Related work

There are many algorithms to address the video tracking problem. Our main concern was to focus on some machine learning techniques to face this problem.

One option is to use ANNs, this model tries to simulate human neurons to respond and they are interconnected. This approach receives an input, it processes this information and it constructs an output; it is a very popular *machine learning* technique for *pattern recognition*. Takaya and Malhotra [TM05] use this concept and applied it to moving objects. It is an interesting approach but it is not accurate when trying to recognize specific targets (like full body recognition). Mussi *et al.* [MPC07] use an interesting approach where the input image is segmented to follow the target (they use a training dataset for detecting the target) and then, they applied an Evolutionary Algorithm to achieved the tracking process. Fan *et al.* [FXG10] did a remarkable research tracking objects as a learning problem of estimating the location and the scale of an object with previous information; they use a combination of ANNs to learn features. This technique first extracts and combines local features of the input image, and these features are combined by the subsequent layers in order to obtain higher order features, these actions produce a very accurate method. This research is similar to HTM but they do not developed a topology and distribution of neurons like the neocortex.

A distinct way to do pattern recognition is through SVMs. They use *supervised learning algorithms* for classification. SVMs are not interested in simulating human brain but to use an accurate classifier system to perform the pattern recognition task. Avidan [Avi04] uses SVMs to track objects, in his work he is interested on a system that detects and tracks the rear-end of vehicles from a video sequence taken by a forward looking camera mounted on a moving vehicle. He got interesting results but he could not overcome some fundamental problems like target loss. And Asha *et al.* [AKN12] developed a similar research: a vehicle detection with an aerial camera (aerial surveillance). They combined Dynamic Bayesian Networks (probabilistic model using temporal information) for tracking vehicles and SVMs as a classifiers.

Despite these two techniques (ANNs and SVMs) there is an emerging one: HTM. In this Chapter HTM technology was described and it was the motivation of this thesis: to use a model that simulated the brain structure to do a tracking task.

CHAPTER 2 HYPOTHESIS, OBJECTIVES AND SCOPE

"You don't understand anything until you learn it more than one way"

Marvin Minsky

Many techniques have been used for the video tracking problem, most of them like the *particle filtering* and *gradient based tracking* are very accurate. Most of the video tracking challenges (see section 1.1.2) are overcome by combining different techniques.

The occlusion is an interesting challenge and overcoming it was the focus of this research. To measure this aspect Occlusion Success Rate (OSR) will be used. This is a rate created for this research, it is the rate between the quantity of frames (video sequence) and the classifier hits, over a percentage of occlusion. When an OSR is high, it means that over a specific percentage of occlusion the algorithm is accurate; otherwise when OSR is low. So this rate helped to measure the accuracy degree of the different implementations.

Some research have been done with ANNs and SVMs with interesting results (see Section 1.3). All these techniques are accurate-localization algorithms, they need a step of pre-learning a representation of the object (given by training datasets). This research was based on the hypothesis that an algorithm can do a tracking process using HTM networks to obtain a low OSR.

2.1 The Problem

In a research the most difficult challenge is to find the problem of study. As Booth *et al.* [BCW08] pointed out: the problem is the starting point of any research, once you have a **practical problem**, it inspires a **question** that nobody knows what it answer is. In this point is when a researcher has found a **research problem** and with that at hand, a researcher can start the whole process of finding an **answer** to that problem. Figure 2.1 shows a graphical explanation of this idea.

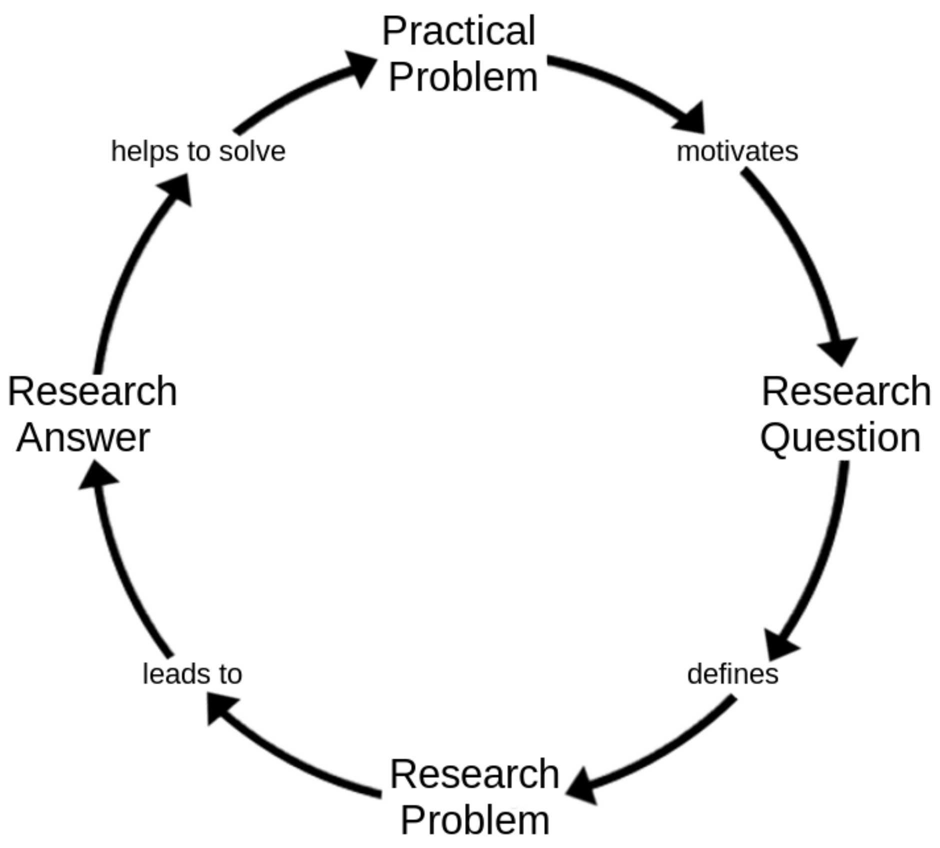


Figure 2.1 A research process, taken from [BCW08]

This thesis started on our interest on *signal processing*. Raw data is the most important thing on signal processing and it depends on the problem, it can be difficult to get it or it can be not. On that point we realized that video sequences are an easy and available source of raw data, and video tracking was found as a very interesting problem and an important topic in academic institutions (See section 1.1.1).

Based on the Figure 2.1, the practical problem found was: certain object tracking algorithms do not overcome the occlusion challenge. It motivates a research question: Can an object tracking algorithm using HTM Networks be more accurate than using other techniques? It generated the research problem. Finally, this thesis responds to that problem.

For the purpose of this thesis the term **Occlusion Success Rate (OSR)** was created, as mentioned before. It can be defined as: *given an occlusion challenge (partially* *occluded), it indicates the success rate of classification frames in a video sequence (a specific quantity of frames).*

2.2 Hypothesis

The thesis hypothesis was:

> *An Object Tracking algorithm that uses Hierarchical Temporal Memory Networks will get a statistically significant higher OSR, than using another classifications techniques.*

The main focus is doing a tracking process trying to simulate the brain structure. When a person visually tracks an object, it can be partially occluded, but that person can continue with the tracking process. It is not a problem for the human brain but a very challenging problem for a tracking system. This proposal is intended *to train* the tracker to do an accurate tracking process over a different occlusion challenges on the scene.

2.3 Objectives

2.3.1 Main Objective

To study the performance of Object Tracking algorithms using different classification techniques.

2.3.2 Specific Objectives

- Develop a prototype of an Object Tracking algorithm using three different classifiers: a HTM classifier, an ANNs classifier and a SVMs classifier.

- Compare the results from the different algorithms.
- Develop a Design of Experiments (DoE) analysis combining different factors (and their levels) over the implementations.

2.4 Scope

The scope is referred on what is going to be done by the researcher.

2.4.1 Deliverables

The following aspects describe the scope of this research, presented as deliverables:

- Four object tracking algorithms that uses different classifiers.
- A prototype that implements these algorithms to process a video sequence.
- Different data-sets and training-sets of video sequences to feed the prototype.
- Statistical Analysis of Experiments (DoE) to construct the conclusions.

2.4.2 Limitations

Due to limitations there are some things that this research will not take into account:

- Videos of different resolution.
- A complete system of video tracking.
- Another video tracking challenges, such as: illumination, clutter, video quality, etc. (See section 1.1.2).
- Real time video analysis.
- Performance analysis of classifiers (time or complexity).

- A theoretical analysis of the three different classifiers.
- Different types of target.
- Any other aspect outside the aforementioned deliverables.

CHAPTER 3 IMPLEMENTATION

"Focus on the journey not the destination. Joy is found not in finishing an activity but in doing it"

Greg Anderson

As mentioned in section 1.1.3 about the algorithm's tasks, four different algorithms were developed, using the described model in section 1.1.4.

3.1 Creating the algorithm

The four algorithms developed use three types of classification (ANNs, SVMs and HTM). This is the list of algorithms:

1. HTM's hard implementation: we use the name "Hard" to describe a full or complete implementation of HTM, it means using all the HTM components, as described in Section 1.2.1. This is a new implementation done and the details are on Section 3.2.

2. HTM's soft implementation: When we were developing this research, we realized some advises from other researchers. We took their recommended version for image recognition algorithm and adapted it to our problem. This implementation is a recommended algorithm from the Numenta Platform for Intelligent Computing (NuPIC) researchers [Num15].

3. SVMs implementation: it is an implementation using one of the most *State of the Art* libraries, it is described on Section 3.4.

4. ANNs As well as SVMs, it uses a strong research library, the Python-Based Reinforcement Learning, Artificial Intelligence and Neural Network Library (PyBRAIN). Its description is on Section 3.5.

3.1.1 Components

Figure 1.11 shows the full implementation of our object tracking algorithm. The main goal is to compare four different machine learning techniques (inside of an object tracking algorithm), that is why our algorithms differs from the normal video tracking implementation, we do not implemented the visualization and localization tasks; we were interested on the classification result. Figure 3.1 shows the Object Tracking algorithm structure developed for this research, it does not include any localization or visualization task (as mentioned), because it is not the main issue of this research, the principal focus was to detect a specific object over a sequence of frames, and then compare the performance of the classifiers.

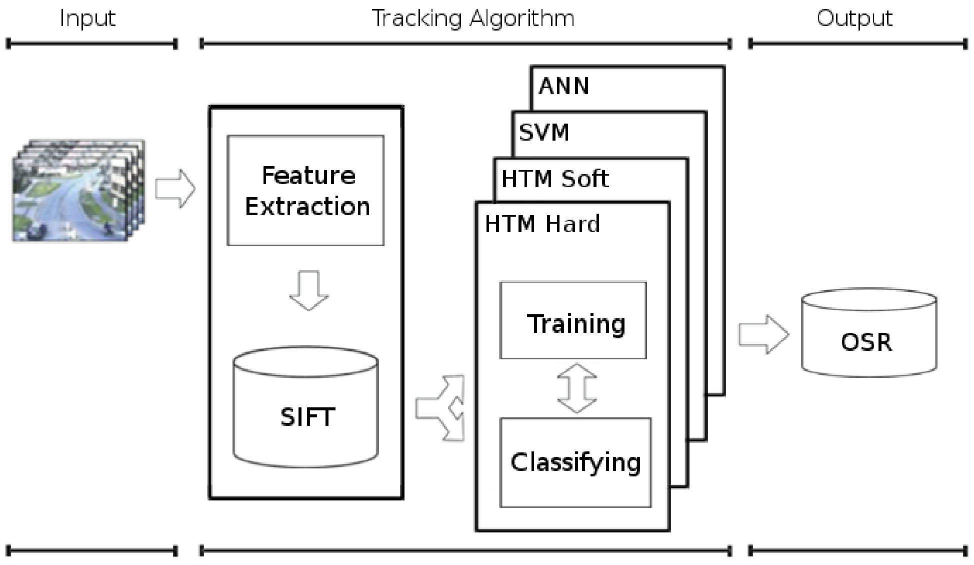


Figure 3.1 The Object Tracking algorithm structures, developed for this research.

In summary, four different types of algorithms were developed, and the result of every run was the OSR. For doing the classification, first we needed to get useful information from the raw data (frames), that is the the feature extraction process (as shown in Figure 3.1). The next Section will explained the method used for doing this task.

3.1.2 SIFT

As seen in Figure 3.1 the first task was the *Feature Extraction*, Scale-Invariant Feature Transform (SIFT) was used to achieved this goal. Section 1.1.3 mentioned that there are three types of features: low level, mid-level and high level features; SIFT features are classified as high level features because of the type of information it retrieves. This technique was introduced by Lowe [Low99] and it is one of the most successful feature extraction techniques. It does two main actions: it detects interest points from an image (called *interest point detector*) and get a local descriptor from each point (called the *descriptor*). As Solem points out [Sol12]: "SIFT features are invariant to scale, rotation and intensity and can be matched reliably across 3D viewpoint and noise", its main advantage is to avoid differences in rotation and scale, these are some common issues while analysing video frames, due to constant movement of the target.

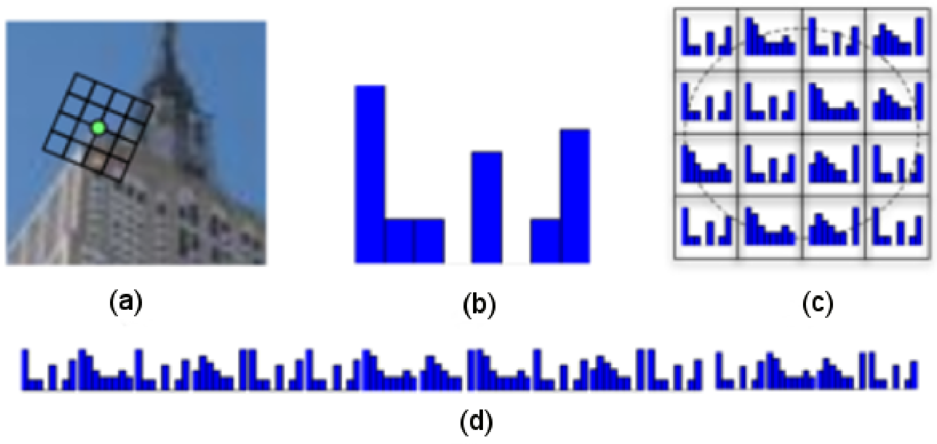


Figure 3.2 One point detected by SIFT and the feature vector generated (the descriptor). (a) a frame around an interest point, oriented according to the dominant gradient direction. (b) an 8 bin histogram over the direction of the gradient in a part of the grid. (c) histograms are extracted in each grid location. (d) the histograms are concatenated to form one long feature vector. Taken from [Sol12]

Figure 3.2 explains the whole SIFT process. First, all interest points from the image are detected using *difference-of-Gaussian* functions [Low99]. Then as Solem [Sol12] describes, it takes the dominant gradient of the point based on the direction and magnitude of the image gradient around each point (the direction can be seen in Figure 3.2 (a)). Next, it takes a grid of subregions around the point (Figure 3.2 (a)) and for each subregion computes an image gradient orientation histogram (Figure 3.2 (b) shows one subregion histogram, Figure 3.2 (c) shows all subregion histograms). Finally, the histograms are concatenated to form a descriptor vector (as shown in Figure 3.2

(d)). The most common settings used are: 16 cells to describe the interest point (4 x 4) and each cell has a 8 bin histogram for a total of 128 values ($4 \times 4 \times 8 = 128$).

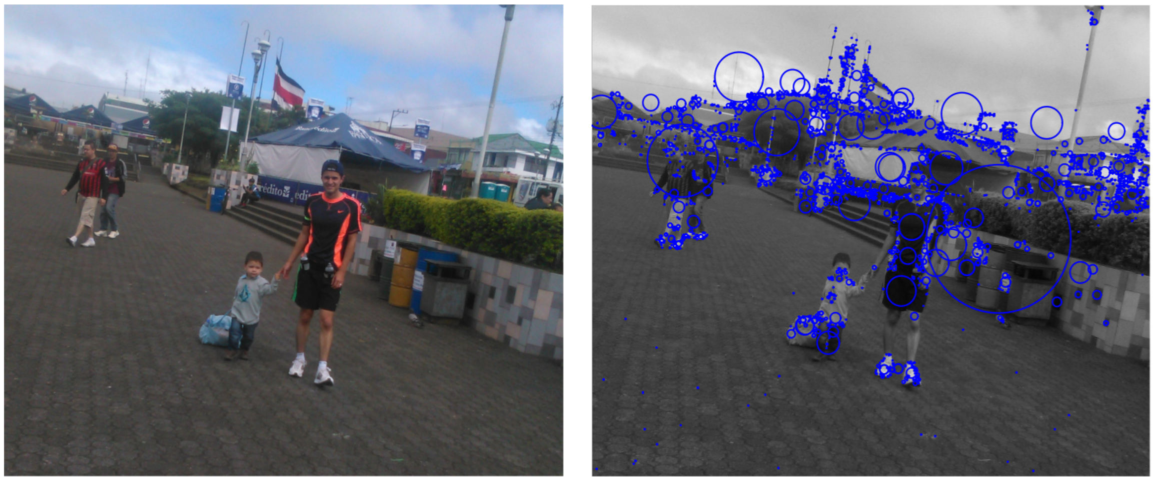


Figure 3.3 SIFT process on an image taken by us: the image on the left shows the original image; the image on the right shows the processed image. The SIFT process first converts the original image to gray scale and then the interest points are detected. A total of 3365 interest points on this image.

Figure 3.3 shows a SIFT process over an image, on the left the original image and on the right the resulting interest points detection, for a total of 3365 interest points. Each point has a descriptor of 128 values (floating values) for a total of 430720 values for the image on Figure 3.3. This quantity is big for the available hardware used on this research, that is why we implemented some restrictions: we use images of 50 pixels (50×50) and 12 interest points were chosen to apply this research.

To test the accuracy of the SIFT features a method was developed for matching descriptors, we use a proposal from Lowe [Low99]. This is a robust criteria for matching a feature in one image to a feature in another image, it uses the ratio of the distance to the two closest matching features. Figure 3.4 shows the result of this method from two different images, one image is similar to the image on Figure 3.3, and the other image is edited to show just the child of the picture.

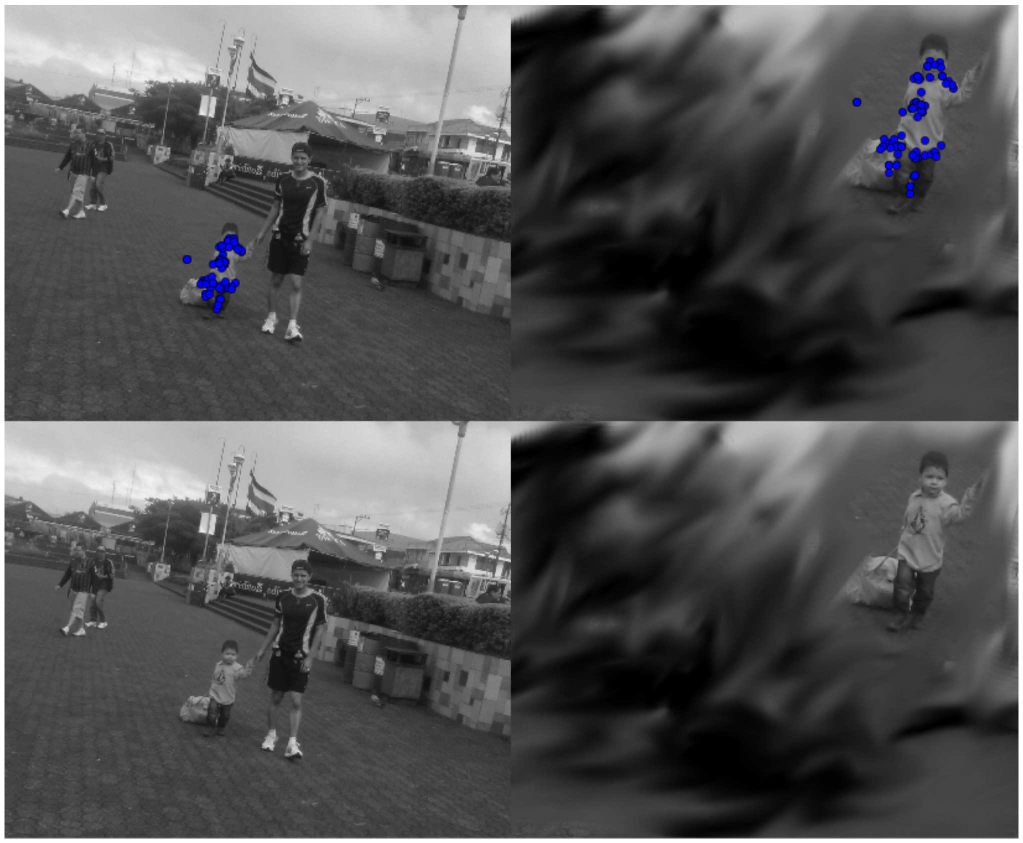


Figure 3.4 Method for matching descriptors from two different images. Lower side images: the image one (left lower side) and the image two (right lower side) to run the method. Upper side images: the results from the method, they show the points that are equal from image one (left upper side) and image two (right upper side).

The original images are shown on the left and right lower side, the resulting images are shown on the left and right upper side. As seen there is an effective accuracy for detecting the descriptors. This was a test to be ensure of the SIFT properties, the next Section is going to specify the technique that we used in this research, due to the limitations mentioned before.

3.1.3 Dense SIFT

Figure 3.5 shows an example of an image used for doing our classification process. It is a person on the left side raising his hands (wearing a red shirt), this was one frame of the videos we recorded. It is important to point out that this implementation is different to normal SIFT, because 12 static points are established and the radius of each point is increased, to form a grid of circles, this implementation is called *Dense SIFT*. The benefits of this approach is to have SIFT files of the same size and to recognize any changes on the image. The current machine learning techniques chosen for this research are classifiers, hence the importance of using SIFT files of the same size. For this research we extracted features from images via Dense SIFT.

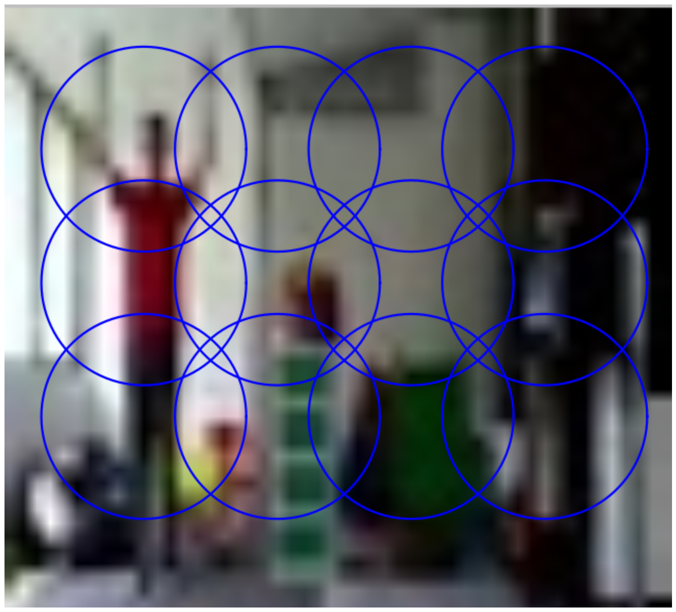


Figure 3.5 Our implementation of SIFT: it is an approach using a common technique called *Dense SIFT*. Establishing 12 points and increasing their radius of each one. The image is taken from one of the videos we recorded.

As Han *et al.* say [HLJ11], in summary, **Dense SIFT** can be described as: (a) the location of each keypoint is not from the gradient feature of the pixel, but from a predesigned location; (b) the scale of each keypoint is all the same which is also predesigned; (c) the orientation of each keypoint is always zero. With this assumptions, Dense SIFT can acquire more feature in less time than SIFT does.

Hassner *et al.* [HMZ12] also uses Dense SIFT to classify images and got better results than using any another technique. Han *et al.* [HLJ11] also developed an interesting research classifying people.

3.1.4 Target

As Han *et al.* [HLJ11] did, this research focuses on people. As mentioned in Section 1.1.1, there are many interesting applications where object tracking can be used. Locating people in a video sequence have many potential applications, such as: building security, person recognition, human computer interaction and so on.

3.1.5 Scenario

The video sequences where done bearing in mind two types of scenarios. Figure 3.6 shows an example of these two types: a simple scenario (3.6 left) which has a white background and only a few objects for a future target interaction (and the occlusion challenges); and a complex scenario which has more objects, distance and light contrast. These two types of scenario helped to have a diversity of experiments.

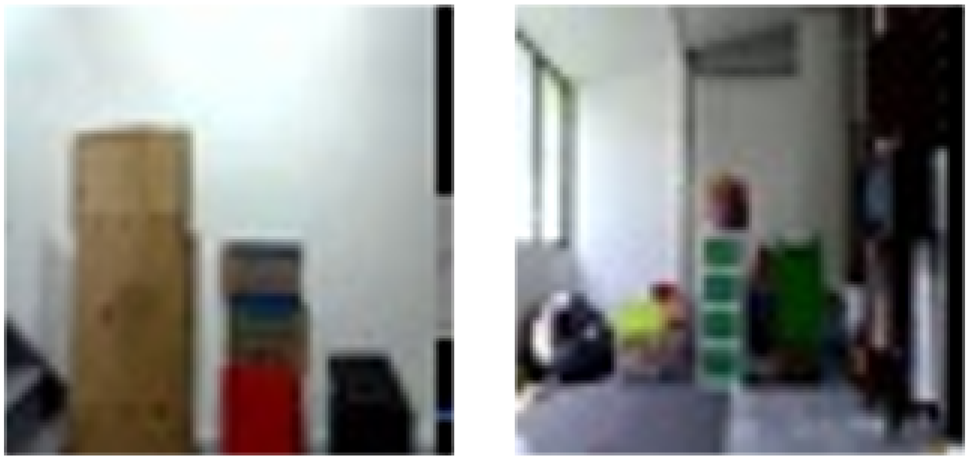


Figure 3.6 Two types of scenarios: simple (left) and complex (right).

3.2 HTM Hard

HTM is a methodology for creating Neural Networks, the team behind HTM are aware of it (Hawkins *et al.* [HAD11]), but the main difference is the logic used and the structure behind this machine learning technique, as explain in Section 1.2. The framework

NuPIC gives the proper environment to use HTM. There are many possible ways to use HTM, as pointed in Section 1.2.1. We created a new algorithm based on a full structure of HTM.

3.2.1 OPF

In order to create a complete structure of an HTM network (as shown in Figure 3.7) the Online Prediction Framework (OPF) tool was used. This is a framework which creates an HTM structure with all components of this technology. In Section 1.2.1 is a brief explanation of each module.

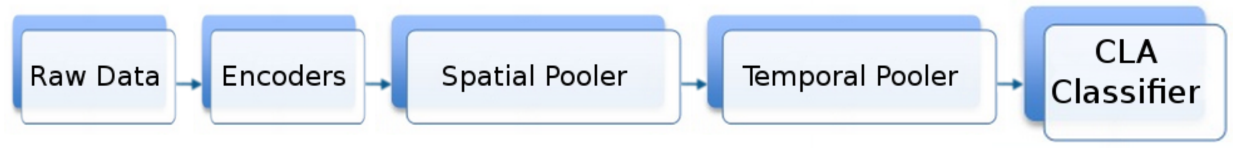


Figure 3.7 A complete structure of a HTM network. It is a complete hierarchy.

It is important to say that a HTM structure is a *prediction* technique, as mentioned in Section 1.2.1. That is why a deep modification had to be done (based on Costa's proposal [Cos14]), because a classifier is needed. The first step was to create a model in OPF, this model is created to handle streaming data but in a low scale, for example receiving one to ten values at a time; as told in Section 3.1.3 we use 12 interest points (from a *Dense SIFT* process), each point has 128 values (from the *descriptor*), for a total of 1536 values (12×128). Figure 3.7 shows an OPF structure of HTM, it uses all the components: encoders, Spatial Pooler, Temporal Pooler and the CLA classifier. We modified an OPF model in the following aspects:

(a) The input size: the model was modified to accept 1536 floating values. Each value has to be **encoded** into a 121 SDRs, thus an input of 185856 (1.536×121) was done. Every frame needs this input and one second has 15 frames (we are using a low resolution of video, just 50 pixels), that is a number of 2787840 (2.79 million) values processed in one second.

(b) Inference: previously it was said that HTM is meant to do predictions (see Section 1.2.1), and every component of HTM is prepared for this task. So in the OPF model this option was change to do *Non Temporal Classification*, it means that our model will not take the factor *time* into account. Our goal is to do just a classification task.

(c) Other components: the Spatial Pooler and the Temporal Pooler were modified to accept our input size (185856 values).

(d) Classifier result: to do classification, only one predicted value was enabled. Thus when an input sequence is processed, the predicted value is going to be the classification result. In summary, we are manipulating this prediction framework to do a classification task.

Algorithm 1 HTM Hard

Precondition: $frames$ are preprocessed files in a form of SIFT files, $answers$ are the expecting results (correct results).

```
1: function CLASSIFY($frames, answers$)
2:     $encodedFrames \leftarrow frames$                          ▷ Encode all entries
3:
4:     for $i \leftarrow 1$ to $encodedFrames.size$ do
5:         $spResult \leftarrow SP(encodedFrames_i)$
6:         $tpResult \leftarrow TP(spResult)$
7:         $predictedValue \leftarrow CLA(tpResult)$
8:
9:         if $predictedValue = answers_i$ then
10:            $\delta \leftarrow \delta + 1$
11:        end if
12:    end for
13:
14:    OSR $\leftarrow \frac{answers.size}{\delta}$               ▷ OSR: Occlusion Success Rate
15:    return OSR
16: end function
```

As can be seen in Algorithm 1 there is a sequence process. First the raw data (in our case SIFT files) is converted into SDRs by the *encoders* (line 2). Then, these SDRs are processed by the *Spatial Pooler* (line 6) and its output is processed by the *Temporal Pooler* (line 7); finally by the CLA (line 8). The latter one gives a predicted value, that is, our classification result. The last step of the algorithm is to calculate the **OSR** (lines 10 to 12) previously defined in Section 2.1.

The training algorithm is similar as the shown in Algorithm 1, the difference is that the learning process is activated on training and there is no calculation of the OSR.

An interesting aspect of this implementation is that the processing time is very high than the other implementations. A simple run of this algorithm can take 60 or 70 times more than others. This is because the original entry of 1536 has to become in an entry of 185856 (explained in Section 3.2.1) and the OPF model has to build all the structure and relations between components, this produces a significant consume of computational resources. But this is not an important issue (available resources), what matters is the accuracy of the classification process.

3.3 HTM Soft

The previous Section explained a full implementation of HTM (Figure 3.7), using the OPF tool. This Section presents another way to use HTM, combining a module of HTM (specifically the **Spatial Pooler**) with a popular method for classification (specifically the **kNN**). Figure 3.8 shows the components used on this implementation.

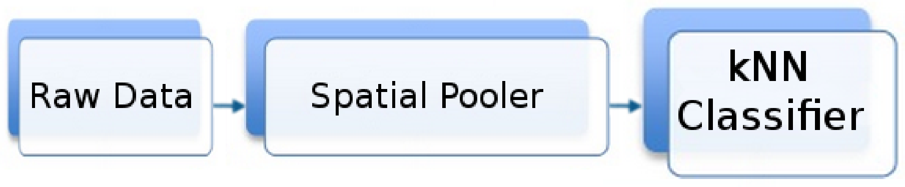


Figure 3.8 A simpler implementation of HTM. Spatial Pooler directly connected to a kNN classifier.

kNN is one of the most used methods for classification, it simple compares an object (in our case a vector of values) to be classified with all objects in a training set with known class labels and lets the k nearest vote for which class to assign. Into NuPIC there is an implementation of this algorithm and some researches have used this implementation to classify images, the official documentation of NuPIC advises the use of this implementation for vision problems [Num15]. An individual Spatial Pooler module plus the kNN algorithm was used to implement this approach.

Algorithm 2 HTM Soft

Precondition: $frames$ are preprocessed files in a form of SIFT files, $answers$ are the expecting results (correct results).

```
1: function CLASSIFY($frames, answers$)
2:
3:     for $i \leftarrow 1$ to $frames.size$ do
4:         $spResult \leftarrow SP(frames_i)$
5:         $resultingClass \leftarrow$ kNN($spResult$)
6:
7:         if $resultingClass = answers_i$ then
8:             $\delta \leftarrow \delta + 1$
9:         end if
10:    end for
11:
12:    OSR $\leftarrow \frac{answers.size}{\delta}$
13:    return OSR
14: end function
```

This implementation only uses the Spatial Pooler module from HTM, unlike *HTM Hard* implementation (as shown in Figure 3.8) that uses more components. Algorithm 2 shows that the layer *encoders* is not used here. The first step is to send the raw data

(SIFT values) to the Spatial Pooler (line 4), this produces an output with the form of a SDRs, this output is called *active array*; then, this active array is send to the kNN classifier (line 5), and it responses with the class (hopefully the correct one).

3.4 SVMs

SVMs are a very popular type of classifiers. As Solem [Sol12] explains: a SVMs finds a linear separating hyperplane with the best possible separation between two classes. With a feature vector (a SIFT point in our case) x, the decision function is:

$$f(x) = w \cdot x - b \tag{3.1}$$

where w is the hyperplane *normal* and b an offset constant, they are found by solving an optimization problem on a training set of labelled feature vectors x_i (SIFT points) with labels (*categories*) $y_i \in \{-1, 1\}$ so that the hyperplane has maximal separation between the two classes. The *normal* is a linear combination of some of the training feature vectors:

$$f(x) = \sum_{i}^{n} \alpha_i y_i x_i \tag{3.2}$$

where n is a subset of training vectors, as a result our final decision function is:

$$f(x) = \sum_{i}^{n} \alpha_i y_i x_i \cdot x - b \tag{3.3}$$

The selected training vectors x_i are called *support vectors* because they help define the hyperplane. If the feature vectors (the inputs) are non-linear, it uses *kernel functions* to map features vectors to a different dimensional space. These *kernel functions* replace the inner product of the classification function, $x_i \cdot x$ (on Equation 3.3), with a function $k(x_i, x)$. A *linear kernel* was used in our implementation, it is just a simple case where $k(x_i, x) = x_i \cdot x$.

LibSVM is a library providing a commonly used implementation of SVMs [CL01] and it was the library used here. The Algorithm 3 shows the pseudocode of the implemented algorithm. As it can be seen, this code is simpler than the previous codes; it just needs the raw input (SIFT vectors), then it starts the classification process (line 4) and finally it calculates the OSR.

Algorithm 3 SVMs

Precondition: $frames$ are preprocessed files in a form of SIFT files, $answers$ are the expecting results (correct results).

```
1:  function Classify($frames, answers$)
2:
3:      for $i \leftarrow 1$ to $frames.size$ do
4:          $resultingClass \leftarrow$ SVM($frames_i$)
5:
6:          if $resultingClass = answers_i$ then
7:              $\delta \leftarrow \delta + 1$
8:          end if
9:      end for
10:
11:     OSR $\leftarrow \frac{answers.size}{\delta}$
12:     return OSR
13: end function
```

Figure 3.9 shows the visual connection of the components used on this implementation. The factor of time was not required, but it is interesting to point that this was the fastest implemented algorithm.

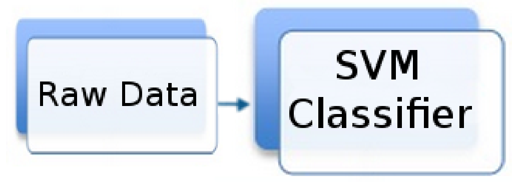


Figure 3.9 The SVMs implementation, it receives directly the raw data.

3.5 ANNs

ANNs are a technique of machine learning inspired in biological neurons and their connections, to exchange messages between each other. Also they have weights that can be adjusted to learn patterns.

Algorithm 4 ANNs

Precondition: $frames$ are preprocessed files in a form of SIFT files, $answers$ are the expecting results (correct results).

```
function CLASSIFY($frames, answers$)

    for $i \leftarrow 1$ to $frames.size$ do
        $resultingClass \leftarrow$ ANN($frames_i$)

        if $resultingClass = answers_i$ then
            $\delta \leftarrow \delta + 1$
        end if
    end for

    OSR $\leftarrow \frac{answers.size}{\delta}$
    return OSR
end function
```

Algorithm 4 shows up the implemented algortihm with ANNs. It presents the same structure of the SVMs algorithm, because it can receive the created *raw data* (our SIFT vectors). PyBRAIN introduced by Schaul *et al.* [Sch+10], was used to implement an ANNs. In Figure 3.10 the structure of this implementation can be seen. Similar to the SVMs implementation, this algorithm can receive our *raw data* (SIFT vector) without any previous processing.

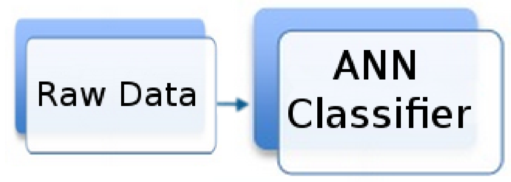


Figure 3.10 The ANNs implementation, it receives directly the raw data.

A **Feedforward Neural Network** was implemented to do the classification task. With this kind of ANNs, the information moves in only one direction: forward; from the inputs through the hidden nodes and later to the output nodes. The selected configuration was: an input layer that accepts 1536 values (the size of the SIFT vector) and the output layer has just three neurons, because it has to show the result of the classification task with few neurons.

This implementation only has one hidden layer, as Bishop says [Bis95] if there is only one input, there seems to be not advantage to using more than one hidden layer. The quantity of neurons in a hidden layer is a very challenging problem, because this unit depends on: the number of inputs and outputs, the number of training cases, the amount of noise in the targets, the architecture, and many others factors. In most situations, there is no way to determine the best number of hidden units without training several configurations and comparing accuracy. That is why the *rule of thumb* was chosen to establish this unit, as Blum explains [Blu92] (p. 60): this unit must be selected as a number somewhere between the input layer size and the output layer size. For simplicity the number of 80 neurons was selected, because our low number of classes.

CHAPTER 4 EXPERIMENTATION, RESULTS AND ANALYSIS

"The true method of knowledge is experiment"

William Blake

It is necessary to test the hypothesis, **Design of Experiments (DoE)** was used for achieving this goal. DoE is referred to a statistical model to find out if specific factors influence over a variable of interest, and if there is any influence, quantify it [FR01]. ANOVA is a vital tool for DoE, it compares the variance among the factor means versus the variance of individuals within the specific factors. The following section will describe the DoE components.

4.1 DoE components

Response Variable

Response variables depend on what is our focus of study. This research is based on object tracking to study different classification techniques (like HTM) and compare their results. The factors and levels (in the following section) were defined as a result of one response variable. The main goal is to know if these factor have an influence over these response variables.

Our response variable was:

- **OSR**: previously define in Section 2.1. It measures the percentage of classifying hits in a video sequence. This rate helped to measure the different classifiers accuracy.

Factors and Levels

During the experimentation the results are affected by different factors. It is mandatory to know which factors have a real influence and measure that influence; thus it is necessary to establish these factors. Here our factors are:

- **Technique**: classification technique to be used during the *Object Tracking* process, these techniques were defined in Section 3.

- **Training-set**: the size of the training set of the algorithm. There are three sizes, Table 4.1 in column *Training-set* shows them, it is the column labeled *Training-set*.

- **Occlusion**: the percent of occlusion of the *target*. As mentioned in Section 3.1.4, the targets are people (human shape). It is important to say that during tests a person was occluded a given percentage (showed on Table 4.1 on the column *Occlusion(%)*).

- **Scenario**: the complexity of the scenario. Only two scenarios were done (specified on Section 3.1.5).

There could be more interesting factors but due to time and budget limitations, only four factors were evaluated. Table 4.1 shows up the factors mentioned above and the corresponding levels of each factor, known as *experimental domain*.

	Factors			
	Technique	Training-set	Occlusion(%)	Scenario
Levels	HTM Hard	38	75	Simple
	HTM Soft	176	50	Complex
	ANNs	474	25	
	SVMs		0	

Table 4.1 Factors and levels to be analyzed in this research.

As a result, Table 4.1 gives us a number of 96 combinations ($4 \times 3 \times 4 \times 2 = 96$) and the number of replicas is 4, for a total of 384 runs. This number was chosen because during every run a video sequence is analyzed and each video sequence can have hundreds of frames. In out case, these are some examples:

- For training: one replica does a 32 training runs of 38 frames, 32 of 176 and 32 of 474 frames, for a total of 22016 classification tasks.

- For testing: one replica does a 24 testing runs for every occlusion challenge (0%, 25%, 50%, 75%), each testing run has a 180 frames, for a total of 17280.
- Each replicas does a 39296 classification tasks, for a total of 157184 classification tasks.

Software

The software used to statistical computing was R[1]. It is very widely used among statisticians and researchers for data analysis [Iha98].

4.2 Data

The data are a set of videos, these videos are of low quality (50×50 pixels) for simplicity and there was a hardware computing restriction: the HTM Hard implementation requires approximately 600 times more Random Access Memory (RAM) than the other implementation; because it has to build the full OPF structure and there is not precalculated values. Also these videos only have 15 frames per second. These videos were done with the following rules:

1. Every replica has two types of scenario (*subreplicas*): simple and complex (as explained in Section 3.1.5).
2. Each subreplica has a 36 seconds train video, for a total of 540 frames ($36 \times$ 15frames). Depending on the size of the training set (see Table 4.1), the different algorithms were trained.
3. This research focuses on 3 classes: a male, a female and the background. For the background 2 videos were taken, with different positions of the *occlusion objects*.

[1]Available on the site: www.r-project.org

The videos were done with our targets moving all over the scene, with different occlusion percentage.

4. For every subreplica there were a 12 seconds test video for every occlusion percentage, that is, 4 videos of occlusion challenge for the male target and 4 videos for the female target.

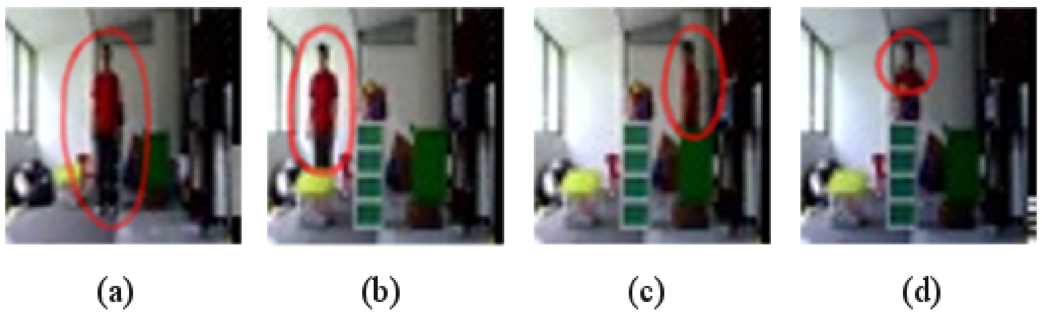


Figure 4.1 Occlusion challenges at different percentage: (a) 0% of occlusion, (b) 25% of occlusion, (c) 50% of occlusion, (d) 75% of occlusion. In every image, the target is enclosed in a red circle for visualization purposes, but it is not present on real data.

These rules help to make all the replicas videos, every replica had a total of 24 videos. The summary of the replica is:

- Background 1: the background with occlusion objects.
- Background 2: the background with occlusion objects in different positions.
- Training sets: the targets (male and female) moving on the scene (2 videos, one for each target).
- Testing sets: one video for every target, implementing an occlusion percentage (0%, 25%, 50% or 75%). Examples of these occlusion percentages can be seen on Figure 4.1.

4.3 Results

4.3.1 First “Look” at the data

Figure 4.2 shows the resulting histogram of data frequency plot. As can be seen it does not have a normal distribution, this is a positive aspect due to some interesting “peaks” in the histogram.

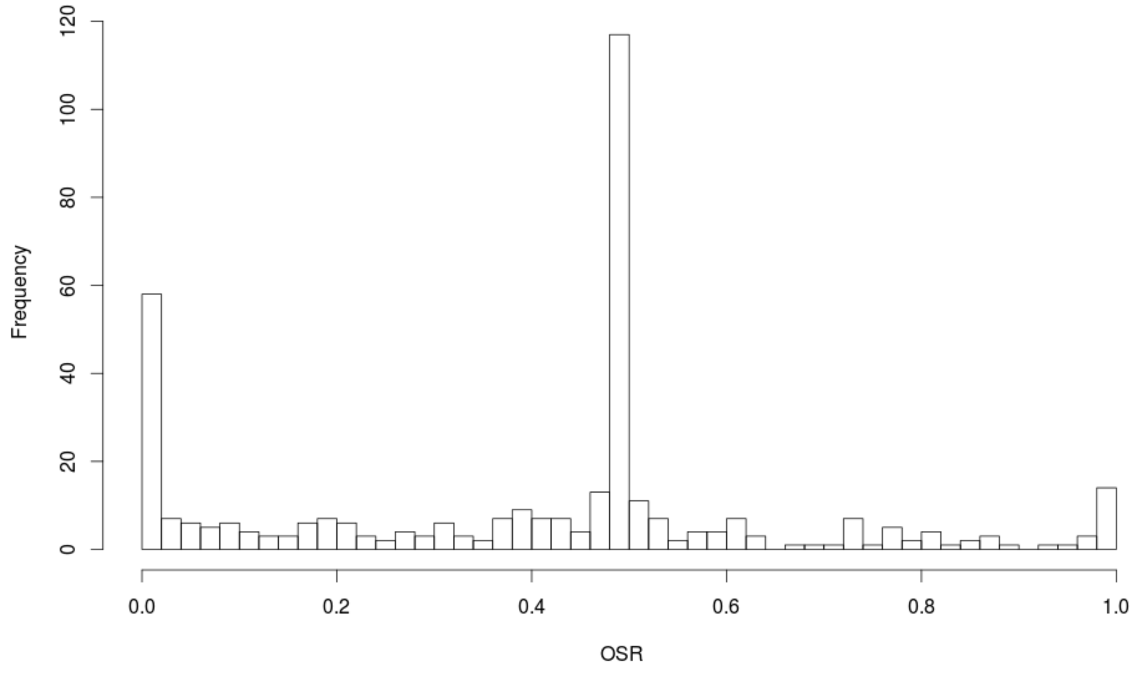


Figure 4.2 Histogram of data frequency.

Is important to remember that this plot shows the frequency of the response variable (OSR). There are three values to stand out: 0.0, 0.5 and 1.0 (our “peaks”). The first one is the value where the classifier failed completely, as mentioned in Section 4.2, the data has a low quality (due to some limitations, see Section 4.2). Hence the classifiers tend to failed. The second “peak” of 0.5 can be explained for the quantity of classes presented in the data. In our data set only two classes were taken into account to classify (male and female), thus there is more chances to get just the 50% of accuracy;

probably if more classes were used, this “peak” could be more distributed. The third “peak” is the 1.0, where the classifiers got the 100% of accuracy; an interesting aspect is that this value is isolated, from the middle of this plot to the right, the value of 1.0 is the highest, showing that there are specific conditions that helped to have a complete accuracy.

Figure 4.3 shows the histogram frecuency for every classification technique. They preserve the pattern from the general histogram (Figure 4.2), but there are some interesting aspects. The first aspect is that HTM Hard (Figure 4.3a) focuses mainly in the 0.5 peak, its rates are very consistent. HTM Soft (Figure 4.3b) has more distribution and a more failures. Hard and Soft have few complete (1.0) success rates.

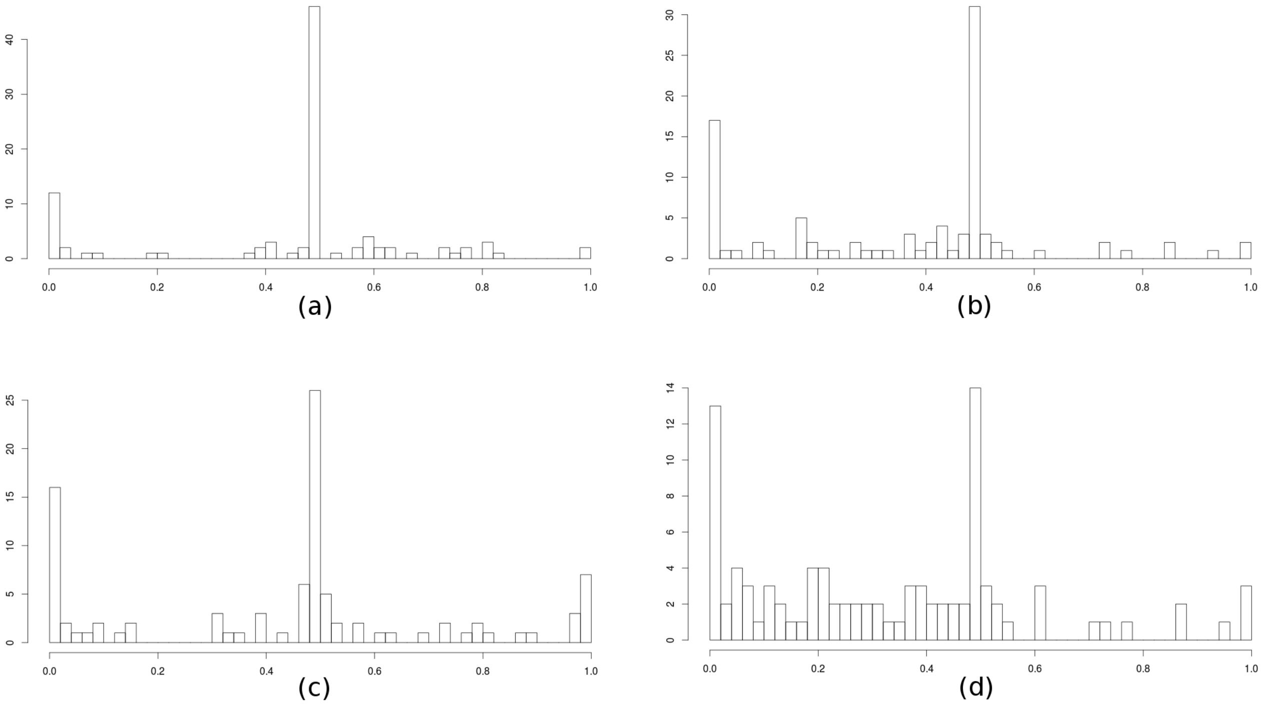


Figure 4.3 Histogram of data frequency for every technique: (a) HTM Hard, (b) HTM Soft, (c) SVMs, (d) ANNs.

Figure 4.3c shows the histogram for the SVMs, which is constant with the general histogram and it has a better 1.0 hits. Finally the ANNs (Figure 4.3d) shows more failures (0.0) and from 0.0 to 0.5 a normal distribution, which means it can be similar to random values.

4.3.2 ANOVA

To use ANOVA, it is necessary to establish a *null* hypothesis and an alternative hypothesis; also it is necessary to accomplish three assumptions.

Alternative and null hypotheses

The hypotheses of interest in ANOVA are as follows:

$$H_0: \quad \mu_1 = \mu_2 = \mu_3 = ... = \mu_k \tag{4.1}$$

$$H_1: \quad \exists \mu_j \neq \mu \qquad j = 1, 2, ..., k \tag{4.2}$$

The null hypothesis in ANOVA (equation 4.1) is always that there is no difference in means. And the alternative hypothesis (equation 4.2) is always that the means are not all equal.

Assumptions

As Anderson and Whitcomb explain [AW07] (p. 65), for statistical purposes it is assumed that residuals are normally distributed and independent with constant variance; ANOVA requires the accomplishment of these assumptions. Two plots and one action are recommended to check the statistical assumptions:

- Q-Q plot: it is a probability plot for determining if two data sets come from populations with a common distribution. In our case these two data sets are: the standardized residuals (quantiles) and theoretical quantiles of the response variable. If the two sets come from a population with the same distribution, the points should fall approximately along a 45 degree reference line.

- Residuals vs fitted values plot: it is a scatter plot of residuals on the y axis and fitted values (predicted values/responses) on the x axis. It is a tool for detecting non-linearity, unequal error variances and outliers. Ideally the vertical spread of data will be approximately the same from left to right.
- Independence of observations: it is a requirement to randomize the order of an experiment. Randomization acts as an insurance against the effects of lurking time-related variables.

Checking the assumptions

The first step was to randomize the run order to achieve the independence of observations, that way this assumption was completed.

Figure 4.4 shows the Q-Q plot, as indicated before, it determines if the standardized residuals and theoretical quantiles have a common distribution. As seen in Figure 4.4 the points fall along the reference line (the 45 degree continuous line).

The deviations from linear are very small, so it supports the assumption of normality. If any nonlinear patterns had been seen, such an "S" shape, a transformation should had been necessary; in this case it is not.

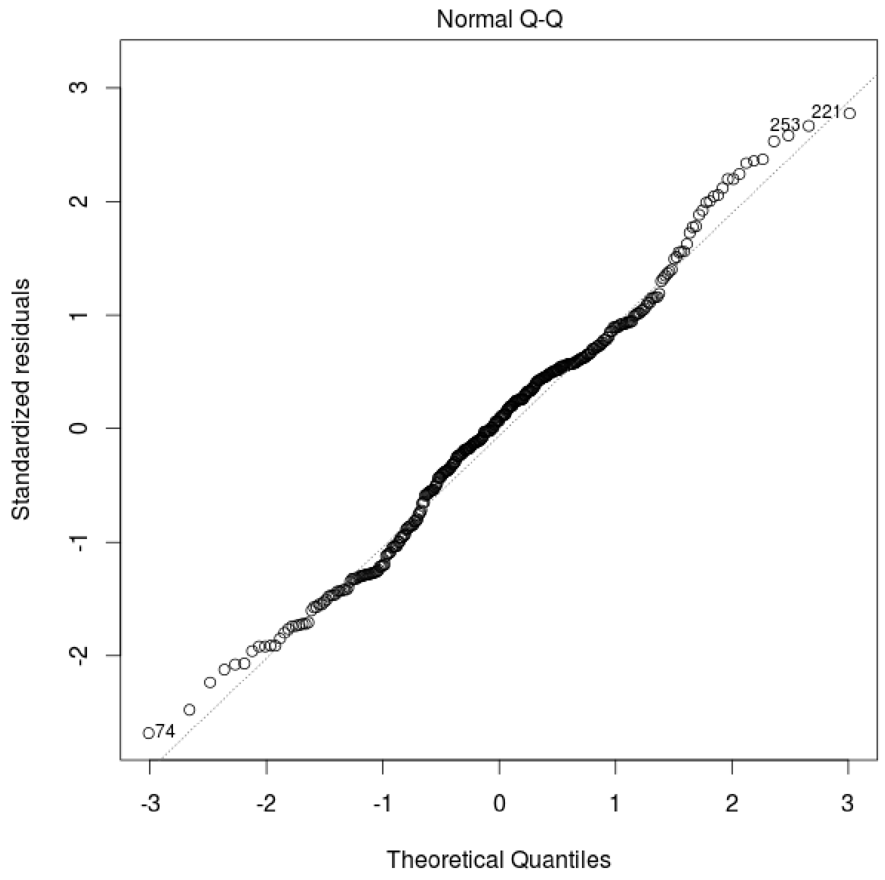


Figure 4.4 Q-Q plot: standardized residuals (quantiles) and theoretical quantiles of the response variable.

The second recommended plot is shown on Figure 4.5, the residuals vs fitted values plot. As can be seen on Figure 4.5 the vertical and horizontal spread of data is approximately the same, it is hard to identify if one side has more circles than any other.

There is not a megaphone (<) pattern, where the residuals increase with the predicted level; this is not the case, it is not a definite increase in residuals with predicted level, which supports the statistical assumption of constant variance.

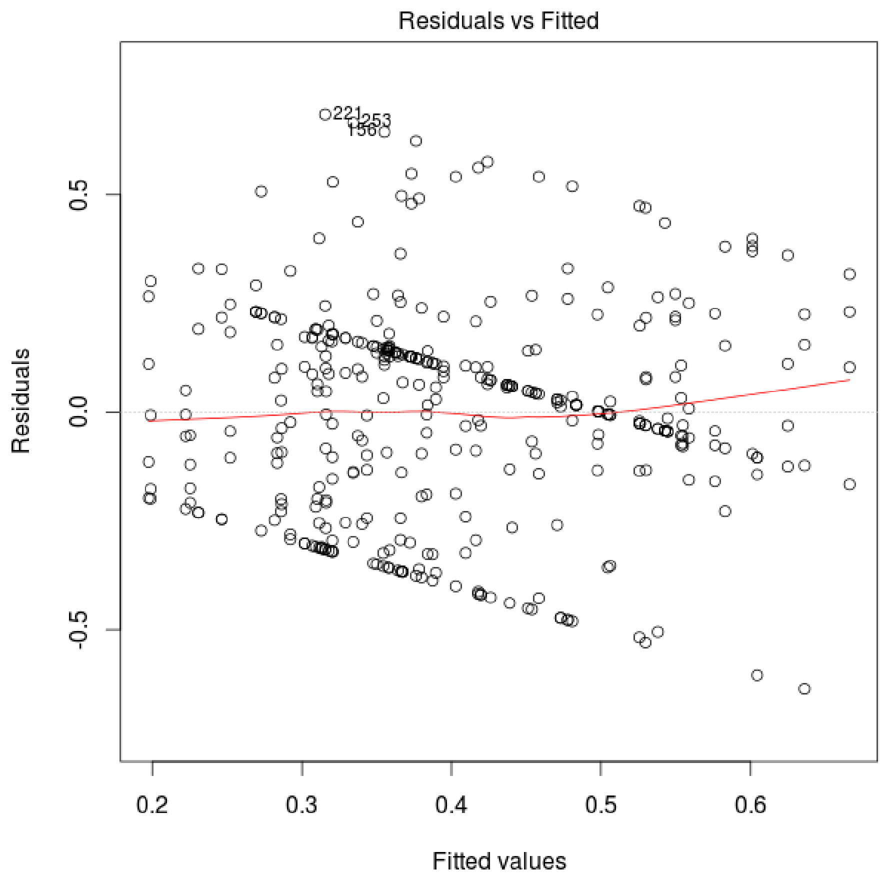


Figure 4.5 Residuals vs fitted values (predicted responses) plot.

4.3.3 Analysis of results

The Table 4.2 is an ANOVA table, with *Degrees of Freedom*, *Sum of Squares*, *Mean of Squares*, the *F values* and *Probability values* (shown as *Pr(>F)*).

On this table the probability value of *Technique:Occlusion* is 7.15^{-5}, with a significance of 0. It means that there is a complete confidence that the interaction of these factors is significant. With this aspect, at this point the null hypothesis (as mentioned in Section 4.3.2) is rejected.

There are another two interesting results: *Technique* with a probability value of 0.0022, that is a 99.78% confidence that our *response variable* is significantly affected by this factor; a similar issue happens with the factor *Occlusion*, with a probability

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Technique	3	0.96	0.32	4.97	**0.0022**
Training.Set	1	0.03	0.03	0.54	0.4623
Occlusion	1	0.51	0.51	8.02	**0.0049**
Scenario	1	0.16	0.16	2.50	0.1148
Technique:Training.Set	3	0.41	0.14	2.12	0.0970
Technique:Occlusion	3	1.44	0.48	7.49	**7.15e-05**
Training.Set:Occlusion	1	0.01	0.01	0.11	0.7369
Technique:Scenario	3	0.31	0.10	1.62	0.1840
Training.Set:Scenario	1	0.06	0.06	0.95	0.3303
Occlusion:Scenario	1	0.08	0.08	1.20	0.2746
Technique:Training.Set:Occlusion	3	0.06	0.02	0.33	0.8012
Technique:Training.Set:Scenario	3	0.20	0.07	1.05	0.3695
Technique:Occlusion:Scenario	3	0.06	0.02	0.30	0.8247
Training.Set:Occlusion:Scenario	1	0.03	0.03	0.40	0.5255
Technique:Training.Set:Occlusion:Scenario	3	0.06	0.02	0.30	0.8256
Residuals	352	22.57	0.06		

Table 4.2 ANOVA table

value of 0.0049, there is a 99.51% confidence that this factor is significant. As a result of Table 4.2, the next step is to analyze each significant element.

Technique

The *Technique* is the one of the significant factors, on Figure 4.6 the box plot (as define by Massart *et al.* [Mas+05]) for this factor is shown. This box plot helps to identify the middle 50% of the data, the median (the thick horizontal line inside the boxes) and the extreme points. One aspect is that the technique of ANNs has the lowest quartile [2] and the lowest median, showing that this technique tend to failed more than the others.

HTM Hard and SVMs have similar boxes and median, but SVMs has the upper quartile and the median slightly higher, given better results. Also the maximum point of the SVMs and ANNs reaches the OSR value of 1.0 unlike the others, showing that these technique were accurate in some runs (with specific conditions).

[2]Quartile: the quartiles of a ranked set of data values are the three points that divide the data set into four equal groups, each group comprising a quarter of the data.

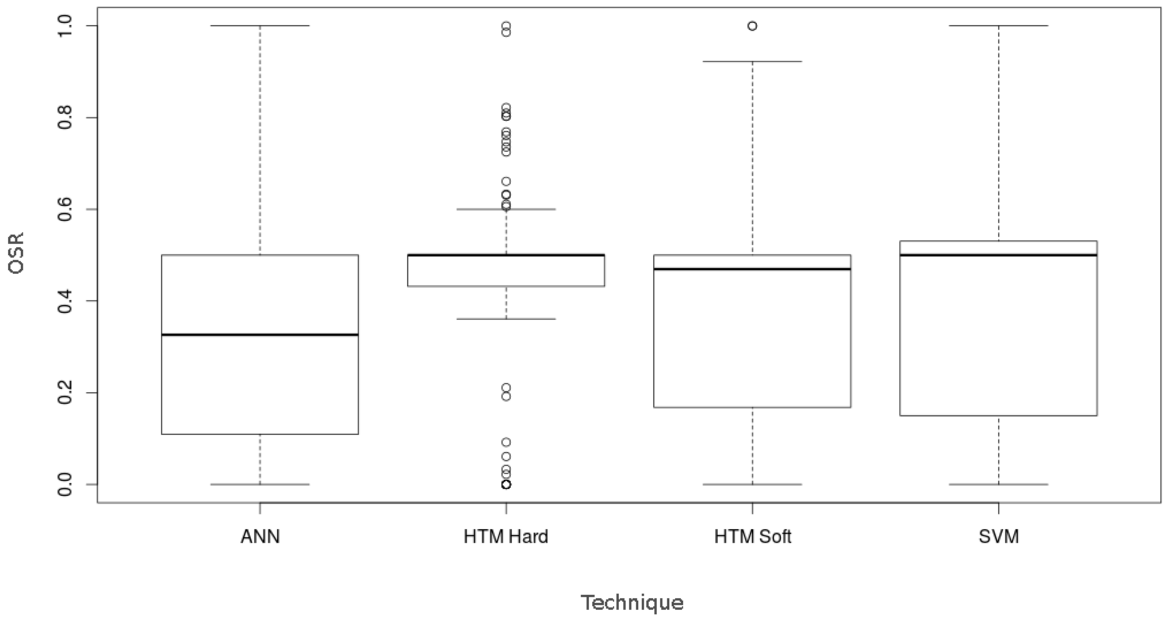


Figure 4.6 Box plot for the significant factor Technique.

Finally the most interesting aspect is the HTM Hard box, it encloses its 50% of values in the thinnest box and the smallest range from the upper quartile to the maximum point and from the lower quartile to the minimum point. This means that most of the values from this technique are around the OSR value of 0.5, this is a similar value to the others, but showing that this particular one tends to have less failures during tests. The Figure 4.6 also shows many outliers, which is normal because of the small range of quartiles. Comparing with the other techniques, these outliers represents that failures and hits are atypical; anyway it is the most stable one (because its small range).

Occlusion

Figure 4.7 shows the factor Occlusion against OSR. As a result of this plot we can point out that when there is an increment of the occlusion value (0, 25, 50, 75) the lower boundary of the box gets closer to the OSR value of 0.0, an expected issue, because if the occlusion challenge is raising its value, it becomes more difficult for a classifier to hit.

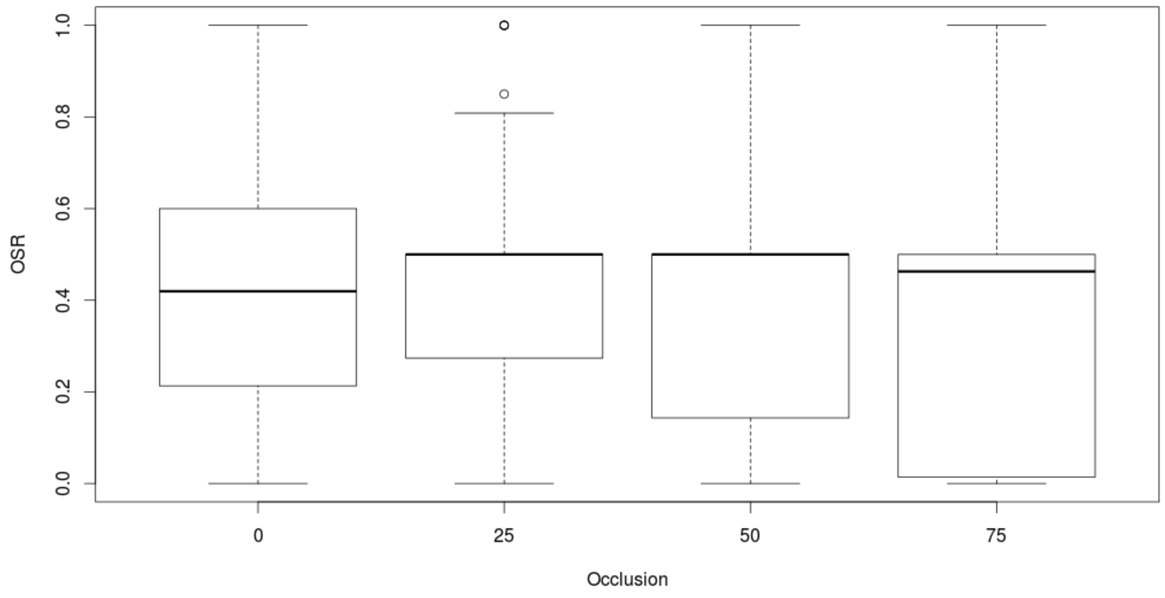


Figure 4.7 Box plot for the significant factor Occlusion.

The median for 25 and 50 have a similar value, it has sense because targets (in our case: people) are more representative from the hips to the head. Another aspect is that 25, 50 and 75 have approximately their upper box boundary near of the 0.5 OSR value, but the box of 0 starts on 0.6, showing that with 0% of occlusion there are more hits, but it has the lowest boundary and the lowest median, this issue indicates that some conditions produced this decrement (with the interaction plot analysis this decrement will be clarified).

Interaction: Technique and Occlusion

The last aspect to analyze as a result of the Table 4.2 (ANOVA table) is the interaction of the factors: Technique and Occlusion. As Anderson and Whitcomb [AW07] define: Interactions occur when the effect of one factor depends on the level of the other. The first aspect to notice is that from left to right there is a trend to decrease, which is expected due to the difficulty to classify when the occlusion is higher.

The second aspect clearly shown on Figure 4.8 is that HTM Hard and SVMs have an acceptable performance, unlike HTM Soft and ANNs. It would be obvious to think that with 0% of occlusion the techniques would have the best performance, but there is a fact: with 0% of occlusion, the targets have more freedom to move, in our videos the targets tend to move faster and all over the scene. The targets had 0% of occlusion but more flexibility and freedom of movements. This is an excellent aspect for this research, because this aspect is similar to real world behaviour.

Hence the best performance of HTM Hard and SVMs indicates that they are better to **track** objects in video sequences, with real movements of the targets. The other values for occlusion (25, 50, 75) limited the movement of the targets and the behaviour of the targets was limited to a specific zones of the scene.

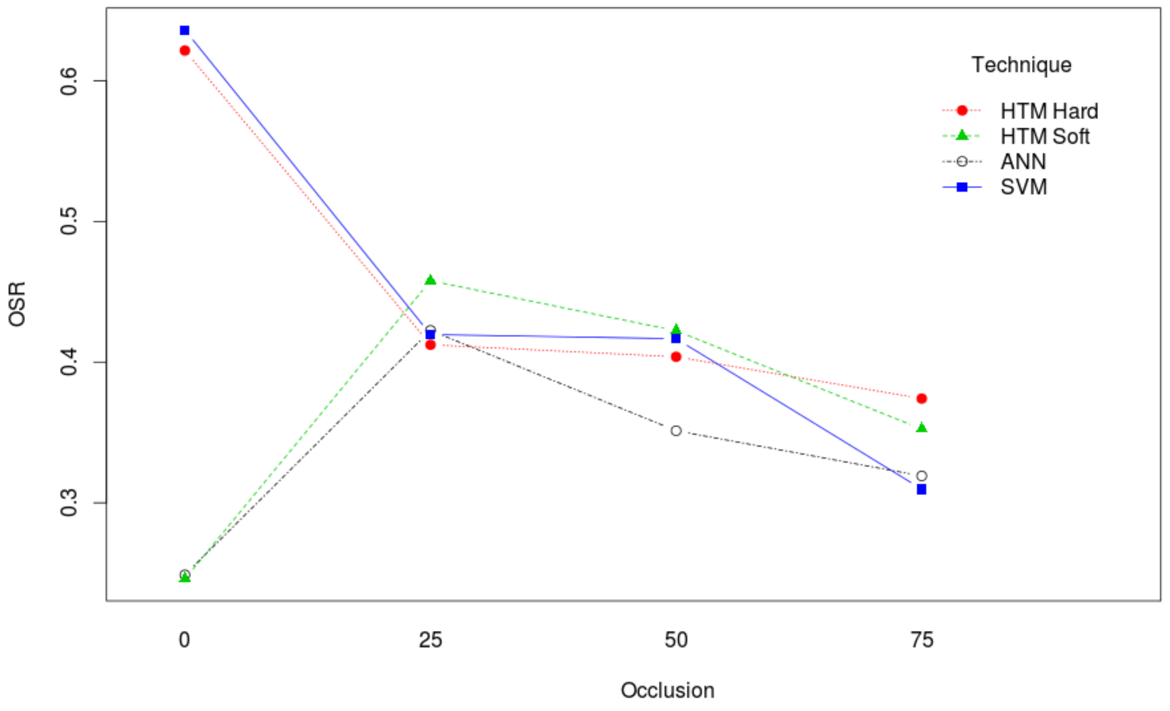


Figure 4.8 Interaction plot between the most significant factors: technique and occlusion.

If we put our focus on the HTM Soft development, leaving aside the 0% aspect, it has the best performance on the values of 25% and 50% of occlusion, and it has an acceptable performance on 75%. This is a positive result but the crash on the value 0% makes this implementation not good. Because it means that it has not the accuracy required when the target moves freely, and this is an important aspect in a video tracking issue.

Similar to HTM Soft, the ANNs implementation, has a bad performance on 0% of occlusion, thus it is not an accurate technique for moving objects. Its behaviour with 25% is acceptable (similar to SVMs and HTM Hard) but decreases in 50% and it has an acceptable one with 75%. Based on Figure 4.3d, where it shows a normal histogram distribution (from 0.0 to 0.5) and the Figure 4.8, the performance of this implementation is not accurate and the other algorithms have better results.

About the SVMs implementation, besides its remarkable performance on 0%, it shows an acceptable performance on 25% and 50% (HTM Hard as well), it is almost equal, because the most significant features of the target (people) are from the hips to the head (as mentioned in the previous Section). But surprisingly, it decreases dramatically on the 75%; the occlusion of this type produces the worst results of all classifiers.

Finally the HTM Hard shows another remarkable performance on 0% (as mentioned before) and acceptable 25% and 50% results. And the most interesting fact was its performance on 75% occlusion, where it was the best classifier. The SVMs and this algorithm are the most stable implementations.

4.3.4 Discussion

The use of Feed Forward Neural Network did not behave as expected, it was difficult for this technique to get hits from different videos, its histogram shows a pattern of random values, it got the lowest OSR in almost all occlusion challenges and it failed with 0% of occlusion, when the target moves more freely. This type of ANNs is not efficient to face this problem, another type of ANNs can be test to see its accuracy (like Recurrent Neural Networks).

The Support Vector Machines algorithms is a feasible option for this kind of problem, because it had good results with 0%, 25% and 50%; with 75% it decreased but it had some positive hits. Also it was the fastest implementation to run, despite the time was not a measured factor (it was not the interest of this research) it is important to express the experience from these experiments. Thus using SVMs to create object tracking algorithms (in the classification task) is feasible.

On the other hand we have the HTM implementations (Hard and Soft algorithms). The Soft algorithm, with 25% and 50% it got the best results and an acceptable one with 75%. But as well as ANNs, it had a bad performance on 0% of occlusion; which means that with more target unexpected movements, it could not hit; that is why this implementation is not recommended. It is important to remember that this implementation was done using the Spatial Pooler and the kNN classifier, so an interesting aspect would be the combination of the Spatial Pooler with SVMs (which got better results), or another top level classifier.

The other option of HTM was the Hard implementation. As mentioned in Section 3.2.1, this was the slowest algorithm, there are many reasons for this duration (the OPF was not conceive to process streaming data, the use of Encoders, etc.) but the important thing is that it is not feasible the implementation of this algorithm with the available hardware, it lasted 60 or 70 times more than other implementations. Besides this situation (time), it had the most stable performance; it is remarkable its development in the two more challenges aspects: with 0% of occlusion with a moving target, it had a performance similar to the SVMs implementation, and with 75% of occlusion it had the best OSR value. In general, this is the most stable implementation, it got the best results, it is not feasible the implementation of this algorithm with the available hardware, but this limitation may be negligible in a few years.

Some concerning issues of Object Tracking like the scenario or the size of the training set are not significant, for this specific problem these machine learning techniques can run with small data sets. Also the complexity of the scenario does not influence over the accuracy of the process, which is good for real world problems.

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

“A good conclusion leaves a clear statement of your point and renewed appreciation of its significance”

The Craft of Research

The main concern of this research was to use Artificial Intelligence methods to imitate our brain solving a human cognitive issue, recognize objects and track them with sight. Four different algorithms were developed, compared and analyzed.

5.1 Conclusions

This thesis has shown the development of a new algorithm using the OPF tool to implement an HTM structure to classify patterns. With the results obtained, there is statistical evidence that supports our hypothesis. Using this Object Tracking algorithm, which implements HTM as a classifier, we got a statistically significant higher OSR than using other classification techniques.

We presented this new algorithm modifying the OPF input module, and it improves the OSR results. But we got an unexpected issue: the execution time. It increases considerably than the other implementations. In a normal execution it lasted 60 or 70 times more, because of the complexity of the model in memory. That is why, with the available hardware it is not feasible its implementation to huge amount of streaming data.

In addition to this results, the implementation using SVMs had remarkable OSR results, moreover it was the fastest algorithm. It is feasible to implement this algorithm for this kind of problem. On the other hand it is not recommendable to use Feedforward ANNs or the Soft algorithm, because of their poor performance.

We also provided more evidence to support the idea of the use ofSIFT features to pattern recognition. We used the well known technique of Dense SIFT and due to the given results, it is an excellent tool for the Feature Extraction task. Furthermore, we prove that with few SIFT points, the results are satisfactory; it is not necessary to process big amount of information to get acceptable results.

We also discovered that SVMs and HTM Hard have more sensibility to targets movements, proving that these machine learning techniques are accurate with small quantity of SIFT points. This sensibility is essential for any Object Tracking algorithm, due to recognize the target in different situations or noise.

5.2 Future work

This research has led open many questions to work on. There is some future work:

1. It would be interesting to test another combination of technologies, for example: Recurrent Neural Networks, Bayesian networks, Reinforcement learning, Sparse dictionary learning, Genetic algorithms. It would be interesting comparing these results with another implementation of HTM, in this research the OPF and Spatial Pooler plus kNN were used; but there are another ways to create HTM algorithms, like the *Network* framework, where independent modules can be used, or build manually the structure. Also the Spatial Pooler can be combined with another *top* level classifier (instead of the kNN classifier), like the SVMs classifier or any other.

2. Also to work with high quality video resolution, here the 50×50 pixels resolutions was used due to the HTM Hard's implementation, but if another options are used, it is feasible to run these algorithms over high resolution videos.

3. Another issue is to use more Dense SIFT points, due to restriction in this research, the number of 12 points was imposed. But on future implementation another *magic* number can be used, for example 60, 90, 500 points; especially if it is applied over high resolution video.

4. Furthermore, to combine classifying processes with *State of the Art* tracking algorithms, for example the *Particle Filtering.* This combination can help to overcome occlusion issues and a lot more issues like: clutter, rotation, illumination, etc.

5. And another interesting issue is to use another *Feature Extraction* techniques, here the Dense SIFT process was implemented to extract interest points; but another techniques can be used to test the accuracy in the classification process.

BIBLIOGRAPHY

[AKN12] G.S Asha, K.Arun Kumar, and D. David Neels Pon Kumar. "A Real Time Video Object Tracking Using SVM". In: *Internacional Journal of Engeneering Science and Innovative Technology (IJESIT)* 1.2 (2012), pp. 302–312.

[Avi04] Shai Avidan. "Support Vector Tracking". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26.8 (2004), pp. 1064–1072. DOI: `10.1109/TPAMI.2004.53`.

[AW07] Mark Anderson and Patrick Whitcomb. *DoE Simplified.* Boca Raton, FL, USA: CRC Press, Taylor and Francis Group, 2007.

[BCW08] Wayne Booth, Gregory Colomb, and Joseph Williams. *The Craft of Research.* Chicago, CH, USA: The University of Chicago Press, Third Edition, 2008.

[Bis95] C.M. Bishop. "Neural Networks for Pattern Recognition". In: (1995).

[Blu92] A. Blum. "Neural Networks in C++". In: (1992).

[CL01] Ching-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines.* 2001. URL: `http://www.csie.ntu.edu.tw/~cjlin/libsvm/` (visited on 4/2/2015).

[Com01] Reading University Computational. *Performance Evaluation of Tracking and Surveillance. A dataset for Computer Vision and Pattern Recognition.* 2001. URL: `ftp.pets.reading.ac.uk/pub/PETS2001` (visited on 2/25/2001).

[Com09] Reading University Computational. *Performance Evaluation of Tracking and Surveillance. A dataset for Computer Vision and Pattern Recognition.* 2009. URL: `www.cvg.rdg.uk/PETS2009/a.html` (visited on 3/15/2009).

[Cos14] Allan Costa. *Nupic Classifier MNIST.* 2014. URL: `http://github.com/allanino/nupic-classifier-mnist` (visited on 8/8/2014).

[FR01] Joan Ferré and Xavier Rius. "Introducción al diseño estadístico de experimentos". In: (2001).

[FXG10] Jialue Fan, Wei Xu, and Yihong Gong. "Human Tracking Using Convolutional Neural Networks". In: *IEEE Transactions on Neural Networks* 21.10 (2010), pp. 1610–1623. DOI: 10.1109/TNN.2010.2066286.

[HAD11] Jeff Hawkins, Subutai Ahmad, and Donna Dubinsky. "Hierarchical Temporal Memory and Cortical Learning Algorithms". In: (2011).

[HB05] Jeff Hawkins and Sandra Blakeslee. *On Intelligence*. USA: St. Martin's Griffin, 2005.

[HLJ11] Bing Han, Dingyi Li, and Jia Ji. "People Detection with DSIFT Algorithm". In: (2011).

[HMZ12] Tal Hassner, Viki Mayzels, and Lihi Zelnik-Manor. "On SIFTs and their Scales". In: (2012).

[Iha98] Ross Ihaka. "R: Past and Future History". In: (1998).

[Low99] David G. Lowe. "Object recognition from local scale-invariant features." In: *International Conference on Computer Vision* (1999), pp. 1150–1157. DOI: 10.1109/ICCV.1999.790410.

[Mas+05] D.L. Massart et al. "Visual Presentation of Data by Means of Box Plots". In: (2005).

[MC11] Emilio Maggio and Andrea Cavallaro. *Video Tracking: Theory and Practice*. UK: Wiley: a John Wiley and Sons, Ltd, 2011.

[MHM07] Chris McIntosh, Ghassan Harmanrneh, and Greg Mori. "Human Limb Delineation and Joint Position Recovery Using Localized Boundary Models". In: *IEEE Workshop on Motion and Video Computing* (2007).

[MPC07] Luca Mussi, Ricardo Poli, and Stefano Cagnoni. "Object Tracking and Segmentation with a Population of Artificial Neural Networks". In: *Workshop Italiano di Vita Artificiale e Computazione Evolutiva* (2007).

[Num15] Numenta. *Numenta: nupic vision*. 2015. URL: http://github.com/numenta/nupic.vision (visited on 11/5/2015).

[Sch+10] Tom Schaul et al. "PyBrain". In: *Journal of Machine Learning Research* (2010).

[Sol12] Jan Erik Solem. *Programming Computer Vision with Python*. USA: O'Reilly Media, 2012.

[SS11] Anand Singh Jal al and Vrijendra Singh. "The State-of-the-Art in Visual Object Tracking". In: *Informatica: An International Journal of Computing and Informatics* 36.3 (2011), pp. 227–247.

[TM05] Kunio Takaya and Rishabh Malhotra. "Tracking Moving Objects in a Video Sequence by Neural Networks Trained for Motion Vectors". In: *Communications, Computers and signal Processing* (2005), pp. 153–156. DOI: 10.1109/PACRIM.2005.1517248.

[Wik14] Wikipedia. *Computer Vision – Wikipedia, The Free Encyclopedia*. 2014. URL: `http://en.wikipedia.org/wiki/Computer_vision` (visited on 5/19/2014).

[ZLN08] Li Zhang, Yuan Li, and Ramakant Nevatia. "Global Data Association for Multi-Object Tracking Using Network Flows". In: (2008).