TRACKING BY DETECTION

A DISSERTATION SUBMITTED TO THE UNIVERSITY OF MANCHESTER
FOR THE DEGREE OF MASTER OF SCIENCE
IN THE FACULTY OF SCIENCE AND ENGINEERING

2017

By
Louis Vilpoux
10079590
School of Computer Science

Contents

Number of words : 16414

List of Tables

List of Figures

Abstract

TRACKING BY DETECTION
Louis Vilpoux
10079590
A dissertation submitted to the University of Manchester
for the degree of Master of Science, 2017

The problem of tracking people is important, concerns research and new technologies. This dissertation presents an algorithm for the task of tracking by detection, its design, implementation, tests and evaluation. Detection and tracking modules have been developed with the aim of tracking without knowing in advance the object to track.

Tracking by detection is often object-oriented, meaning that a machine-learning algorithm compares the detection with data of the appearance of the object to track. Different datasets, *PETS2016* and *HIGHWAY*, containing respectively pedestrians and cars, have been used to select the best parameters to implement a tracking by detection application that adapts to the environment.

The implemented technique challenges models from the literature that use similar tracking method. This dissertation shows that results are comparable whether or not the object to track is defined. However, abrupt motion changes and weather conditions reduce the performance of the algorithm.

Declaration

No portion of the work referred to in this dissertation has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Copyright

Acknowledgements

My first thanks go to my supervisor, Dr. Tim Morris. This challenging subject has been a satisfying experience and has given to me the desire to continue to work in this area. His advice has always been a great help.

Then, I would like to thank my family for their constant support. And then, my last thanks go to my roommates, Alexis, Gregory and Tony, for our interesting discussions and our exchanges of opinions and ideas.

I would also like to acknowledge *SceneBackgroundModelling.net* and especially Yi Wang, Ph.D student, University of Sherbrooke, Canada, Martin Cousineau, University of Sherbrooke, Canada, and the staff of the CVPRLab (http://cvprlab.uniparthenope.it/), University of Naples Parthenope, Italy, for their work on the datasets that have been used in this dissertation.

Chapter 1

Introduction

This chapter introduces the dissertation. The motivation is firstly described. Then, the aims are analysed in three stages, the learning objectives, the tracking objectives and the deliverable objective. The plan of the dissertation ends this chapter.

1.1 Motivation

In the world, tracking is used to increase people safety with video surveillance, to help the machines to see the environment with autonomous navigation, to automatically annotate multimedia content with video indexing and to recognise gesture with human-computer interaction. This is only few examples of the huge possibilities of tracking by detection. Given the fact that these technologies are young, they just recently start to enter in our life but they will be present every where.

Currently, the research is mainly made for the topic of autonomous car driving and the technologies are more powerful and accurate every day.

Tracking by detection needs to deal with issues from different sources. The first one is the object itself. It can appear and reappear in the video frame, and its appearance can change at each moment. For the people, they can be partially or fully occluded without time limits. The second source is the background. As described in [18], changing backgrounds complicate data association across multiple frames because it can be a barrier for the modelling of the background, for example. Then, the general conditions of the video, such as illumination changes, cluttered scenes or noise, are some circumstances to consider. Finally, the detection algorithms, which estimate object location at every frame, have drawbacks. They require offline training stage, meaning that they are time and memory consuming. And the result of an object

detector, used by a tracker, is not full of true positive elements. There are some missing or wrong detections.

1.2 Objectives of the project

The subject of the dissertation is to select and implement a method from the literature to track pedestrians in a mall and vehicles in a junction, without previously building a model of the object to track. Three methods have been proposed. As a result of that, an area of work can be presented. First, the methods should be compared in order to decide which one is the most appropriate to our subject. Second, another key point is the fact that the tracking has to concern pedestrians as well as cars while they are totally different. Third, the lack of model is underlined and it emphasises the fact that the tracking method used should be able to deal with diverse moving objects in different environments. This notion of environments is also significant because computer vision algorithms have to ignore some conditions that will not be ideal. Lastly, an implementation of the chosen method will be done with a comparison of the performance between the literature and the implemented software and an overview of the strength and weakness of the project will be presented.

1.2.1 Learning objectives

The objectives of the dissertation are to read scientific papers related to tracking by detection, understand the key points and compare the methods. The background part and the implementation part represent challenges because I have never studied or done some tracking or detection on people or cars in videos. I am also a novice in the implementation technologies related to computer vision.

1.2.2 Tracking objectives

In this dissertation, some tracking objectives have been extracted. Firstly, videos, in which cars and pedestrians will be tracked, with ground truth have to be collected. Secondly, the algorithm has to deal with objects of interest on the inside or on the outside. The targets have no special condition, they are free of their moves, the algorithm should solve some problems explain above. Thirdly, the objects have to be annotated. The output will be a labelled video to illustrate the detection and the tracking.

The project does not mention any constraints with the recording camera. As a result of that, the best situation will be chosen with a stabilised camera. A pre-processing of the data could be required, for example a noise reduction.

1.2.3 Deliverable objective

The main parts delivered are :

- The design and the implementation of the algorithm of tracking by detection. The source code has, as input, a video and outputs a video annotated with the people or cars, identified and tracked.
- The tests and measures of the software. Accuracy and performance of the program are compared to the results from the chosen method.
- The MSc dissertation with the background that is the basement of the project, the design, implementation and evaluation of the preferred approach.

1.3 Plan

The structure of the dissertation is detailed below :

- Chapter 2 - Background : overview of the topics of the project with a literature review about the tracking and detection technologies, and the methods proposed. The comparison of the methods is concluded with a choice of a method for the tracking by detection.
- Chapter 3 - The detector confidence particle filter : deep explanation of the algorithm particle filter and its main parts.
- Chapter 4 - Research methodology : explanation of the pipeline of tasks. It starts with the collection and the pre-processing of the data, then the design of the algorithm and its implementation with all the components. Tests and evaluation are the final steps.
- Chapter 5 - Implementation : the technical aspects about the implementation are explained in order to build a performing software. The analysis of the code is made by considering modules.

- Chapter 6 - Experiments : the research of the optimal parameters that come along the code, the results and the performance are analysed and compared with the literature.
- Chapter 7 - Discussion : the strengths and the weakness of the implemented algorithm are detailed.
- Chapter 8 - Conclusion : sum up of the results of the selected technique and its used, limitations and future works.

Chapter 2

Background

This chapter overviews the tracking and detection technologies. It also presents three methods, taken from the literature, that have been proposed in order to track by detection pedestrians and cars in video scenes. The analysis and the comparison of them are the ways of selecting the best method.

2.1 Tracking technologies

Tracking in computer vision is the fact of following an object or a person in a video scene even if the motion, the occlusion, the light and the position change.

Different methods have been created to compute tracking. As mentioned in [13], tracking methods are divided into three categories. They are illustrated in the following figure and explained below.

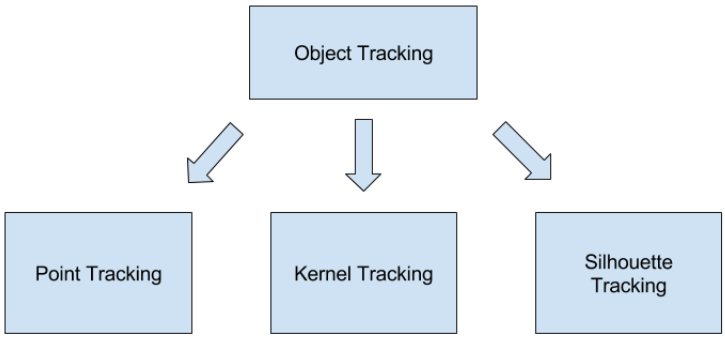

Figure 2.1: The general tracking methods

- Point tracking. Points represent objects to be detected, and the previous object state is used to find the association of points. Kalman filter, particle filter and multiple hypothesis tracking are point tracking methods.

- Kernel tracking. A kernel, a mixture of simple geometric shapes, is used to track objects by computing motions, represented by parametric transformations, such as affine, translation and rotation, in the frames. Simple template matching, mean shift method, support vector machine and layering based tracking are kernel tracking methods.

- Silhouette tracking. Because some objects are more sophisticated and cannot be approximated with the previous method, the object region is estimated with information from the appearance density and the shape models in the frames. Contour tracking and shape matching are silhouette tracking methods.

It is important to notice that, in point tracking, detection is not only done when the object or the person appears for the first time in the video, like in the kernel and silhouette tracking, but in every frame. Given this fact, it is difficult to compare these methods and conclude on the best one. But, [21] draws some conclusions. A point tracking method, particle filter, has the best results with a decent processing time per frame, kernel tracking is limited by parameters such as the shape of the object and the background, and silhouette tracking is more efficient in order to handle complex object shapes.

2.2 Detection technologies

The detection verifies and locates an object or a person in an image or in a video frame. Different methods exist and some are briefly presented below.

2.2.1 Single scale detectors

Single scale detectors use interest points to perform the detection. In the case of the Harris corner detector, corners are preferred instead of edge. Indeed, ”corners provide repeatable points for matching” [8], better than edges, defined as physical events : edge of object, change in colour or in surface orientation. Then, as mentioned in this kind of method, only one scale is considered because basic information from images are used.

In this algorithm, a window is shifted in directions. The aim is to detect corners by computing intensity change. As a result of the method, corners, edges and flat regions are highlighted.

2.2.2 Multi-scale detectors

Multi-scale detectors contribute to solve the main problem of the previous subsection : scale. Interest points still are compared but from a descriptor of a region that is scale invariant. In the difference of Gaussian algorithm, DOG, the maximum of a subtraction in the scale space is selected and points with low contrast are deleted [8].

This algorithm is based on two images, which have been transformed with Gaussian kernels. Shortly, the Gaussian kernel smoothed the images. Thanks to the subtraction, spatial information is saved. The second derivatives of Gaussian, one way to computed edges, is not needed because the algorithm naturally performs a well approximation of it. Hence, edges are more visible and a maximum in scale has been found.

2.2.3 Image feature descriptor

The Image feature descriptor is a group of algorithms that describes part of images. Each region of an image is analysed and a descriptor, a result that characterised the region, is obtained. Then, descriptors are processed by the user, depending on what he wants.

The histogram of Gaussian, HOG, is a feature descriptor and its algorithm is explained in section *3.4.3*.

2.3 Explanation of the different methods

This section explains three algorithms that facilitates tracking by detection. They are attached with the subject of the dissertation.

2.3.1 Tracking-Learning-Detection

The aim of this method is to find the object location, or to indicate that it is not visible, and to track it. As explained in [35], the algorithm is decomposed in three main tasks.

- In the tracking part, the object is followed from frame to frame. Given the assumption that the object is visible, the object motion is estimated. This step is fast because only the initialisation is required. However, some errors appear and the disappearance of the object will result in the program failing.

- The detection part performs a scan of the video frames and estimates the object position. It also localises all the different appearances that have been observed in the past frames. Some particularities of this step are that the detector, contrary to the tracker, do not fail in the case of the object disappears and cannot be used for objects that will not already be considering.

- The learning part calculates the detector error. It corrects the detector by updating it to improve the results of the algorithm. This step is done by *P-N learning*, with P-expert that estimate the false negative elements and N-expert that estimate the false positive elements of the detector.

In order to not use a model, an assumption is the base of the object detection: the background is more frequent than the object. As a result of that, a large number of patches is analysed by a classifier. The object, described by a target template [35], an image patch, has its motion defined by transformation. These templates have limited modelling capabilities and will not be able to precisely model the object in case of complex background.

2.3.2 Monocular 3D Pose Estimation

The aim of this method is to find the 3D pose of each person in all frames. Three stages are defined in [18]:

- The first stage computes an initial estimation of the 2D articulation of the person in all the frames, independently. Detector parts also produce the viewpoint of the person. The 2D position of body parts are measured in single frames.

- The second stage performs a data association [18]. The estimations from the previous stages are linked. A weak detection has been transformed into a robust estimation of people. Short image sequences, also called tracklets, gather the 2D positions.

- The last stage is the outcome of the algorithm. The image observations, containing the evidences, recover the 3D pose estimation. 2D tracklets are used for 3D tracking.

The article [18] describes as well the use of a classifier, Adaboost. A calculation of the probability of dependencies between body parts is done. For example, if the head is firstly considered, the probability of having an arm near the head will be computed. And vice-versa. As a result of that, these likelihoods will estimate the body parts. The following image is a representation of the 3D pose estimation of a person.

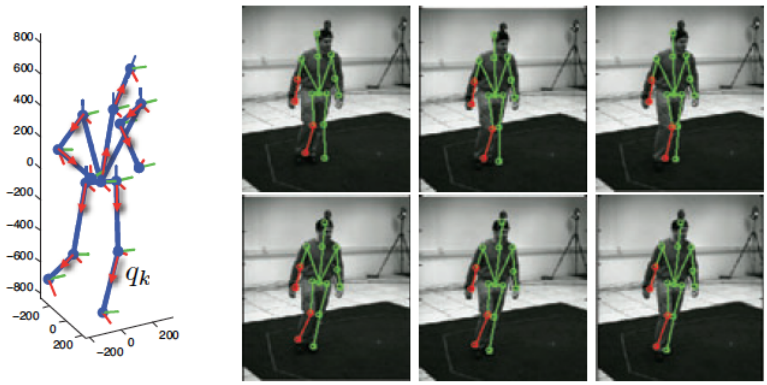

Figure 2.2: Example of 3D pose estimation. [18]
3D pose estimation of a model (left), initialised on a sequence (right)

2.3.3 Detector Confidence Particle Filter

The aim of this method is to automatically detect and track objects or people in video scenes with a particle filter framework. [1] described that the particle filter uses two different models. A dynamic model is responsible of the prediction and the observation model evaluates the probability of a predicted state. As explained above, the object detection is used in every frames by the observation model. The algorithm first assigns a particle, to each person or object detected in the observation model, and then assesses the links between the detection and the particles in each frame. This is the data association. Then, the importance weight of each particle is computed. As explained in [1], a bootstrap filter is used as an importance distribution to approximate the probability density function. Then, the particles are resampled and propagated in the next frame to create a robust tracking.

The detection model uses a well-known detection method, the histogram of gradient. It outputs the people present in the video. The data association uses this result to allocate a particle to each detection. Then, the common particle filter algorithm is applied with the estimation of the conditional likelihood, that represents the importance weight of a particle, the resampling and the propagation. As a result of that, people will be tracked by an amount of points in the video scene.

2.4 Comparison of the results of the different methods

The three methods from the literature all use different datasets. In order to analyse the different results of the methods, it was necessary to consider their results only on datasets that show similar situations. In our case, pedestrians crossing streets or just walking is the common point between all the experiments from the literature. The criteria for selecting the best method are the possibility to track people, the possibility to track vehicles, the possibility to track multiple objects and the accuracy-precision-performance of the method, as high as possible.

Some words related to the accuracy of algorithms have to be defined. Let us consider some elements that are true or false. After a classification test, some of them are classified as positive and others as negative.

- True positive : given the true elements, the elements that are estimated positive.
- False negative : given the true elements, the elements that are estimated negative.
- False positive : given the false elements, the elements that are estimated negative.
- True negative : given the false elements, the elements that are estimated negative.
- Recall : this is the proportion of true positive over ”the total relevant instances” [33]. The abbreviation of recall is R.
- Accuracy : how close a measured value is to the actual (true) value. This is the number of the correctly classified elements divided by the total number of elements.
- Precision : how close the measured values are to each other. The abbreviation of precision is P. This is the number of true positive elements divided by the total of classified positive elements.

- F-measure : this is the "harmonic mean of precision and recall" [33]. The mathematical formula is :

$$F = 2 * \frac{P * R}{P + R} \tag{2.1}$$

In the article related to particle filtering, the results from the dataset ETHZ Central have been considered. This dataset deals with people crossing a street. The precision is 70%, the accuracy is 72.9%, the false negative rate is 26.8% and the false positive rate is 0.3%.

In the article related to the tracking-learning-detection algorithm, a dataset used in the literature has been selected, the PROST dataset. The average recall is 96.7%, the average localization error is 12.03%. In a different dataset, the TLD dataset dealing with people, the precision is 99%, the recall is 51.8% and the f-measure is 62%.

In the article [18], the results of the HumanEva II dataset have been considered. It deals with people walking in a scene. The mean error of the algorithm is 10.72 pixels, but there are no quantitative results.

2.5 Conclusion about the choice of the method

From the definitions of the accuracy, the precision, and given the results of each methods, tracking-learning-detection is not interesting because of the f-measure (68% in [1] greater than 62% in [35]).

Another argument for not studying this method is the fact that this method only tracks a single object. Given the ambition of this project, a powerful method is desired.

Because the 3D Pose Estimation is most human oriented, the chosen method is particle filter, also called condensation.

Starting with a general overview of tracking and detection, a method based on the particle filter algorithm has been selected. In the next chapter, a focus is made on the literature related to this algorithm.

Chapter 3

The detector confidence particle filter

This chapter deeply studies the method that have been selected from the literature, in the previous part. The main parts of the algorithm are detailed and also its origin.

3.1 Context

The particle filter methods, or sequential Monte Carlo (SMC) methods are Monte Carlo methodologies to solve filtering problems.

3.1.1 Monte-Carlo methodologies

A Monte Carlo method is a statistical approach in data simulation [19]. A simulation is a process that uses sequences of random numbers as data.

The main idea of these algorithms is the fact that the repetition of random sampling is used to solve problems [30]. The Monte Carlo methods can be described with similar steps. First, a likelihood distribution generates random inputs. Then, a deterministic algorithm is applied on the data. Finally, the results are combined.

These procedures are used in diverse domains, such as the climate change, the biology in order to study genomes and proteins, in finance in order to estimate the risks that will arise from decisions, and also in computer vision.

3.1.2 Filtering problems

The main idea of the filtering problems is, given observations with noisy or missing data, to find the best estimate of the model that represents the problem. As explained in

[26], the solution of the filtering problems is infinite-dimensional. In order to be implemented, the approximation of the solution has to be finite-dimensional. For example, the Kalman filter is one of the estimation in finite dimension of the linear filter.

To be more precise with the definition of a filtering problem, the current observation is used to estimate the future state variable density. The representation of this phenomenon is a hidden Markov model.

3.1.2.1 Markov model

A Markov model is a stochastic model that has the property that the current state only depends on the preceding states. Different Markov models are known. Four models are possible and the choice depends on the system and its observation. Given the subject of this dissertation, autonomous systems are appropriate. Depending on the fact that every state of the system is observable or partially observable, two classes exists.

In the first case, the model is called a Markov chain. The distribution of a random variable is calculated with the distribution of the previous state. In the following figures, the state variable is X and the observation variable is Y.

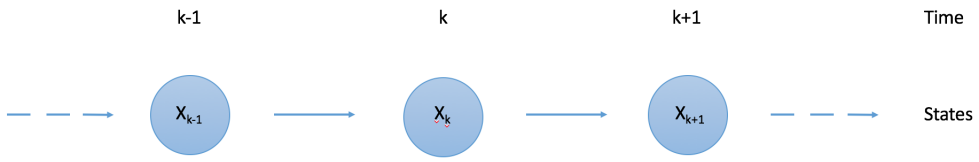

Figure 3.1: Example of a Markov chain

In the second case, the model is called an hidden Markov model, HMM. It is kind of similar to a Markov chain but the particularity is that the state is not directly observable.

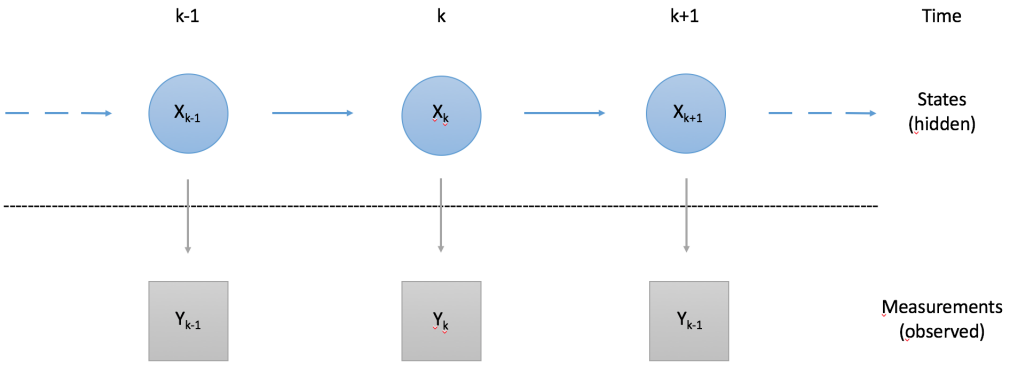

Figure 3.2: Example of a hidden Markov model

The measurements correspond to a state of the system but they do not permit to truly determine it. Nevertheless, the output, that depends on the state, is visible.

3.1.3 Kalman filter

The Kalman filter is an algorithm that produces an ”optimal estimator” of data that have been measured with noise [17]. From what has been explained so far, it is linear because it best performs on Gaussian noise. It minimises the mean square error of the approximated parameters.

The Kalman filter algorithm has two steps, a prediction step and an updating step.

- Prediction step : the algorithm produces an approximation of the state variables at time k
- Updating step : with the measurements at time $k+1$, the estimations are updated. A high weight is the result of a high confidence in the optimal estimator.

A major advantage of this algorithm is that it is recursive. It only needs the current measure and the previous state to be able to estimate the next state, called the state transition model. It can also run in real time. The figure *3.3*, below, translates the Kalman filter processing of the data.

Before presenting the particle filter algorithm, it can be important for the reader to have an overview of a notion that will be discuss later in the case of our method. This is the relation to the recursive Bayes estimation [28]. The Kalman filter uses measurements and a mathematical model to predict correct states at each time. In the

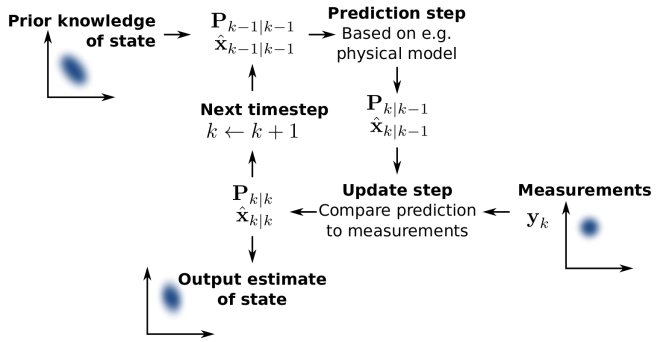

Figure 3.3: Concept of Kalman filtering

case of a recursive Bayesian estimation, a probability density function is estimated in a recursive manner at each time, thanks to measurements and a mathematical model. As a result of that, a dynamic Bayesian network represents the method. And then, in the case of a recursive Bayesian estimation, ”the true state is assumed to be an unobserved Markov process, and the measurements are the observed states of a hidden Markov model (HMM)” [28].

To sum up, the Kalman filter algorithm uses the fact that Gaussian densities are omnipresent and it has easy update steps. But, it only deals with uni-modal distribution and, because of its linearity, in practical, the group of motions, made by an object in video scenes, is limited [9].

3.2 Particle filter (condensation) algorithm

The particle filter algorithm has been introduced in [14] and this is the reference document.

But first, it is important to link what has been explained so far to this section. The particle filter methods, or sequential Monte Carlo methods, are used to fix the hidden Markov model. Unlike the Kalman filter, this algorithm solves non-linear filtering problems, ”in a system that has non-Gaussian noise, the Kalman filter is the optimal linear filter, but again the particle filter may perform better” [23].

3.2.1 Principle

The algorithm uses the principle of prediction-correction, similar to the prediction-update, explained in section *3.1.2*. Each sample of a distribution is made of particles and each particle has a weight. This weight describes the probability of a particle of "being sampled from the probability density function" [32]. Because of its belonging to the Monte Carlo methods, the algorithm of particle filter uses the process of factored sampling, at each iteration. The resampling step is responsible for removing low weight particles by high weight particles from the same sample.

3.2.2 Algorithm

The original algorithm of particle filter has three steps : a selection step, a prediction step and a measurement step. It is iterative. Some initial and terminal conditions should be decided but, in this section, they will not be part of our attention. The next section, responsible of the creation of a whole algorithm, explains these points. The objective of this subsection is to give to the reader a well knowledge about the functioning of the particle filter algorithm by presenting only one time-step in this algorithm of the probabilistic propagation of a sample-set.

The input of this algorithm is a sample-set ($s^{(n)}$ with n from 1 to N) with weights (π) at time *k-1* and the output is a sample-set with weights at time k with the same number of elements, N.

The figure *3.4* below helps to understand the particle filter algorithm.

3.2.2.1 Selection

The selection refers to the factored sampling. Given a set of samples and each sample having a weight, the idea is to randomly build a new sample-set with the most likely samples.

In order to better visualise the process, a subdivision is created where the length of the subdivision is the value of the weight. As a result of that, with normalised weights, the section length is one (a section is the concatenation of all the subdivisions).

A random number between zero and one is chosen. The sample that has its subdivision that contains this number is picked. A new set is constructed by repeating the operation with replacement. Consequently, a set can have the same samples several times and this proceeding contributes to naturally removes the low weighted samples. The word *drift* often described this stage.

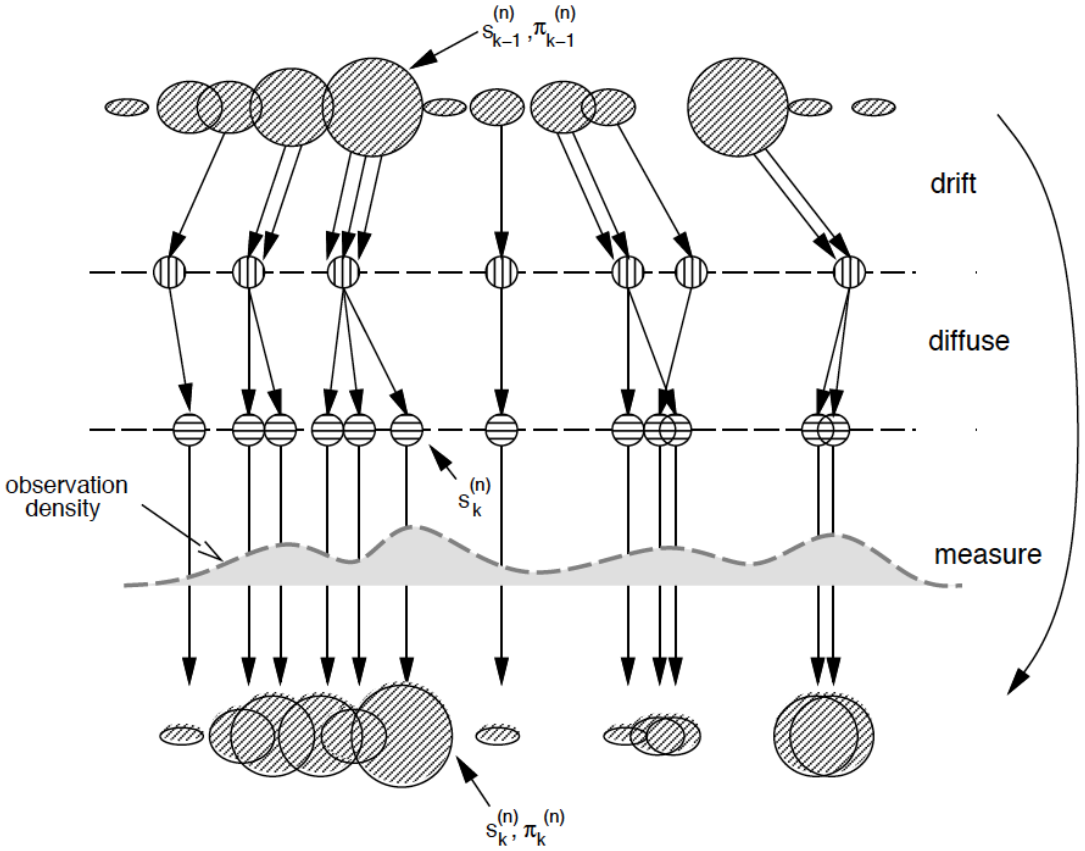

Figure 3.4: One time-step in the particle filter algorithm

3.2.2.2 Prediction

The prediction stage calculates the possible position of the new sample-set. A differential equation is used to update the samples with some noise. The word *diffuse* often described this stage.

Therefore, the set of the predicted states, given its previous values, is determined. This can be understood as the potential position of a point in a 1-dimensional space. The weights are not yet computed.

3.2.2.3 Measurement

This final step uses two sources of data : the predicted positions of the samples and measured features. The aim is to obtain the new weights of the samples.

The likelihood of a state, given its measure, is used to obtain its weight. Indeed, the measures correct the predictions. This stage is often described by the word *measure* .

Back to the 1-dimensional space example, given all the possible positions of a point, the measured feature is used to attribute high weights to well predicted states and conversely, low weights to mis-predicted states. And this step should let the *selection* stage maintains the well predicted points and removes the wrong predictions.

To sum up, the particle filter algorithm deals with random densities and different state spaces. But, because it is dependent to the number of particles, it can become computationally expensive.

3.3 Algorithms and distribution

In this section, some algorithms and a distribution that are needed to well understand the rest of the dissertation are presented : the support vector machine, boosting, the mean shift algorithm, the Gaussian mixture model and the Normal distribution.

3.3.1 Normal distribution

The Normal distribution, also called the Gaussian distribution, is a "continuous probability distribution" [31] that is used in statistics but as well in computer vision, in our case. The probability density is :

$$f(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{3.1}$$

The parameters μ and σ^2 are respectively called the mean and the variance of the distribution.

3.3.2 Mixtures of Gaussians

The Gaussian mixture model, GMM, is a combination of different Gaussian distributions. They are linearly superposed. If the concept seems easy to understand, its resolve is more complex. The following graph presents the model.

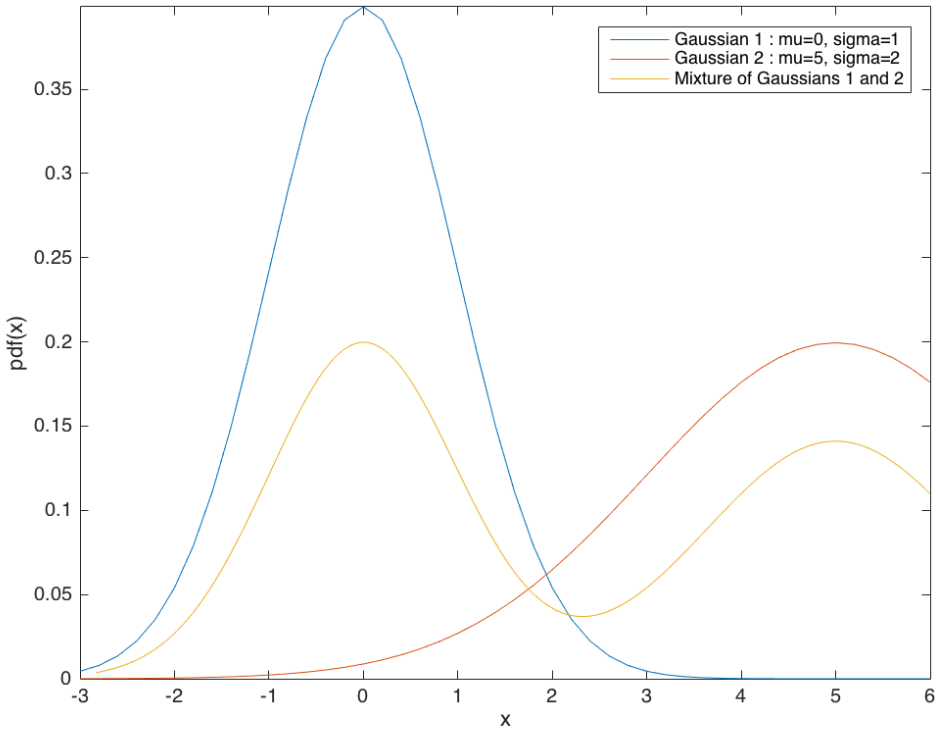

Figure 3.5: Example of a Gaussian Mixture Model
pdf(x) : the probability density function of x

The equation of a GMM is defined by its probability density [29]:

$$p(\mathbf{x}) = \sum_{k=1}^{K} \alpha_k p_k(\mathbf{x}|\mu_k \theta_k) \tag{3.2}$$

The different parameters are :

- K : this is the number of components.
- θ_k : this is the covariance matrix of the k component.
- μ_k : this is the mean of the k component.
- α_k : this is the mixture weight of the k component. In other words, the likelihood of selected $\mathbf{x}$ by the component k.

The aim is to find θ, μ and α.

3.3.3 Mean shift algorithm

The Mean shift algorithm is a clustering method that does not require any mathematical model. Given a distribution, its aim is to find ”local maximum density” or ”modes” [7].

After the initialisation of a window size, the centre of gravity of the window is evaluated and the window is shifted to the mean. Then, this process is repeated until convergence, meaning until the window stays at the same position.

This general model-free algorithm only relies on the window size but the fact that the output also depends on it can be computationally expensive.

3.3.4 Boosting

The boosting is a group of algorithms and the most well-known algorithm is *Adaboost*. The concept is going to be briefly explained.

Boosted algorithms train ”models sequentially, with a new model trained at each round” [2]. After each step, the examples that have been misclassified are used in the following model and this new model is then trained.

The aim is to consider that the mistakes of some models can be balanced by new models.

3.3.5 Support vector machine

The support vector machine (SVM) is a learning model used to cluster data. Indeed, in its linear form, it finds ”the optimal linear decision boundary” [2], meaning the boundary in the distribution with maximum margins. However, the SVM is also used to solve non-linear problems. In this case, as it can be understood with the name of the method, the kernel trick, it requires kernels. They are applied to represent input data in an higher dimension in which a linear classification is the solution. Popular kernels are polynomial and radial basis function, better known under the name Gaussian.

The model depends on a parameter C, the slack variable penalty. A huge value of C means ”a strict SVM solution” [2] while a small value permits the data to be misclassified, with a soft margin. The following figure represents the impact of C on the boundary.

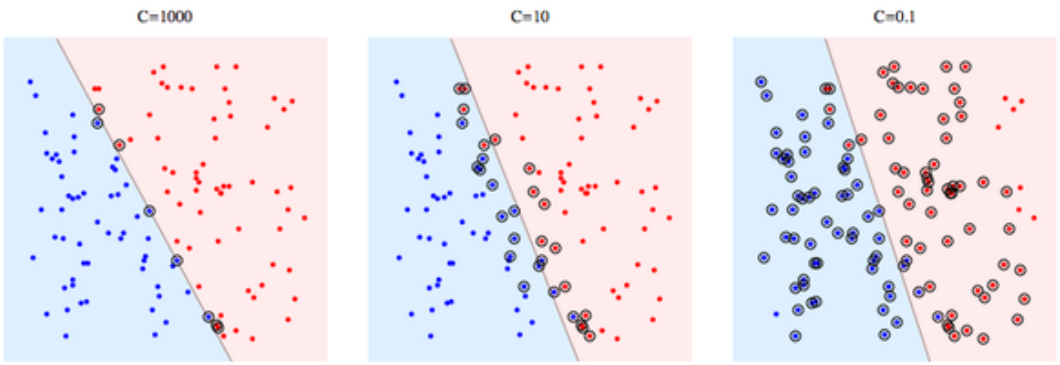

Figure 3.6: The effects of C on the decision boundary

3.4 Detector confidence particle filter

The particle filter algorithm, also called condensation algorithm, is used to process a detection and tracking of one or multiple objects in a scene with a camera using a particle filtering framework. From this situation comes the name tracking by detection. The scene can be simple or much sophisticated with lighting variation, occlusion or not constant movements from the objects of interest.

3.4.1 Vocabulary

In this subsection, some notions are described to help the reader to understand the next parts.

- Tracker : a tracker is a sample-set, it contains an amount of particles. This is the system that follows the persons in a video. This is the abstract element of the algorithm.
- Target : a target is the object of interest in the video. In this dissertation, pedestrians and cars are the targets. The target is also called the object of interest (or just the object).
- Detection : a detection is the result of the detector. It should concern people and cars. In this dissertation, a detection is represented by a shape with a centre, the detection centre.
- Particle : a particle is described as a point with a weight. It can move in the

frame in all the directions. For example, a tracked person is followed by the representation of a particle in the video.

3.4.2 Context

The algorithm presented in [14] has many common points with the one in [1]. Here, the automation has been studied one step further by replacing the measured features with a detection stage.

In the case of the method developed in [1], the distribution of a target state is estimated with a particle filter. A particle filter algorithm is made of two models. A dynamic model is responsible of the prediction of a state and the observation model is responsible of the evaluation of the predicted state.

Before the explanation of each stage of the algorithm, the general case should be explained with an example. Given a video scene of people walking in a street, each frame of the video is computed. The persons are detected with the detector and a particle filter is initialised for each person detected. A level of confidence describes these detections because tracking only happens on high-confidence detection. In the next frame, the particles that describe the persons are propagated and the new detections correct the predictions. An association between a tracker and a detection is made. In the case of a well reliability of the tracker-detection association, the detection is considered as the best guide for the particles of the tracker. Finally, an observation likelihood function is computed, for both each particle filter and the associated detection, to obtain the weights of each particle. This process continues in an iterative way with the resampling.

The fact of using a detector involves some missing detection or some detection not made from persons. Consequently, a first-order Markov model is used, meaning that the data of the current and the last time step are needed.

3.4.3 Histogram of gradient detection

The detection part of the algorithm uses the histogram of gradient detection method (HOG). The feature descriptor is applied at each frame on every part of the image.

3.4.3.1 Concept

A feature that can be considered as global is used to describe the object of interest instead of a set of local features.

As explained in the original paper [3], the context is that "local object appearance and shape can often be characterised rather well by the distribution of local intensity gradients or edge directions". More specifically, after a cutting of the image, a window is moved on it and the histogram of gradient describes every part. Then, a classifier is used to conclude on the object of interest or not. In this case, a SVM is the classifier.

This is a multi-scale detector because different sizes of the image are used and analysed.

3.4.3.2 Algorithm

The algorithm is divided in six parts after a pre-processing of the input image.

- Preparation of the image : the notion of block and cell is used there. A 8*8 pixels is a cell and cells are contained inside a block. Usually blocks overlap each other, so that the same cell may be in several blocks.

- Normalise gamma and colour : this is a minor step but the aim is to increase performance in the research of the object.

- Compute gradients : for each pixel of the cells, the horizontal and vertical gradients are computed. The horizontal gradient is defined as :

$$G_x(x,y) = I(x+1,y) - I(x-1,y) \tag{3.3}$$

  The vertical gradient is defined as :

$$G_y(x,y) = I(x,y+1) - I(x,y-1) \tag{3.4}$$

  where *I(x,y)* is the intensity of the pixel located at the coordinates x and y.

- Weighted vote into spatial and orientation cells : a weighted vote for a class of the orientation-based histogram, based on the values found in the gradient computation, is computed for each pixel within the cell. The histogram channels are uniformly spaced over 0 to 180 degrees or 0 to 360 degrees, depending on whether the gradient is unsigned in the first case or signed in the second case. For each cell, 64 (8*8) gradient vectors are obtained. It has been found, in [3], that unsigned gradients used with 9-bins histogram performs best in their experiments. The pixel contribution in the weighted vote is obtained with the

gradient magnitude. The magnitude is defined as:

$$G(x,y) = \sqrt{G_x(x,y)^2 + G_y(x,y)^2} \tag{3.5}$$

The phase of the gradient is defined as :

$$\Theta(x,y) = arctan(\frac{G_y(x,y)}{G_x(x,y)}) \tag{3.6}$$

The HOG can be created for each cell, given the fact that each gradient vector has a contribution for the histogram, by the magnitude of the vector. Consequently, high gradients will be more representative of the histogram.

- Contrast normalised over overlapping spatial blocks : in order to deal with images with uniform contrast, a normalisation is made. Concretely, the method is "based on grouping cells into larger spatial blocks and contrast normalising each block separately" [3]. Considering v as the not normalised vector of the histogram within a block and ε as a small constant, L1-*norm* or L2-*norm* are used.

$$L1\text{-}norm, v \rightarrow \frac{v}{\sqrt{\|v\|_2^2 + \varepsilon^2}} \tag{3.7}$$

$$L2\text{-}norm, v \rightarrow \sqrt{\frac{v}{\|v\|_1 + \varepsilon}} \tag{3.8}$$

- Collect HOGs over detection window : for each detector window, a descriptor is attributed. It is represented by the histograms of all the blocks within the detector window and the descriptor is the final information : "the final descriptor is then the vector of all components of the normalised cell responses from all of the blocks in the detection window" [3].

- Linear SVM : the descriptors are used by a linear SVM considering as soft because of $C = 0.01$ [3].

The figure *3.7* represents the entire algorithm, in the case of the classification of a person.

To sum up, the HOG descriptor is not really presented in [1] but its main advantages are that it "captures edge or gradient structure that is very characteristic of local shape" [3] and it is invariant in rotation, translation and scale.

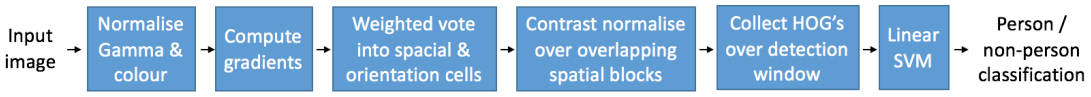

Figure 3.7: The different steps of the HOG algorithm [3]

3.4.4 Particle filtering

This subsection reminds the condensation algorithm explained in section *3.2*. However, a special attention is given to the mathematical formulas and to the details of the different steps. Some examples are also presented to justify the choices and to clarify some points. Besides, it is valid for the other subsections.

3.4.4.1 Bootstrap filter

The aim of the particle filter algorithm is to approximate the distribution of the state of the targets. Before explaining this point, it is important to defined some mathematical variables related to the particle filter algorithm.

The state of a target is $\boldsymbol{x} = \{x,\ y,\ u,\ v\}$ with *(x,y)* the position of the target in the image in 2-dimension and *(u,v)* the velocity of the target in the same space.

The weight of a particle i at the time t is specified by :

$$\omega_t^{(i)} \propto \omega_{t-1}^{(i)} \cdot p(y_t|\mathbf{x}_t^{(i)}) \tag{3.9}$$

3.4.4.2 Resampling

The resampling part is similar to the selection part of the particle filter algorithm, in section *3.2.2.1*.

In our case, the resampling stage is computed at each time step of the algorithm. The number of particles is kept over the time as N.

3.4.4.3 Propagation

The propagation part corresponds to the prediction part in the original *condensation* algorithm, section *3.2.2.2*. The propagation of all the particles is made given a "constant velocity motion model" [1]. It means that, for each time t, the position and the velocity of the target changes. This model is described by the following equations :

$$(x,y)_t = (x,y)_{t-1} + (u,v)_{t-1} \cdot \Delta t + \varepsilon_{(x,y)} \tag{3.10}$$

$$(u,v)_t = (u,v)_{t-1} + \varepsilon_{(u,v)} \tag{3.11}$$

Δt is a difference of time, $\varepsilon_{(x,y)}$ and $\varepsilon_{(u,v)}$ are some noises. They are obtained from a random value of a Normal distribution, with zero mean but they have different variances. In the first case, the noise of the position is proportional to the size of the target. In the second case, the noise of the velocity corresponds to the inverse of the "number of successfully tracked frames" [1]. The idea is that the number of particles decreases when the time of successfully tracking increases.

In order to obtain the size of the target, the position of the target is first estimated. The mean shift algorithm is used because an amount of particles described a tracking target. Then, its size is the result of the "average of the last four associated detections" [1].

3.4.5 Initialisation and termination of the tracker

One of the major asset of this algorithm is its automation and this is the case for the appearance of trackers and their deletion.

A tracker is created in the case where an object has been detected and this object is not associated to another tracker. From the fact that pedestrians cannot appear in the centre of the video frame, but only from the sides, a constraint in position is created. Hence, trackers are established along the image border. The position of the particles of the tracker follows a Normal distribution centred at the detection centre. The number of particles is a parameter of the algorithm and the spreading uses the size of the detection. And then, the direction of the tracker is set to be "orthogonal to the closest image border" [1].

A tracker is finished after not being associated in a certain number of frames.

3.4.6 Data association

The data association part of this algorithm creates the bridge between the tracking and the detection, to obtain tracking by detection. It has been mentioned in the previous subsections and it is now clarified.

The idea is that the pair of tracker and detection is searched such that at most one detection should head at most one target through the frames. Each pair is evaluated and the maximum score of the matrix of the matching score of the pairs is preferred. In the case that the best pair has a score greater than a threshold, chosen by the user, the detection will have the major influence on the tracker.

In the following subsections, the tracker is *tr* and the detection is *d*.

3.4.6.1 Matching score

As presented above, the score of the tracker-detection is needed and the search of the highest value has to be done.

The distance between the detection and the tracker is the first element that can be decisive in the association. The function *s(tr,d)*, called "matching function" [1], computes it and is given by the following function :

$$s(tr,d) = g(tr,d) \cdot (c_{tr}(d) + \alpha \cdot \sum_{p \varepsilon tr}^{N} p_N(d-p)) \tag{3.12}$$

Different new terms have been employed and they are going to be explicated :

- $g(tr,d)$ is called the "Gating function" [1]. It refers to another criterion of the association, different from the distance. The next writing focuses on it.

- $c_{tr}(d)$ refers to a classifier that is trained on the tracker *tr* and evaluates on the detection *d*. The further next writing focuses on it.

- α is a constant. It has to be defined by the user.

- $p_N(d-p)$ is approximated, thanks to [1], by the Normal distribution $N(d-p;0,\sigma_{det}^2)$. This Normal distribution has zero mean and it is evaluated for the distance between every particle of the tracker and the detection. And the variance is "inversely proportional to the velocity" [1]. Hence, in practice, it is chosen by the user.

This score function evaluates all the pairs of tracker-detection by computing the distance of each particle of the given tracker to the detection.

3.4.6.2 Gating function

As mentioned above, the distance between a tracker and a detection is not the only parameter to consider. Another criterion is the fact of considering that the future associated tracker should correspond to the detection in location, in velocity but also in motion direction.

In order to complete that, the "probabilistic gating function" [1] has its importance. Its formula is detailed below :

$$\begin{aligned} g(tr,d) &= p(size_d|tr)p(pos_d|tr) \\ &= \begin{cases} p_N(\frac{size_{tr}-size_d}{size_{tr}}) \cdot p_N(|d-tr|) \\ p_N(\frac{size_{tr}-size_d}{size_{tr}}) \cdot p_N(dist(d,\upsilon_{tr})) \end{cases} \end{aligned} \tag{3.13}$$

In equation *3.13*, the probabilities are obtained from a Normal distribution with zero mean and the variance is specified as the argument inside the brackets.

The first term only uses the sizes of the detection and the tracker. In the case of different values, the ratio will be large in absolute value. Similar parameters make the ratio closer to one and increase the value of the Normal distribution.

The second term differs in both conditions in equation *3.13*. A high velocity of the target implies that the object cannot modify its direction. This is linked to the inertia in physics [1]. As a result of that, what can be called a slow speed target has an inertia that is considered as non-existent. And the Normal distribution is evaluated on the distance between the detection and the tracker. The velocity does not appear there. In the second case, with a high speed velocity target, the inertia influences the object. Therefore, the final term uses the shortest distance between a point, the detection, and a vector, the direction of the velocity of the tracker, also called the motion direction. For these second terms, a small value between the detection and the tracker or between the detection and the motion direction of the tracker is responsible of a great evaluation of the Normal distribution. The gating function, as the product of two probabilities, has thus significant result.

This aspect of fast or slow comes from a comparison of the υ_{tr}, in absolute value, to a threshold, also defined by the user.

3.4.6.3 Boosted classifier

A classifier is used in the score function to add some information about the belonging of a tracker to a detection. This classifier comes from the family of the boosting classifier, developed in the section *3.3.4* and the algorithm is compared to the one developed in [12]. A classifier is trained on every tracker that has not been associated to a detection.

3.4.7 Observation model

The aim of this subsection is to calculate the new weights of the trackers. From the original *condensation* algorithm, it refers to the measurement part, developed in section *3.2.2.3*. The reader can imagine that the earlier subsection has no relation with the initial particle filter algorithm but it is wrong. The data association has now its usage.

The weight of a particle that belongs to a tracker is related to a probability : given the propagated particle, this is the likelihood of a new observation. It can be written as :

$$\omega_{tr,p} = p(y_t|x_t^{(i)}) \tag{3.14}$$

It uses the notion of state and observation, both studied previously in different parts.

This model is based on different terms. The first one comes from the detection and uses the results of the data association. The second one is related to the evaluation of the detector output. And the last one is the result of a classifier, the classifier term. The equation of the model is :

$$\omega_{tr,p} = detection + detector + classifier \tag{3.15}$$

The three of them are explained below.

3.4.7.1 Detection term

As explained above, the detection term is related to the fact of having a tracker associated to the detection. The estimation of the weight is mostly affected by this term in the case of an association. Its equation is :

$$detection = \beta \cdot I(tr) \cdot p_N(p - d^*) \tag{3.16}$$

The explanation of the three terms of this dot product is :

- β : this is a constant that has to be defined by the user.
- $I(tr)$: this is an "indicator function" [1]. It has two possible results : if there is an association between a tracker and a detection, the result is 1, in the other case it is 0.
- $p_N(p - d^*)$: this terms immediately refers to what has been studied in the data association part section *3.4.6.1* but the Normal distribution is evaluated on the

distance between the particle and the association detection (mentioned by the star).

3.4.7.2 Detector confidence density term

The aim of the detector confidence density term is to assess the result of the detector at the position of the tracker. The equation is :

$$detector = \gamma \cdot d_c(p) \cdot p_o(tr) \tag{3.17}$$

Three terms come in :

- γ : this is a constant that has to be defined by the user.

- $d_c(p)$: the SVM produces a reliance indicator during the HOG algorithm. This is the value used. Hence, [1] detailed that there is not a high confidence in this term because some mistakes due to background influence it.

- $p_o(tr)$: To solve this problem, a new function has been created and its aim is to "assess the reliability of the detector confidence density" [1]. Its formula is :

$$p_o(tr) = \begin{cases} 1 & \text{if } I(tr) = 1 \\ max_{tr':I(tr')=1} p_N(tr - tr') & \text{elif } \exists I(tr') = 1 \\ 0 & \text{else} \end{cases} \tag{3.18}$$

The end of the explanation is required to understand the formula. An assumption is made about the fact that, in case that there is another associated tracker near the tracker we are interested in, the likelihood of not tracking a background element is great. In other words, a strong value is obtained when the function is evaluated for a tracker that has a neighbour with an associated detection.

In the first case, $p_o(tr)$ is equal to 1 if *tr* is associated to a detection. In the second case, if another tracker *tr'* is close to *tr*, $p_o(tr)$ is obtained by computing the maximum value of the Normal distribution evaluates on the distance between both trackers. Finally, if there is none associated detection to the tracker *tr*, this function has zero value.

3.4.7.3 Classifier term

The goal of this final term is to increase the robustness of the calculus of the weight of a particle. This classifier term comes from the evaluation of the boosted classifier, detailed in section *3.4.6.3*, trained on the tracker *tr* and evaluated on the position of the particles. Its formula is :

$$classifier = \eta \cdot c_{tr}(p) \tag{3.19}$$

The classifier term is made of two terms :

- η : this is a constant that has to be defined by the user.
- $c_{tr}(p)$: this is the value that comes from the boosted classifier.

The method based on the particle filter algorithm has been detailed, with its origin, its working and each step presented in the literature. With this solid background, the project management is the continuation of it.

Chapter 4

Research methodology

This chapter presents all the project steps, since its beginning until the end of the dissertation. Given the fact that this is a computer vision work, some questions are necessarily raised, about the data, their processing and the way of constructing the final algorithm that is going to be implemented and analysed.

The different stages are explained and justified. Indeed, this section implies choices that have been made and that have a major impact on the result of the dissertation.

4.1 Dataset

A common point between the Computer Science dissertations is their need of data and this is even more true for a computer vision project that requires video scenes to produce a visual result.

4.1.1 Dataset requirements

The choice of a dataset has been a tough step. Some requirements have been formulated in the subject of the dissertation and have to be respected.

First, the description of the subject says ”track vehicles crossing a junction and pedestrians in a shopping mall” and the deliverable ”application that performs tracking, without knowing in advance what is to be tracked”. To sum up, datasets dealing with pedestrians and cars have been searched.

Second, it has been chosen to have two different datasets. This approach permits to have a better critical look by analysing the accuracy of the tracker on videos that do not deal with the same kind of object of interest.

Third, from the chosen article [1], in the experimental section, the authors have tested their algorithm with five distinct datasets. They come from a university in Zurich, ETH, and deals with pedestrians crossing a street, such as in TUD. A dataset from the Image Library for Intelligent Detection Systems, i-LIDS, is focused on people waiting for the subway. And then, in the dataset from the University of Colombia, UBC, and another one not specified, hockey and soccer games are respectively captured. As a first step, none of these have been selected due to their complexity. Indeed, even if the recording conditions are appropriated, people are always mixed to others and the videos are too long (more than five minutes for i-LIDS). The other reason is the lack of videos about cars.

4.1.2 Highway dataset

The first dataset, that has been chosen, comes from the website [22], in the basic category. It is called *HIGHWAY*. Some cars are moving in diagonal from the top right of the video scene and with a relatively constant move. One of the main advantage is the fact that the number of cars increases by time. Consequently, the algorithm is first applied on the simplest situation. And then, other vehicles appear on the screen and should be tracked. In this dataset, there are 1700 frames and the original speed of the video is 40 frames per second.

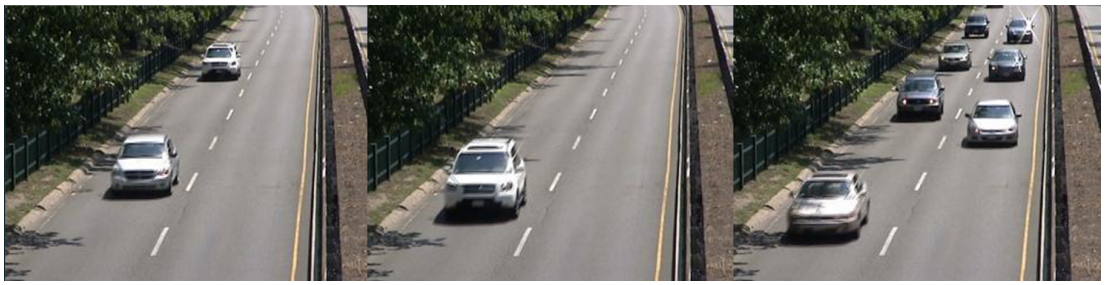

Figure 4.1: Sample of images from the HIGHWAY dataset

4.1.3 Pets2016 dataset

The second dataset, that has been chosen, also comes from the website [22], in the basic category. It is called *PETS2016* and is related with video surveillance, also called CCTV. It happens in a train station. Someone is walking to the centre of the video and stops there. After the walk of other people from side to side, the person leaves a bag

and walks away. The strength of this dataset is the fact of having the succession of one person and multiple people in the same scene and also having a motionless person. This represents some challenges to solve by the algorithm. In this dataset, there are 1185 images and the original speed of the video is 40 frames per second.

Figure 4.2: Sample of images from the PETS2016 dataset

Finally, the diversity of the data is strengthened by the fact that the first dataset is an outdoor video and the second one an indoor video. Different recording conditions are studied.

4.1.4 Ethics

The aim of this project is to track people and objects in video scenes. This implies using dataset where pedestrians are crossing road, as explained above. Consequently, an ethic analysis has to be done.

The fact of recording people in a video implies making them signed a proof of their consent because their image is used in data processing. It is sensitive and controlled. In order to avoid the claim of people consent, a method has been chosen. The choice is to use public datasets containing people crossing roads and moving cars. The literature gives examples of them, and especially in [1], which are only for research purposes.

The researchers, that have worked on the datasets **SBMnet**, ask the users of their data to acknowledge their achievement by the mention of their website *www.SceneBackgroundModeling.net*. This is the case for both dataset used, *HIGHWAY* and *PETS2006*.

4.2 Pre-processing of the data

In the process of the use of data, the step of pre-process is the first one. It represents a possibility of improving the quality of the data by applying some algorithms. Hence, in a data-driven project, the ground and form of the video have a high importance. A basic method is detailed below.

4.2.1 Histogram equalisation

The histogram equalisation is a basic way of improving an image. It belongs to the category of grey-level processing [6]. New colours are assigned but they still have the same order (in the histogram of the image). The formula of the histogram equalisation is :

$$f(i) = \frac{1}{N}\sum_{j} n(j) \tag{4.1}$$

This function outputs a new colour given the total number of colours, defined by *N*, and the number of pixels with the colour *j*, represented by *n(j)*.

The data obtained from [22] have already been prepared and are considered as ready-to-use.

4.3 Design

The design of an algorithm is a process that happens before the start of the implementation. It does not require any computer programming skills but only an accurate vision of the theoretical algorithm to facilitate its development. The following design makes a parallel with the chapter *3*. Some changes are made from this chapter and are justified to obtain a complete application. Moreover, some of them add input parameters, that is why the input variables are detailed in a lower subsection.

4.3.1 Detection module

The design is also the best moment to check if all the parts studied in the literature can be used. The main parts are the tracking and the detection. The only constraint in the deliverable is the fact that it is not possible to use some external models of the objects we want to track. It means that the algorithm has not to make a comparison with an

existing model of the data (the shape of the target for example) that will be tracked. This is not the case in the tracking part.

The detection part, thus, is concerned because of the SVM algorithm. Another method that respects the prerequisite has been found, called *background subtraction*. Briefly, this approach models every pixel of the background with a GMM (explained in section *3.3.2*) [25]. Therefore, none external learning is used. This technique only depends on one parameter, the number of Gaussians, that can be assimilated to the number of background areas. By computing the difference between the current frame and the established model of the background, it is possible to perform the detection of moving objects.

Consequently, this implies changes in the observation model (section *3.4.7*). The detector confidence density term is not considered anymore.

Even if the detection part is not really developed in [1], it should be seen as an important one. An efficient detector is desired.

4.3.2 Tracking module

The initialisation and the termination of the tracker are made by following the section *3.4*, as the data association, the observation likelihood function, the resampling and the propagation. The evaluation of the classifier has been, though, modified.

The fact of creating a boosted classifier respects the subject of the dissertation but it represents performance requirements for the computer used to run the program.

Consequently, another method has been designed to keep this notion of classifier. Given a tracker, the new classifier uses texture information to compare the current detection with the the previous associated detection. To be more precise, the colour is the comparison factor to measure the overlap of the histograms of the colours of the detections. This new classifier can be then called the *overlap histogram classifier*.

4.3.3 Input and output of the application

Different parameters are in the input of the program.

- A video with objects of interest to track.
- Number of Gaussians for the detection : this value influences the modelling of the background. An efficient detector is required. The right parameter is found by doing tests and a visual analysis. An evaluation of the accuracy is done.

- Number of particles : this value represents the candidate points that followed the objects of interest. A high value implies a better probability for a successfully tracking but the speed of the application will nevertheless be reduced.

- Number of frames for the tracker to survive : a tracker is not always associated to a detector because the best influence on the particles is desired. As a result of that, trackers are saved to become new candidates for the next associations. This saving requires computer memory and impacts the speed of the application.

- Threshold for data association : this value is used to estimate if an association of a tracker and a detection is valid or not (explained in section *3.4.6.1*). Even if, at the end of the corresponding module the maximum value is selected, as other previous parameters, the efficiency of the application depends on it because it is not interesting to consider useless results.

Other parameters, mentioned during the section *3.4.4*, that can be called *additional parameters*, are required : α, β, η. It is a challenge to find the best set of parameters because they widely affect this tracking application.

The output is a video with particles that follow the objects of interest (people or cars).

From what has been explained in this section, it is possible to have a clear view of the design of the future application. The figure above is the block diagram of the tracking by detection application. It sums up most of the modules and parts that will be implemented.

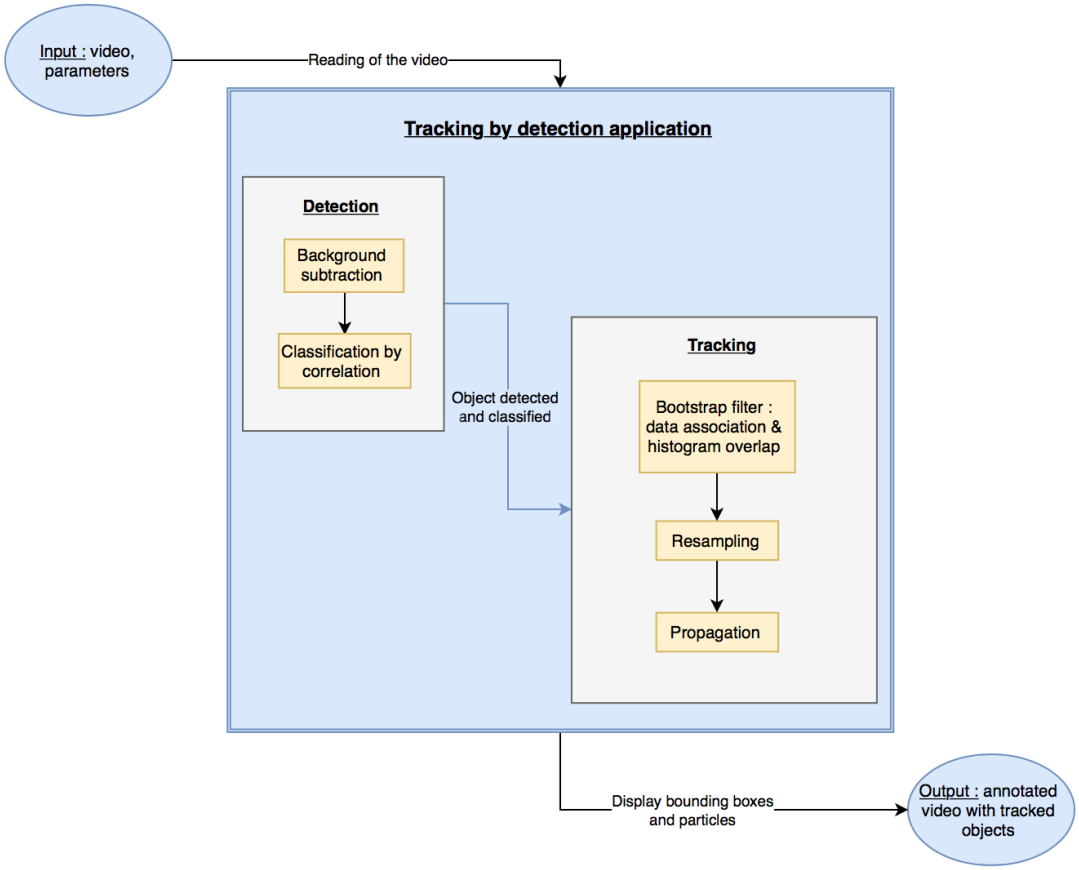

Figure 4.3: Block diagram of the tracking by detection application

4.4 Evaluation

From the description of the subject ”select suitable metrics that you will use to evaluate the tracker’s accuracy” and from the deliverable ”An evaluation of the accuracy”. As a result of that, the idea is to make an evaluation of the different modules that are implemented to be able to make a criticism of the parts. This is used for finding the best input parameters. In order to evaluate the truth of the tracker, different measures are achieved. Some of them are inspired from [1], for example the precision, the accuracy, the false positive and false negative rates.

4.5 Implementation

After the build of a coherent design, the next part is about implementation. From this situation, several decisions have to be made. This section deals with pure implementation but also about the way of leading the implementation of this project in productive manner.

4.5.1 Language selection

As a student in computer science, programming languages are approached as tools. It is a good point to know some of them to be able to have a criticise regard on them to make the better choice.

Different parameters have to be analysed : the known of the language and its syntax, the available libraries related to the subject, the needed environment and the fact that it can be used in computer vision. It is also necessary that all the project have to be developed with the same language. Its performance is the final criterion because the processing speed is researched, even if real-time is not an imposed constraint.

There are only few programming languages that are computer vision oriented and that can satisfy the previous criteria. The candidates are Python, Matlab, R and C++.

From the listed languages, even if it is open-source, R is the less popular in computer vision. After some researches about C++, it is an approved free language with a huge community and optimised libraries. However, the main problem is my knowledge even if it is not a problem in some extent, but it is definitely tough for beginners. Matlab is a well a candidate because it is used for machine-learning and has undeniable qualities for processing performance. It is often used in computer vision because some libraries exist. The cost of the license is the major weakness. The final choice has been made, Python. This well-known and open-source programming language is mostly used in data science and in computer vision thanks to its amount of libraries. Skills in this user-friendly language also represent an interesting benefit. Another aspect in favour of Python is the fact that some possibilities and extensions are possible, for example the creation of a user-interface. Others are discussed in chapter *8*.

4.5.2 Environments

The development environment and libraries represent an efficient way of coding. Indeed, powerful tools exist and it is obvious that it is important to use this opportunity.

- OpenCV : this is the main library for image processing. It has been developed in C++ especially ”for real time computer vision” [24]. The Python version is not as full as the C++ version because there are yet some developments on it. Some functions are missing in the documentation as well.

- Anaconda : this is a ”package manager, an environment manager, a Python distribution, and a collection of over 720 open source packages” [4]. From the number of packages that are contented in it, Conda is another possibility that requires less memory. Some of the used libraries are Numpy, to efficiently deal with arrays, Scipy, for the mathematics calculation, and Matplotlib, the plotting Python library.

- PyCharm : this is an Integrated Development Environment, (IDE), for Python computing [15]. It is cross-platform and user-friendly. It runs the full project.

- Atom : this is an open-source text editor developed by Github in NodeJS [11]. It has been used for the modules creation.

- Jupyter Notebook : this is an ”open-source web application that allows you to create and share documents that contain live code, equations, visualisations” [16]. Its usage is especially for the tests made during the development of the different functions.

4.5.3 Implementation methodology

In a long period of work as the dissertation, the organisation has to be good. The choice of implementing small modules has been made. In a short time, a complete module of the project is made and has to be functional. Some tests and corrections follow this step in order to obtain the desired module. The aim is to be fast and efficient. The following tools have been used to monitor the progress of the implementation of the dissertation subject :

- Trello : this a web application for projects management. It uses boards, lists and cards to check the progress of tasks [34].

- Google Drive : this is a web application for storage and synchronisation. Different applications in Drive have been used : Google Docs, for writing, and Google Sheet to do timelines. Its main advantage is that it is possible to access to this website everywhere [27].

From this explanation of the project stages, with the presentation of the data, the design of the algorithm, its evaluation and the decisions about the implementation, the development step is then precisely enlightened.

Chapter 5

Implementation

This chapter details the different modules that have been implemented for the dissertation in order to build a tracking by detection application. It resumes, explained all the development stages, mentioned the problems met during this part and the used solutions.

5.1 Processing dataset

After having selected the dataset in section *4.1*, *HIGHWAY* and *PETS2016* have been downloaded. Their repositories contain all the frames in image format.

In order to obtain a ready-to-use video as input, the aim is to concatenate all the files. A Linux command, used in the terminal, is the solution to this problem : *ffmpeg*. This command has three inputs : the speed of the desired video, in frame-per-second, the images files and the name of the output. The encoding is automatically made.

The following figure presents the processing of the dataset *HIGHWAY*.

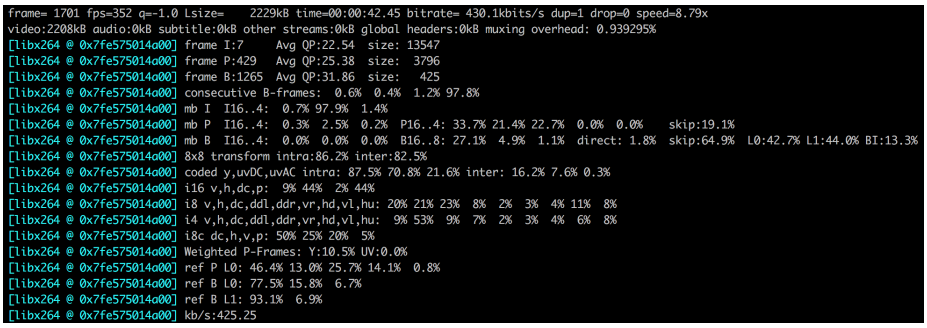

Figure 5.1: Processing of the HIGHWAY dataset

5.2 Data model

In this section, the imagination and the creation of the data model is developed.

In a desire for simplicity and flexibility, it has been decided to not create a database. The data are saved by Python in the main file.

Python has several structures to store data : array, list or dictionary. Of course, it is possible to mix them. Arrays were selected at first but, after different tries, dictionaries have been chosen. The key-value system is useful and there are several methods to quickly operate on it [5]. The variables that are only used to make the code properly run are not explained (for example in a *for* loop).

It is necessary to prepare two dictionaries. The first one concerns the detections. It has the following form :

- Key : the key described the number of the detection. It is assigned in an iterative way.
- Value : the value is a list of twelve elements. There are the histogram of the colours of the detection, the coordinates on the X-axis and the Y-axis of the centre of detection, the four points of the shape, called the bounding box, that described the detection of an object of interest, the group of the detection (to operate on the same detections), the velocity of the target, the time stamp of the detection, the area of the bounding box of the detection and the colour of the detection.

In order to be precise, an explanation is made for three elements. A time stamp is the computer science oriented word for the time when the measure has been made. The colour of the detection is the visual result of detection and the group of the detection is the output of the classifier presented in section *4.3.1*. Similar detections have the same group number near the bounding box border.

The second dictionary is used for the trackers and is made of :

- Key : the key described the number of the tracker. It is assigned in an iterative way.
- Value : the value is a list of the particles of the tracker. And a particle has the form of another list of thirteen elements. There are the coordinates on the X-axis and the Y-axis of the centre of the particle, the weight of the particle,

the coordinates on the same axis of the centre of the associated detection, the size of the associated target, the number of the frame when the particle is born, the motion direction, the velocity of the particle, the number of the tracker this particle belongs to, the number of frames this particle has successfully tracked an object and the colour of the particle.

The last element, the colour of the particle, is a visual indication that described which particle has just been created. It helps the user to differentiate initialised trackers from the others.

5.3 Modules

In this section, the different modules that have been implemented are described in the order of their appearance in the code.

5.3.1 Initialisation of the algorithm

This small module deals with the input parameters, listed in section *4.3.3*, and reads the video with *OpenCV*. Then, a time stamp is also started to measure the velocities.

And then, in order to improve the speed of the algorithm, a Normal distribution is constructed with zero mean and a variance defined by the user. Indeed, this choice is reasonable because in [1], the same value is used for all the Normal distributions in the data association (section *3.4.6*) and in the observation model (section *3.4.7*). Their evaluations are made when they are used.

5.3.2 Detection module

In the detection module, the first step is to realise a background subtraction. *OpenCV* integrates two methods for that. The first one, *BackgroundSubtractorMOG* is the implementation of the method explained in section *4.3.1*. Another one, *BackgroundSubtractorMOG2* can be considered as more evolved. Based on [36], it solves the fact that the user has to choose the number of Gaussians. The algorithm automatically selects the best number of Gaussian distributions to model the background.

The model is now applied on each frame with a learning rate, decided by the user. Then, the contours of each object of interest are found. Concretely, a foreground mask

is determined from the subtraction of the current frame and the model of the background. The following images are examples, for each dataset, of a current frame and the foreground mask calculated by *OpenCV*.

Figure 5.2: Example of a frame and its foreground mask from the PETS2016 dataset

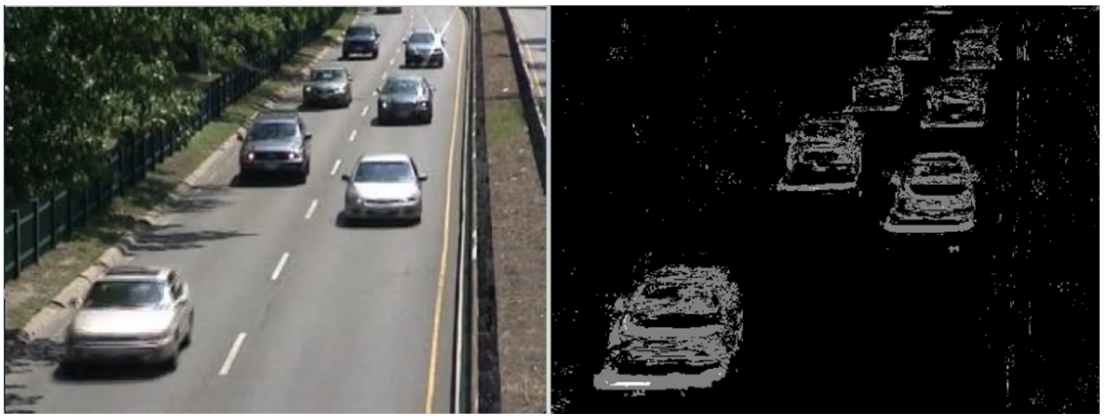

Figure 5.3: Example of a frame and its foreground mask from the HIGHWAY dataset

In order to not detect noisy elements on the video, the area of detection is computed and a threshold is the acceptance criterion. If the returned value is adequate, a rectangular bounding box is drawn with a random colour for the borders. The centre is also obtained, with an *OpenCV* method, *moments*. To facilitate visual interpretation, a point with the coordinates of the centre is designed.

From the data model established above, the histogram of the detection is required. A computing is done because *OpenCV* stores images in the Blue Green Red (BGR) format rather than Red Green Blue (RGB), used by *Matplotlib*, in section *5.3.9*. After

a conversion of the format of the frame, the histogram of the detection is calculated and normalised.

The detections are then stored. If the corresponding dictionary is empty, they are all added to it. In the other case, the detection classifier is used. Briefly, the idea is to evaluate if the recent detection corresponds to a new object or not. For example, if a person appears in a frame, it is considered as a new one because its appearance has never been seen before. This is done by comparing the colour histogram of the detection with the saved detections. Different metrics can be applied with *OpenCV* for the comparison and the correlation has been chosen. A threshold then decides if the candidate histogram belongs to a new group of detections or not. It can be possible that a threshold should let the candidate detection be associated to multiple detections. The position of the centres of the groups of detections are compared by computing their euclidean distance to the input detection centre to finally decide its classification. This two factors clustering and the threshold bring robustness and confidence to the detector.

The different results are finally filled in the dictionary of the detections. The following images are some examples of the result of the detector applied on both datasets.

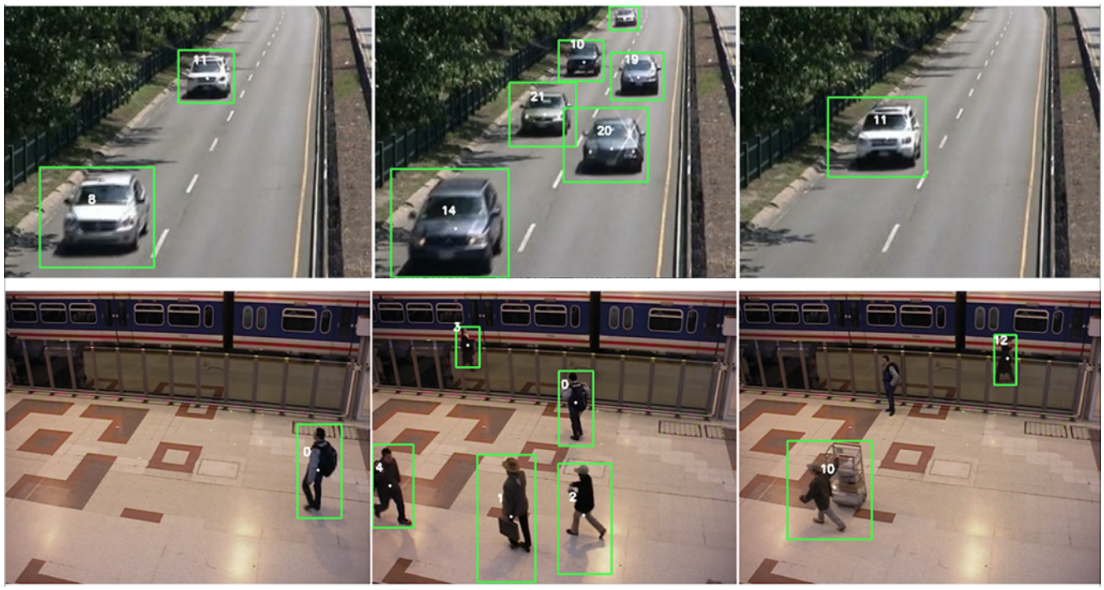

Figure 5.4: Output of the detection module with the PETS2016 dataset (bottom). Output of the detection module with the HIGHWAY dataset (top)

5.3.3 Tracking module

The implementation steps that occur during the tracking stage are described. Because it is not possible to bring a visual for each of them, images are placed at the end of this subsection.

5.3.3.1 Removal of old particles and detections

Initially, as detailed in the section *3.4.5*, unassociated trackers are deleted after a certain number of frames. It has been decided to do a similar thing for the detections.

In order to pre-select the particles that are not associated, which is the first condition of the removal, a test is required. Each particle of each tracker has, when it is stored, the information of a detection centre. If it is not associated, this value is null. A second test about the frame number is realised, thanks to a threshold decided by the user.

For the detections, a basic process has been applied. With a threshold, the user has to decide the maximum number of detections by groups required by the program. The best interest is to not have too many collected data.

5.3.3.2 Data association module

The data association module is the module that has required the most development. Its structure and its way of analysing data are sophisticated.

The entire combinations of the tracker detection pairs have to be tested. The reason of the choice of the structure, stated in section *5.2*, mostly comes from there. In the case of having *lists* selected to represent trackers and detections, the complexity of the establishment of all the pairs is $\Theta(n^4)$ because the particles are *lists* of the *lists* of the trackers and the trackers are gathered in a final list. This is similar for the detections.

The use of dictionaries removes a *list* with the *key* system. Consequently, the complexity of the program is reduced. Moreover, a Python library to construct the pairs exists and the construction is computed as a Cartesian product of the all the detections and trackers.

From all the available associations, the one with the highest score, and greater than a threshold, is selected. The three important data, the tracker, the related detection and the matching score, are kept in a temporary dictionary. From this point, the current frame is considered as successfully tracked by the tracker. It is important for the section *5.3.3.7*.

5.3.3.3 Bootstrap filter module

The bootstrap filter is the observation model. It estimates the new weight of each particle. The previous results of the data association are used there. All the associations in the corresponding dictionary are treated.

First, the detection term is computed with the evaluation of the Normal distribution on the Euclidean distance between the detection centre and the particle centre.

Second, the classifier term is measured by comparing the histogram of the colours of the associated detection and the histogram of the colours of the preceding one. The employed metric is the histogram overlapping, as explained in section *4.3.2*.

Third, the new weight of each particle is the sum of the classifier term and the detection term.

Finally, the data structure of the associated particles is updated with the temporary dictionary to compute the proper information in the future.

5.3.3.4 Resampling module

The resampling module is a challenging module because its understanding implies a certain abstract level.

An iteration is made on every tracker of the corresponding dictionary. The weights are firstly added and normalised. Then, then aim is to generate a non-uniform sample of the size the number of particles, and without replacement. The *choice* method from the *Numpy* library is used. The normalised weights are the probability of the weights. By processing the index of the weights instead of the weights, mistakes made because of identical values are avoided. And then, the dictionary of the trackers is updated with these new elements.

5.3.3.5 Propagation module

The propagation module is considered as a module for the update of the position of the particles and their velocity. It uses the expressions taken from the literature (section *3.4.4.3*).

A first Normal distribution, with zero mean and with the size of the tracker as variance, is randomly evaluated. This is the noise position. A second Normal distribution, with zero mean and with the inverse of the number of successfully tracked frames as variance, is also randomly computed.

The elapsed time is measured by a difference of time stamps between the current and the precedent frame. The value is converted in seconds.

The position and the velocity of the particles are set with those new values and the dictionary of the trackers too. A final condition validates the particles in the case of an appropriate estimation of their position. It uses the fact that a particle cannot have its position outside of the video sizes.

5.3.3.6 Initialisation of the trackers module

The initialisation of the trackers corresponds to them add to the dictionary of the trackers. In a search of performance, because the particles of the next state are created at the position of the current observation, they are stored in a temporary dictionary before them add to the correct one at this right moment, to fit with the particle filter algorithm.

The real creation of the particles is done during the detection part to avoid additional *for* loops, that have a complexity to consider.

A random Normal distribution is applied with the mean as the centre of the detection and the variance as the minimum between the length and the width of the bounding box. Indeed, the spread of the particles is related to the variance and they have to stay inside the box.

The motion direction is then computed as the orthogonal direction to the closest frame border.

5.3.3.7 Display module

The final module displays the particles on the user screen. It prints all the elements of the dictionary of the trackers, thanks to *OpenCV*, by considering them as circles. The positions of the circles are the positions of the particles, the colours of the circles are the colours of the kinds of particles (initialised or not) and the diameters of the circles have firstly been fixed by the user. To appreciate the importance of a particle, it is also possible to set this distance to the weight of the particle.

5.3.4 User interaction

The application is running with Windows, Linux or Mac. Indeed, only a terminal is required with Python (and the listed libraries) installed. Tracking by detection is launched with the command line : *python trackingbydetection.py -v path_to_video*. The last argument is the full path to the video to analyse.

The visuals of the running application are below (figure *5.5* and figure *5.6*).

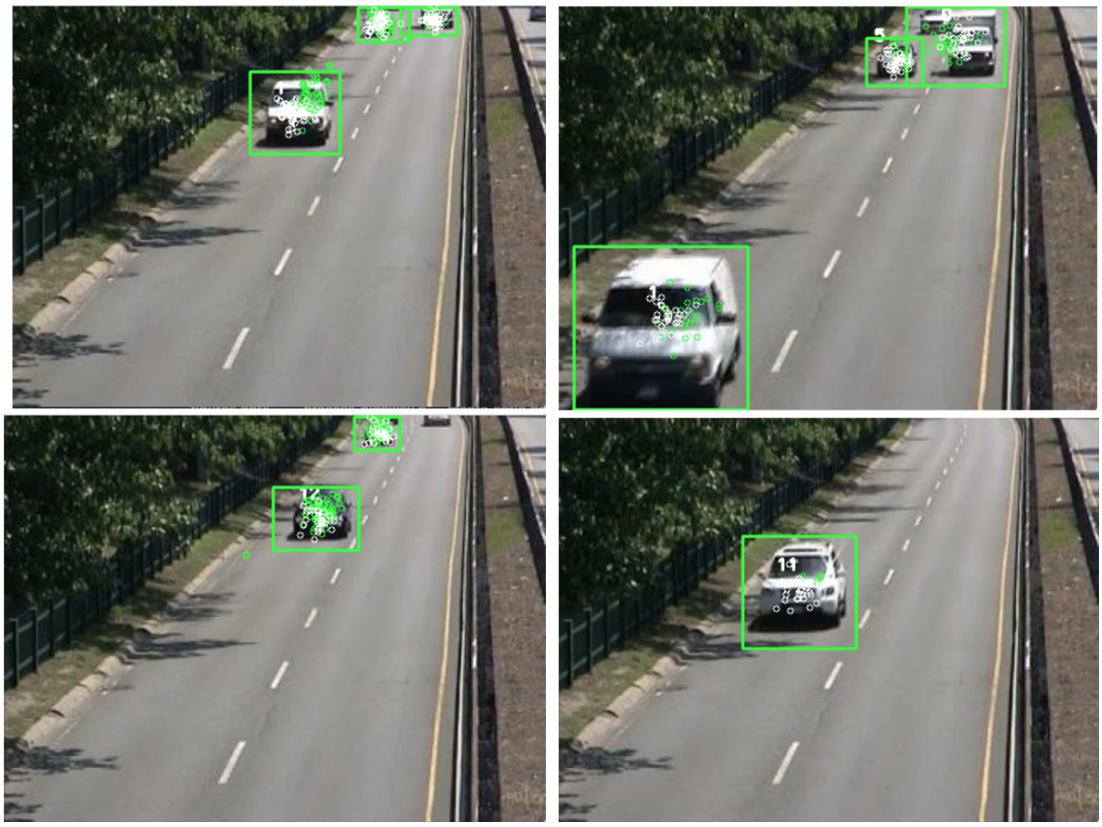

Figure 5.5: Example of the tracking by detection on the HIGHWAY dataset

Tracking by detection is automatically made by the system. The detections are the green bounding boxes, the initialised trackers are the white circles and the associated/unassociated particles are the green circles. The result of the output of the detection classifier has been conserved to give a visual assist to the user.

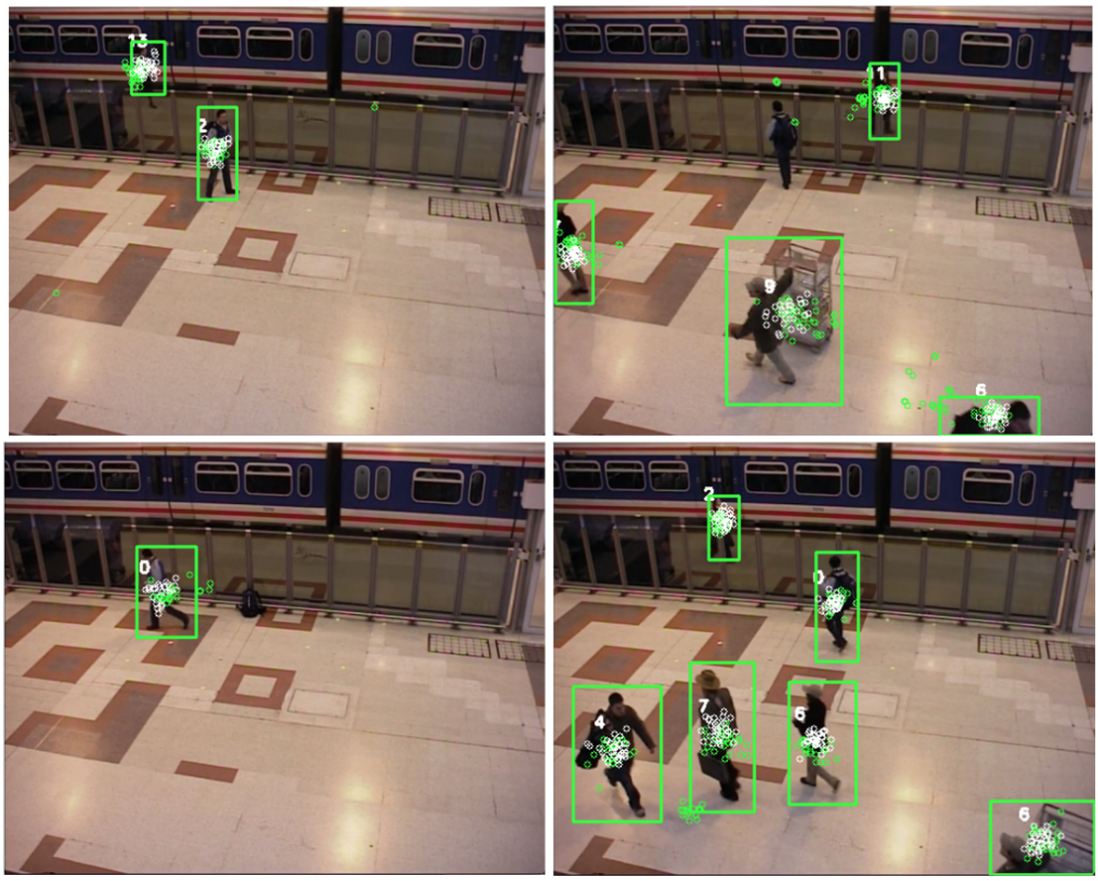

Figure 5.6: Example of the tracking by detection on the PETS2016 dataset

Chapter 6

Experiments

This chapter describes the experiments that have been made in order to test the tracking by detection application, analyse its performance in various manners and confront it to the literature. From the datasets listed in chapter *5*, some situations have been selected. Then, the tests to find the optimal input parameters are presented, followed by the measures of the accuracy of the model.

6.1 Situations selected

Different interesting situations have been extracted from the datasets. The aim is to make experiments of the tracking by detection application in case of varying circumstances. Consequently, from both PETS2016 and HIGHWAY datasets, the scenes are detailed in tables *6.1* and *6.2* with the corresponding number of frames (the reference of the scene is in brackets) :

	PETS2016			
	One person	Two/Three persons	Large amount of persons	
Number of frames	94 (P1)	132 (P2)	105 (P3-1)	129 (P3-2)

Table 6.1: Situations selected from PETS2016 dataset

	HIGHWAY			
	One moving car	Two moving cars	Large amount of moving cars	
Number of frames	76 (H1)	81 (H2)	81 (H3-1)	111 (H3-2)

Table 6.2: Situations selected from HIGHWAY dataset

6.2 Optimal parameters

The number of input parameters is important and can be divided in two categories given their belonging to the detector especially or not. The idea is to first set the detector parameters and second the parameters used by the tracker because tracking is made by detection. The detector can be thus considered as an input of the application.

In order to do the selection of the values of the parameters, tests have been made on the most evolved situations : from PETS2016, large amount of persons (P3-1), and from HIGHWAY, large amount of moving cars (H3-2). The best results of the experiments make the choice of the values of the parameters. The general effects of the parameters on the videos are explained.

6.2.1 Detection

From the literature, and especially [1], this is the module that represents the most change. Given the fact that it is used by the whole application as a base, the detection module has to be efficient. This is possible by selecting its best parameters.

For the estimation of the area of detection and the learning rate, the quality of the detection is required. The evaluation of the detection module can be done by evaluating four measures :

- True positive : the number of objects of interest that are correctly detected.
- False negative : the number of objects of interest that are not detected.
- False positive : the number of detected elements that are not some objects of interest (a background item for example).
- True negative : the number of elements that are not some objects of interest and that are not detected.

The accuracy and the precision of the detector are calculated and the maximum values of these scores are researched.

6.2.1.1 Area of detection

The area of detection is a parameter that has been created to avoid the detection of elements different than the objects of interest. It is used after having find the contours of all the detections and appears as a minimum threshold.

The width of the window of the output video is 500 pixels and the length is 370 pixels. It means that the output area is 185,000. The minimum area for the tests has been chosen starting from 1 over 600 of the output area. Consequently, the evaluation of this parameter is done by increasing it from 300 pixels to 1100 pixels.

The right measure of the area of detection is based on the quality of the detection, as explained in the introduction of section *6.2.1*. The following figure describes the results of the test.

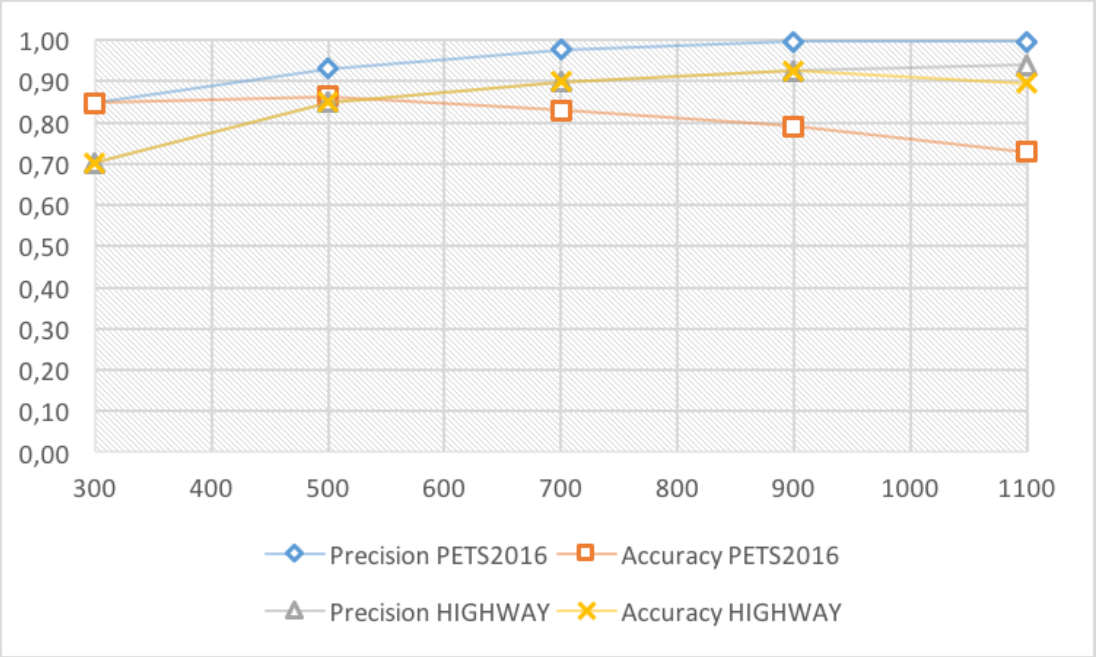

Figure 6.1: Measures of the precision and the accuracy of the detector with varying values of the area of detection. X-axis : area of detection. Y-axis : precision and accuracy measures.

A high detection area rises the precision of the detection but the accuracy slowly decreases. Noisy detections are not considered with such value but objects of interest are more often grouped together. The results are closed but the optimal chosen value is 700. At this point, the best compromise between accuracy and precision of the detection in both datasets is found.

6.2.1.2 Learning rate

The learning rate is a parameter used when the *BackgroundSubtractorMOG2* method is applied on each frame.

In order to find the best learning rate, this value is moved up from 0, the default value, to 0.5.

Again, the choice of the value of the learning rate is based on the quality of the detection, as explained in the introduction of section *6.2.1*. The following figure describes the results of the test.

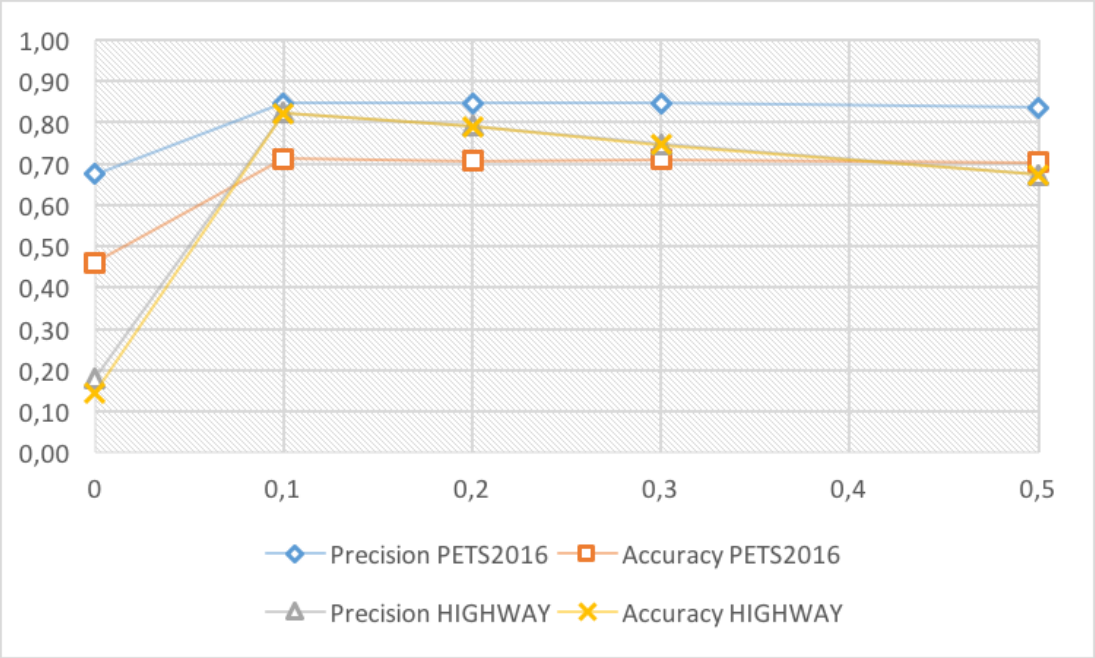

Figure 6.2: Measures of the precision and the accuracy of the detector with varying values of the learning rate. X-axis : learning rate. Y-axis : precision and accuracy measures.

The learning rate represents the way the model evolves and adapts to various video scenes. A high value is responsible of a frequent reconsidering of the background, that is why accuracy and precision decrease. The optimal measure of the learning rate is 0.1 because the maximum precision and accuracy of the detection in both datasets is found.

6.2.1.3 Threshold of classification

A detection classifier, based on the texture evaluation, replaces the boosted classifier, mentioned in [1]. It compares the colour histogram of the current frame with stored detections and a threshold is used to evaluate the comparison before computing the distance to the centres (as explained in section *5.3.2*). In other words, the threshold of the classification assesses the similarity of the detections.

The optimal threshold value is found by varying this parameter from 0.7, or a concordance of 70%, to 0.9, or a concordance of 90%. The quality of the classification is used to find the optimal threshold. Only the number of objects of interest that are correctly classified is calculated. The following figure describes the results of the test.

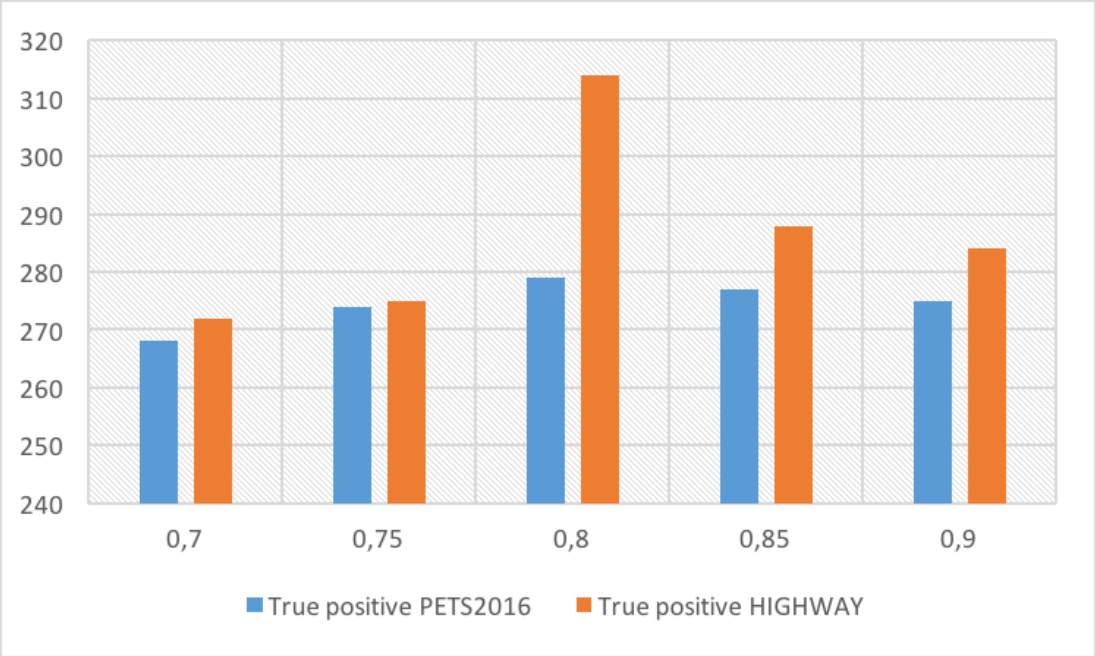

Figure 6.3: Measures of the number of true positives detections with varying values of the threshold of classification. X-axis : threshold of classification. Y-axis : number of true positive detections

People and cars are classified with the same parameters and it is a difficulty. Indeed, the detector uses texture information and cars often have the same colour. Moreover, the video taken from *PETS2016* places some persons in the darkness. It is important to adapt to these conditions. The optimal measure of the threshold of classification is 0.8 because there is the highest number of true positive detections in both datasets.

6.2.2 Tracking

The tracking uses the detection module at each iteration of the algorithm. Parameters estimation of the detector have been set to the results found in section *6.2.1*. The input variables of the tracking module are now studied. In order to do that, an assumption has been made : the detector outputs are perfect. It means that 100% of the detections made are well.

Given that fact, it is possible to evaluate the quality of the tracking. A thought about it has led up to a definition of the quality of the tracking : the number of particles that have been associated and that are located in the zone of the object. It indicates that the propagation of a particle is rightly done because, given the state of a particle, the comparison is made between the estimation of the future position of the particle, its propagation, and the next detection. In the experiments of this subsection, the notion of zone of the object is assimilated to the detection area.

In the following experiments, the number of associated and rightly propagated particles is divided by the total number of areas of the video, called *ratio zones*, and by the total number of particles, called *ratio particles*, used in the video. These comparisons, with the form of ratios, describe the accuracy of the tracking. Ratios are really different under the datasets. The detection classifier is responsible of this because cars are much less classified than people.

It is important to have in mind that the estimation of the following parameters has been tough because there was no clue about their values. Many tests have been made to have an idea about the interesting sets of parameters.

6.2.2.1 Number of frames for the tracker to survive

Without detection associated to a tracker, particles only survive for a limited time. This notion of time is expressed by a number of frames in the application. Unassociated trackers are deleted after a certain number of frames. This removal is done in the removal module (section *5.3.3.1*).

The optimum number of frames is searched by testing numbers from 1 to 7 and analysing the quality of tracking. The figure *6.4* describes the results.

From this experiments, it is possible to conclude that the number of associated and rightly propagated particles rises with an elevated number of saved frames. Indeed, the matching score is not always greater than the threshold and the tracker can be, though, useful in the next frames. The optimal value is there 6.

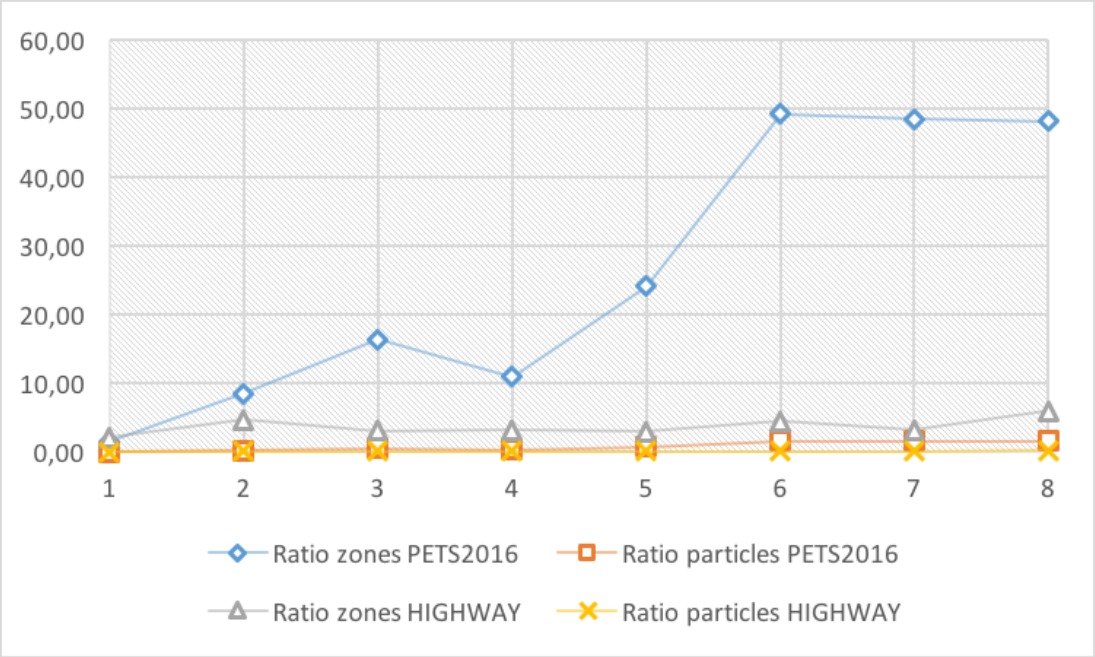

Figure 6.4: Measures of the ratios with varying values of the number of frames for the tracker to survive. X-axis : number of frames. Y-axis : measure of the ratios.

6.2.2.2 Threshold of the data association

In the data association module, trackers and detections are evaluated as pairs (section *5.3.3.2*) and a score is attributed to these pairs. A high score means a strong relation between a tracker and a detection. However, it is important to define from which value an association is satisfying.

The quality of the tracker is related to this threshold. Its value is increased from 5 to 45. The figure *6.5* describes the results.

Thanks to this experiments, a conclusion is made. A decision has been to select the maximum threshold that gives high ratios to be sure that there is a certain amount of trackers that are associated. Even if the number of associated and rightly propagated particles by zone drops, they are always equally efficient in this set. It means that 20 is the optimal number.

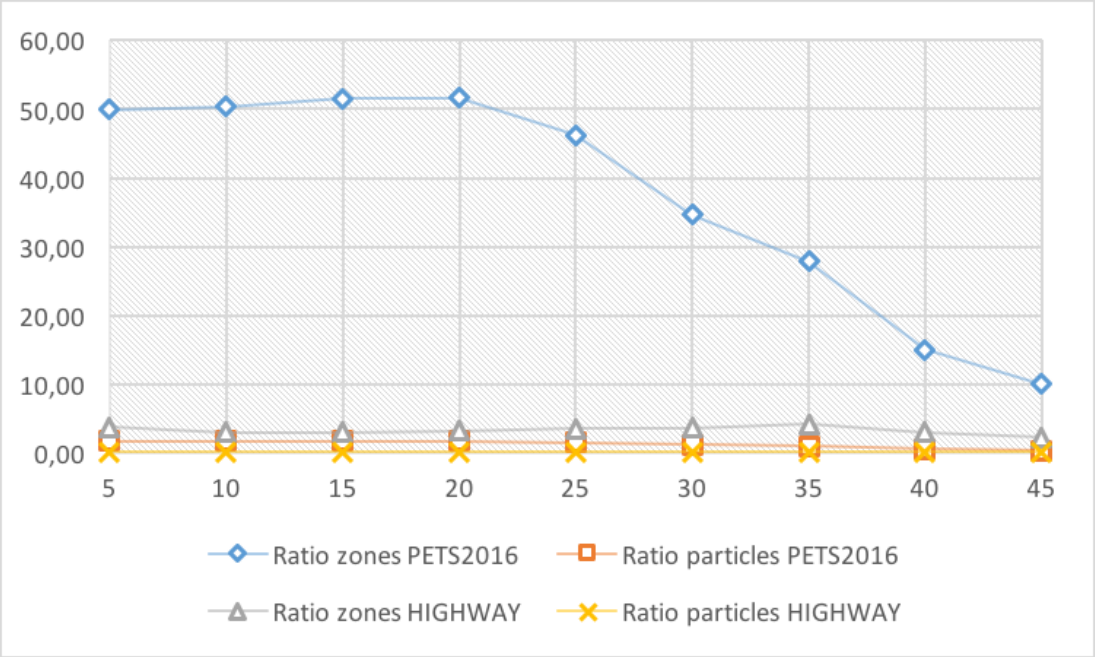

Figure 6.5: Measures of the ratios with varying values of the threshold of data association. X-axis : threshold of data association. Y-axis : measure of the ratios.

6.2.2.3 Variance of the Normal distributions

Normal distributions are often used in the application and have different purposes. Indeed, they appears in the data association module, for the estimation of the gating function for example, and in the bootstrap filter module, for the approximation of the detection term. They have zero mean and, as explained in [1], they all have the same variance : ”identical for all sequences. This was the case for the variances”.

The quality of the motion model depends on this mutual variance. The value is increased from 0 to 0.4. The figure *6.6* describes the results.

The influence of the variance in a Normal distribution is the dispersion of its values. In our case, it is the spread of the particles. In this experiments, it has been shown that the best ratios are found with an important variance and it implies a better tracker. Consequently, 0.35 has been chosen.

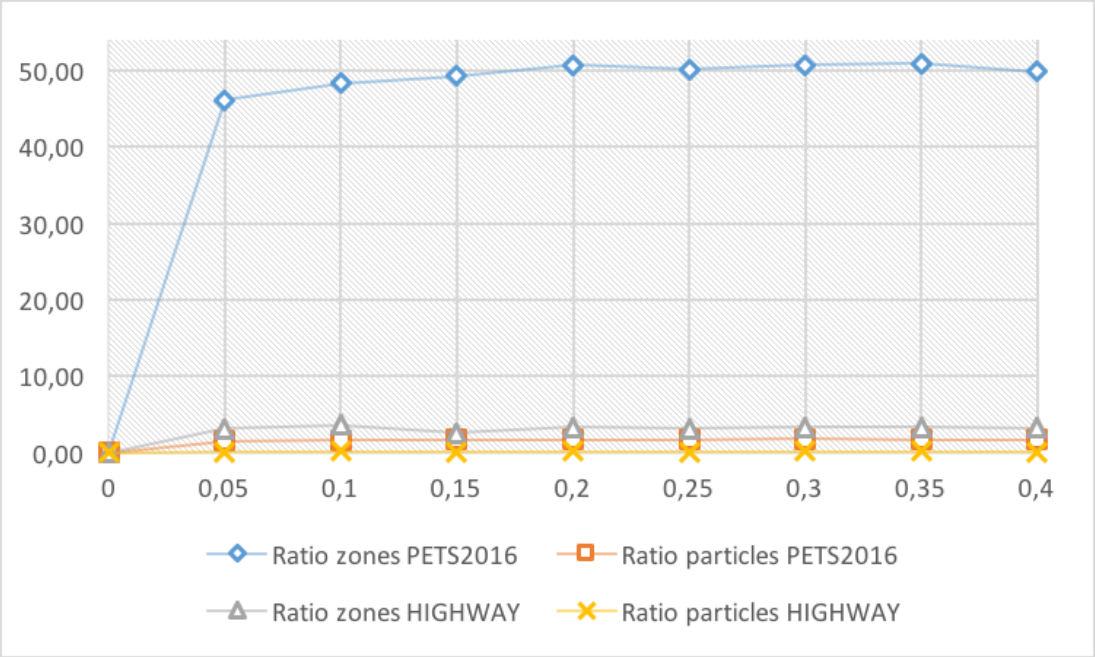

Figure 6.6: Measures of the ratios with varying values of variances of the Normal distributions. X-axis : standard deviation. Y-axis : measure of the ratios.

6.2.2.4 Number of particles

The particles are key elements in the application of tracking by detection. Indeed, they are responsible of the tracking of the objects of interest in an abstract way, by being associated to detections and moving on the frames, and in a concrete manner, by specifying to the user their positions.

They are responsible of the tracking and their number is related to the quality of the tracking. Different values from 10 to 45 have been tested. The figure *6.7* describes the results.

In this experiments, the ratio particles is the most interesting because the idea is to find a correlation between the number of particles by detection and the fact that they are associated and rightly propagated. An important number of particles increases the quality of the tracking. Indeed, there will be more candidates by initialised particles. The optimal number of initialised particles by detection is 35.

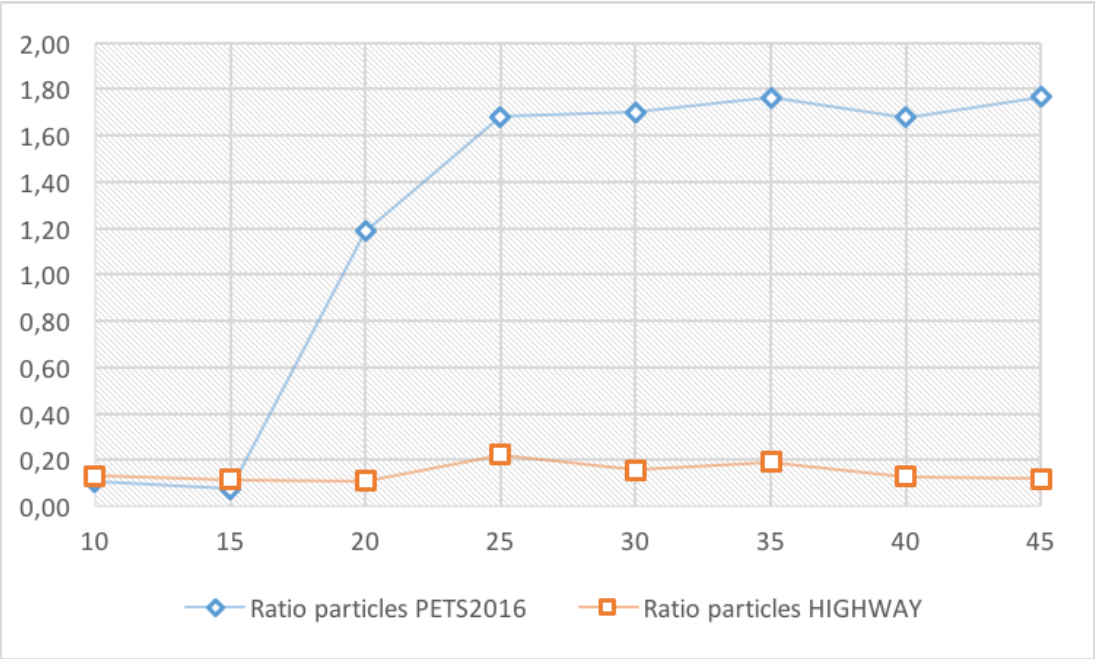

Figure 6.7: Measures of the ratio of associated particles by the number of initialised particles with varying values of the number of particles. X-axis : number of particles. Y-axis : measure of the ratio particles.

6.2.2.5 Additional parameters

Other parameters are used in the application and are fixed values defined by the user. In a way, they give some consideration to the terms they are affected to. Their estimation is thus complex. [1] brings information to the search of the best set β,η "the ratio between the respective terms are about 20:1". No clue for α has been given.

The set β,η is firstly estimated by testing different values that respect the given ratio (detailed in table *6.3*). And secondly, for α, it is similarly done by rising it from 1 to 4. The figures *6.8* and *6.9* describe the results.

	(1)	(2)	(3)	(4)	(5)	(6)
β	2	20	40	60	80	100
η	0.1	1	2	3	4	5

Table 6.3: Sets of β,η tested with the number of the test

From these experiments, it is complex to draw some conclusions about the impact of α, β and η on the model. However, their values have been chosen according to the results of the ratios to obtain the best tracking. Consequently, α is equal to 3, β is equal

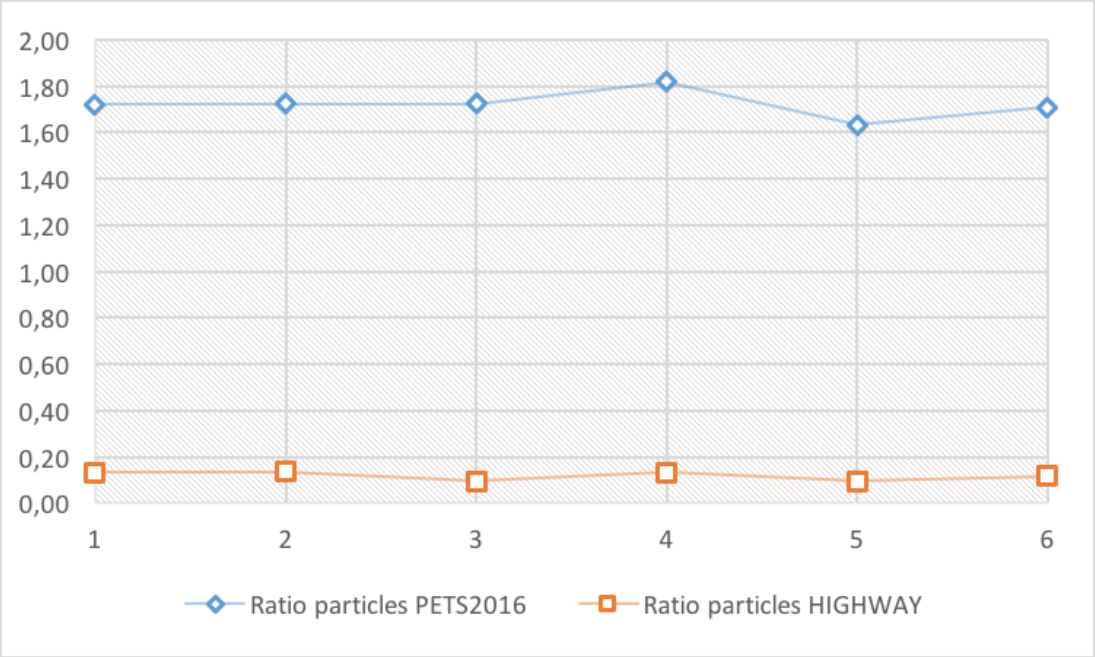

Figure 6.8: Measures of the ratios particles by varying β,η. X-axis : number of the set, from table *6.3*. Y-axis : measure of the ratio particles.

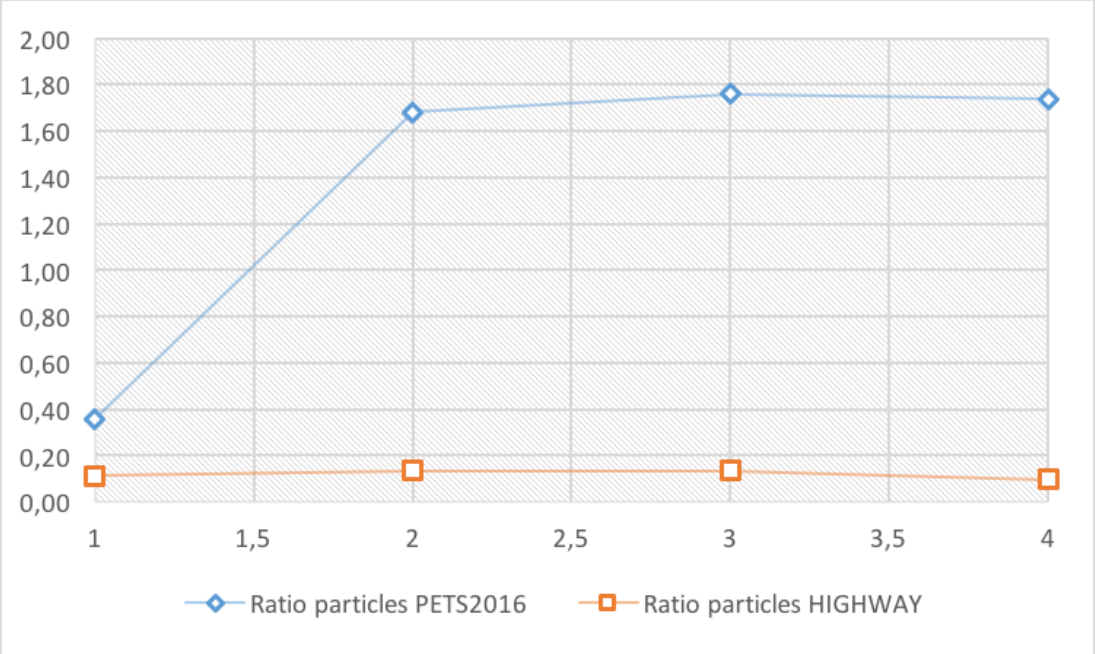

Figure 6.9: Measures of the ratios particles by varying α. X-axis : value of α. Y-axis : measure of the ratio particles.

to 60 and η is equal to 3. With greater values of α, the ratios are lower.

6.3 Performance of the model

This section presents some tests, analyses the results and concludes about the performance of the system implemented for tracking by detection. The input parameters are set to the optimal values determined in the previous section.

The data used in this section are all the situations selected from *PETS2016* and *HIGHWAY* datasets.

Different measures can be made to evaluate the system :

- True positive : the number of objects of interest that are correctly tracked by detection.
- False negative : the number of objects of interest that are not tracked by detection.
- False positive : the number of elements that are not some objects of interest (a background item for example) and that are tracked by detection.
- True negative : the number of elements that are not some objects of interest and that are not tracked by detection.

The accuracy and the precision are then calculated. Other measures are also made, like the speed of the computation of the algorithm. After varying parameters and keeping the same input videos, the aim of this section is to evaluate the model with the same parameters on different input videos.

6.3.1 Measures of accuracy and precision

The main measures of the performance of the computed system are the accuracy and the precision. Of course, the ideal system has the highest accuracy and precision. The figure *6.10* presents both measures on the different video scenes.

The P1 and H1 scenes are basics because one person and one car are moving. Both accuracy and precision between 0.9 and 1 shows that the model has no problem with such scenes.

The P2 and H2 scenes are complex, not because of the number of objects of interest involved but because of the recording conditions. Indeed, the obscurity in P2 and the

effect of wind on the trees in H2 make the tracking by detection more difficult, as can be seen on the figure.

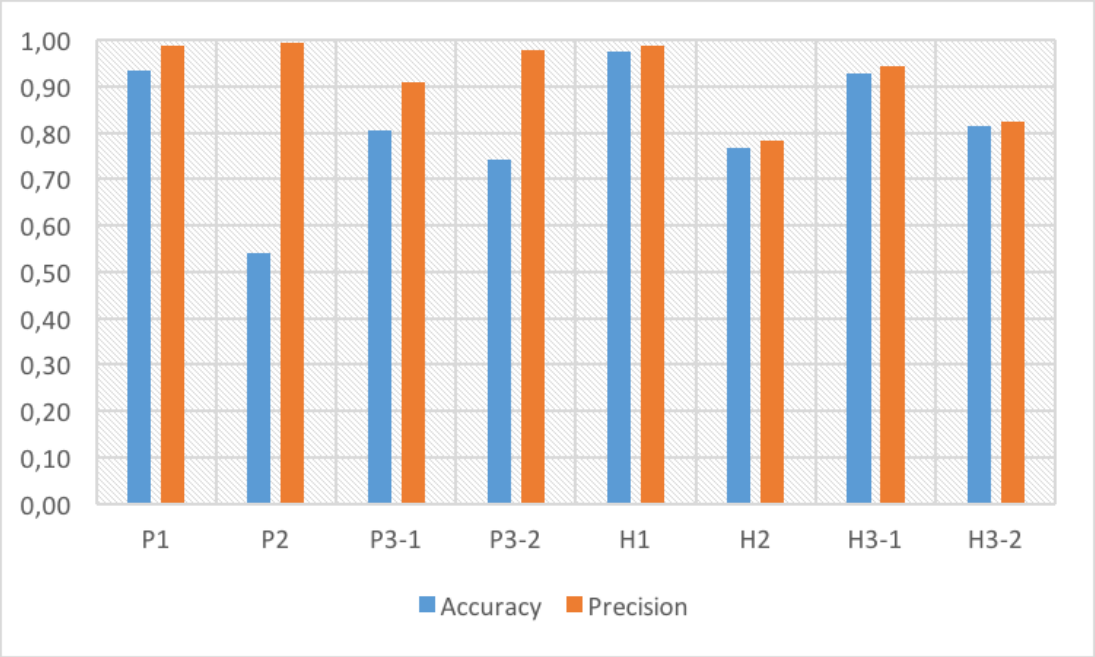

Figure 6.10: Measures of the accuracy and the precision of the model on various scenes. X-axis : video scene reference (tables *6.1* and *6.2*). Y-axis : measure of the accuracy and precision.

The last four scenes are the most evolved situations. P3-1 and H3-1 show many objects of interest, but there are not really tough conditions. That is why the model well performs, with an accuracy greater than 0.8 and a precision greater than 0.9 in both datasets. In P3-2, there are other situations to be taken into consideration. There are the fact that a person can stop moving and has still to be tracked, and the fact that, in crowded environment, pedestrians trajectories intersect can be a problem for the tracking. The detection module uses moves and it fails in such situations. The tracking module has to be robust. An accuracy of 0.74 is suitable and the precision of 0.98 is definitely great. And then, in H3-2, cars move are not the same. Some variations in the cars trajectories are observed and have caused a decrease of the accuracy and precision of the model against in H3-1.

6.3.2 Other measures of the performance of the model

The accuracy and the precision of the model are not the only interesting measures. Before analysing the computational performance of the program, measures of the true positive rate and the true negative rate are presented in figure *6.11*.

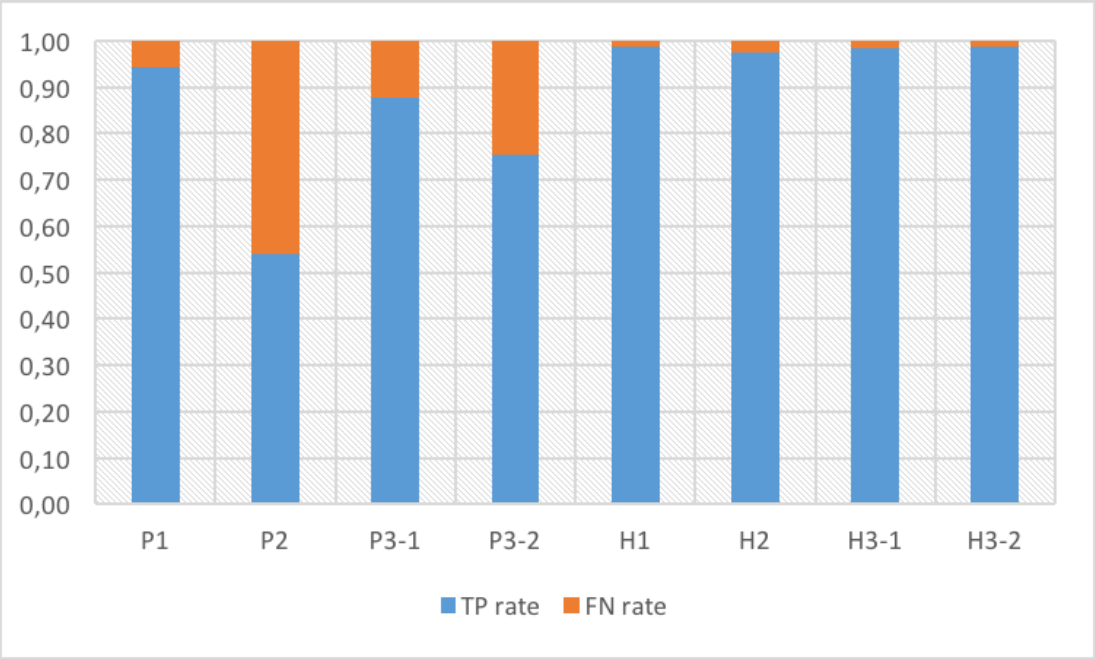

Figure 6.11: Measures of the true positive rate and the false negative rate of the model on various scenes. X-axis : video scene reference (tables *6.1* and *6.2*). Y-axis : measure of the true positive and false negative rates.

From the figure above, the first conclusion is that there is a large diversity of the results between the model applied on *PETS2016* and *HIGHWAY*. In the second situation, the measures are equivalent.

Secondly, given the measures on P2, one main weakness of the system is that it is really affected by darkness.

Thirdly, as mentioned in section *6.3.1* and confirmed by the figure above, moves diversity has disrupt the stability of the model. In *HIGHWAY*, cars follow the road and can only change lanes. By contrast, pedestrians can walk wherever they want in the station in *PETS2016*.

Another experiment has been made to obtain information about the computational aspect of the application. The figure *6.12* shows the processing times of the different video scenes.

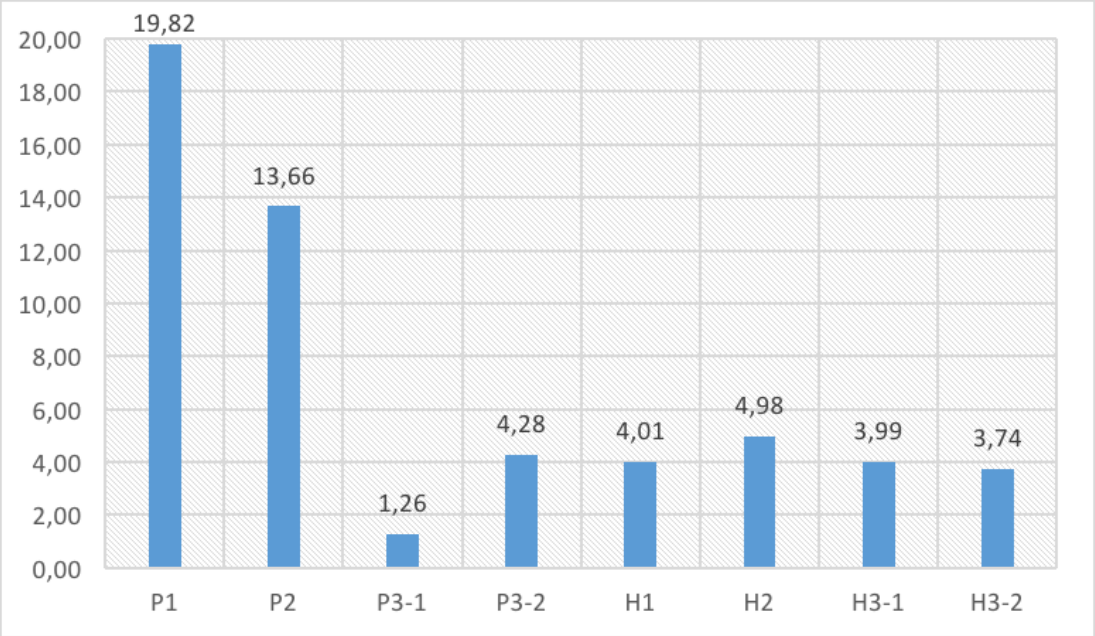

Figure 6.12: Measures of the processing speed of the model on various scenes. X-axis : processing speed in Frame Per Second. Y-axis : video scene reference (tables *6.1* and *6.2*).

Another time, there is a variety of the results between the *HIGHWAY* scenes, that have homogeneous processing speeds, and the *PETS2016* scenes, that have very disparate processing speeds. As one might expect, the number of objects of interest to track by detection causes a drop in the processing speed. Recording conditions affect the performance of the model but it is tough to conclude on their effects on the speed though.

6.3.3 Comparison with the literature

The aim of this subsection is to compare the performance of the system implemented in this dissertation with the model from [1]. Indeed, it is the base of the dissertation and the comparison permits to conclude on the choices made from the subject of the project.

The implemented model in [1] has been tested on a variety of datasets, as presented

in section *4.1.1*. ETH and i-LIDS deals with similar data (pedestrians) as the data that have been used in this dissertation.

Even if both models are not tested with the same datasets, the comparison of the performance is made on the fact that the aim is to do tracking by detection on pedestrians. Consequently, the performance of the model implemented in this dissertation is the mean measures from *PETS2016* dataset only. Results are presented in the figure *6.13*. From this figure, it is possible to conclude that the system implemented in

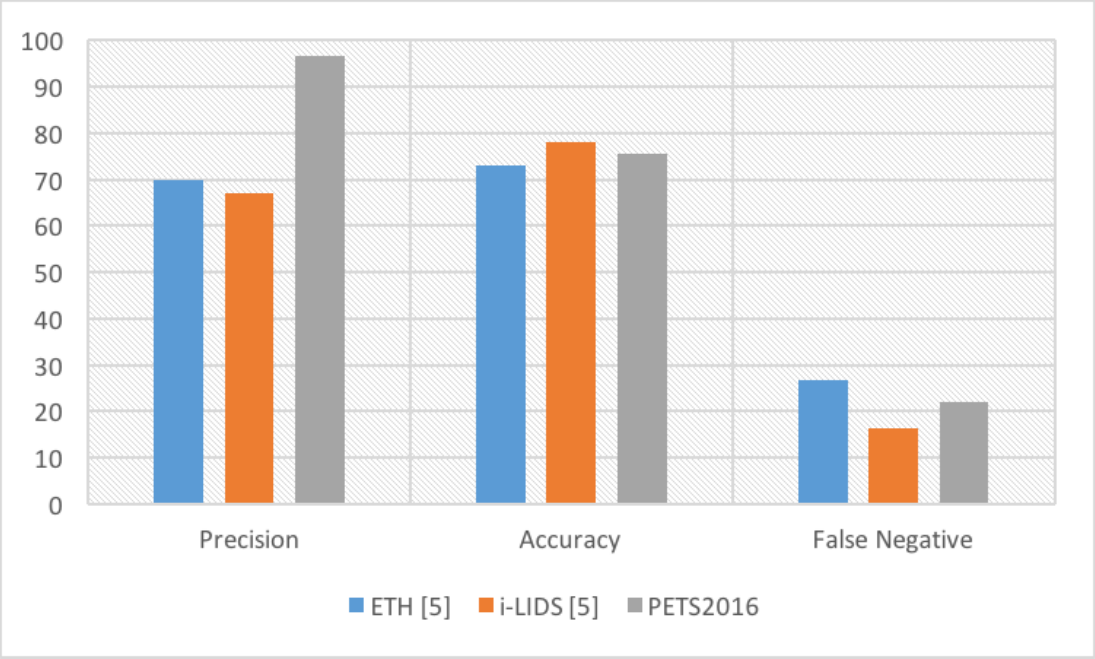

Figure 6.13: Comparison of accuracy, precision and false negative rate of the models. X-axis : name of the measure. Y-axis : value of the measure, in percent.

this dissertation has a better precision than the model in [1]. The accuracy and the false negative rate are equivalent in both models. However, it is a great result because none information of the object to track has been used in this dissertation, unlike in the detection module from [1].

A comparison with [1] using a datasets of cars could be interesting. However, the lack of model of the object to track permit to draw some conclusions about the robustness of the system implemented in this dissertation.

Finally, the last comparison with the literature is the processing speed of the algorithm. [1] precised that ”the runtime of our unoptimized code is 2-0.4 fps”. Again, it is tough to prefer one model against the other because experiments have been made

by using different computer hardware. In this dissertation, results have been obtained with an *Intel Core i5 2,7 GHz*.

As can be understood in this chapter, the experiments have demonstrated some advantages and drawbacks of the achieved implementation. A criticism of it is presented in the next chapter.

Chapter 7

Discussion

In this chapter, the strength and the weakness of the project are explained. The first part concerns the quality of the data, the second part focuses on the improvement of the speed of the program and the last one on the improvement of the use of the application.

7.1 Quality of the data

The data have a great quality. Their origin and the fact that they are used by scientists to design computer vision applications prove it. Moreover, the comparison with well-known datasets already tested on other articles is a strong base.

One idea was to use this application on a personal dataset, meaning recorded by myself and with casual material. The fact of not having ground truth is the main problem even if the idea is interesting.

7.2 Data Model

The modelling of the structure of the data stored by the program has been a huge challenge. Even if it is done in basic Python, the use of *dictionaries* and the fact of being careful to the speed of the program have permitted to obtain a fast program.

A feasible solution is the use of *dataframe*. This data structure, from the *Pandas* library, is a two-dimensional table with labelled rows and columns and can be seen as similar to *dictionaries* [20]. For the programmer, it is easy to access to the data stored with the labels. For the user of the application, the performances are improved because *dataframes* are made to deal with a high quantity of data.

7.3 User interface

The basic way of improving the use of the application is the fact of creating a graphical user interface (GUI). The environment *PyQT* can be used to create a ready to use application, designed for users.

7.4 Recording conditions

The system implemented in this dissertation has performance that is related to the detection module output. After different experiments, in section *6.3*, recording conditions notably affect this system. In the datasets *PETS2016* and *HIGHWAY*, there are obscurity and wind.

The obscurity can be reduce by applying the algorithm presented in section *4.2* for the pre-processing of the data. With more contrast, regions of darkness are reduced.

Wind, and by extension weather effects on videos, are huge problems for computer vision scientists. Some researches are made on these subjects and, for example, [10] presents an algorithm for the pre-processing of the data for the ”detection and removal of rain from videos” [10].

7.5 Model implemented

The model designed, implemented and tested in this dissertation is satisfying in comparison with the literature and especially [1]. It can handle different situations with various kind of objects of interest and challenges object-oriented model, thanks to its performance.

The performance of the whole model is related with the detection achievement. Indeed, as has often been mentioned, its development was long because there is an ambitious to implement a module that can compete with object-oriented detection.

Some factors decrease the accuracy of the tracker : the motion of the objects and the recording conditions. The length of the video is also a factor to take into consideration. Indeed, errors made by the model can much affect its performance in long videos.

Chapter 8

Conclusion

The subject of the dissertation was to select an effective technique from the literature, implement it and evaluate the tracker's accuracy. The main constraint was the fact that the tracker does not know what it has to track.

In order to complete such a project, a large number of stages has been required. The technique selection and its deep study have been realised in the background. Other related topics have been developed, namely stochastic processes, that are essential to understand tracking by detection. Then, the design and the implementation of the application have been achieved. The experiments section permits the long process of estimating the optimal value of the diverse input parameters. Finally, a discussion about topics related to the model has been presented.

The implementation of the application was in two main modules. The detection module is responsible of finding the objects of interest in each frame of the video. It is based on the background subtraction method. And the tracking module follows the target with a prediction-correction system and uses the particle filter algorithm. Both modules are used at the same time.

The deliverables of this dissertation, mentioned in section *1.2.3*, are respected. They appear all along this report. The implemented tracker is performing and adapts well to diverse kinds of objects of interest. However, external conditions, such as the environment and the weather, reduce its performance.

Further work represents a huge challenge by the variety of applications and techniques. In one hand, artificial intelligence emerges in many domains and currently,

computer vision purposes range from autonomous vehicles to user recognition on mobile phone. In the other hand, the means have increased the possibilities. Computer hardware are less and less limited thanks to the cloud and the use of virtual machines. Powerful libraries, such as TensorFlow, have been created to efficiently process data with an API. Neural networks also represent a major progress in computer vision. Caffe, for example, is a deep learning framework and permits to obtain high accuracy in short time.

Bibliography

[1] Fabian Reichlin Bastian Leibe Esther Koller-Meier , Michael D. Breitenstein and Luc Van Gool. Robust tracking-by-detection using a detector confidence particle filter. *IEEE*, 2009.

[2] Gavin Brown. *Machine Learning*. The University of Manchester, United Kingdom, 2015.

[3] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2005.

[4] Docs.continuum.io. Anaconda distribution. https://docs.continuum.io/anaconda/, 2017.

[5] Docs.python.org. Dictionaries. https://docs.python.org/2/tutorial/datastructures.html, 2017.

[6] Aphrodite Galata and Carole Twining. *Lecture 1 : Basic Image Analysis*. The University of Manchester, United Kingdom, 2017.

[7] Aphrodite Galata and Carole Twining. *Lecture 4 : Segmentation and Clustering*. The University of Manchester, United Kingdom, 2017.

[8] Aphrodite Galata and Carole Twining. *Lecture 5 : Local Features*. The University of Manchester, United Kingdom, 2017.

[9] Aphrodite Galata and Carole Twining. *Lecture 9 : Visual Tracking*. The University of Manchester, United Kingdom, 2017.

[10] Kshitiz Garg and Shree K. Nayar. Vision and rain. *International Journal of Computer Vision*, dec 2006.

[11] Github.com. Atom. https://github.com/atom/atom, 2017.

[12] Helmut Grabner and Horst Bischof. On-line boosting and vision. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2006.

[13] Udesang K. Jaliya Himani S. Parekh, Darshak G. Thakore. A survey on object detection and tracking methods. *International Journal of Innovative Research in Computer and Communication Engineering*, Vol. 2, feb 2014. Issue 2.

[14] Michal Isard and Andrew Blake. Condensation - conditional density propagation for visual tracking. *Int. J. Computer Vision*, 1998.

[15] Jetbrains.com. Pycharm. https://www.jetbrains.com/pycharm/, 2017.

[16] Jupyter.org. Project jupyter. http://jupyter.org, 2017.

[17] Lindsey Kleeman. *Understanding and applying Kalman Filtering*. Department of Eletrical and Computer Systems Engineering, Monash University, Clayton.

[18] Stefan Roth Mykhaylo Andriluka and Bernt Schiele. Monocular 3d pose estimation and tracking by detection. *IEEE*, 2010.

[19] Art B. Owen. *Monte Carlo theory, methods and examples*. Art Owen, 2013.

[20] Pandas.pydata.org. Dataframe. https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.html, 2017.

[21] Sandeep K. Patel and Agya Mishra. Tracking techniques: A critical review. *Moving Object Indian Journal of Computer Science and Engineering (IJCSE)*, Vol. 4(No. 2), apr 2013.

[22] SceneBackgroundModelling. Dataset. www.scenebackgroundmodelling.net, 2016.

[23] Dan Simon. *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches*. Wiley, jul 2006.

[24] Jan E. Solem. *Programming Computer Vision with Python,*. Creative Commons, 2012.

[25] Fida El Baf Thierry Bouwmans and Bertrand Vachon. Background modelling using mixture of gaussians for foreground detection - a survey. *Recent Patents on Computer Science*, Vol. 1(No. 3):219–237, nov 2008.

[26] Wikipedia. Filtering problem (stochastic processes). https://en.wikipedia.org/wiki/Filtering_problem_(stochastic_processes), 2017.

[27] Wikipedia. Google drive. https://en.wikipedia.org/wiki/Google_Drive, 2017.

[28] Wikipedia. Kalman filter. https://en.wikipedia.org/wiki/Kalman_filter, 2017.

[29] Wikipedia. Mixture model. https://en.wikipedia.org/wiki/Mixture_model, 2017.

[30] Wikipedia. Monte carlo method. https://en.wikipedia.org/wiki/Monte_Carlo_method, 2017.

[31] Wikipedia. Normal distribution. https://en.wikipedia.org/wiki/Normal_distribution, 2017.

[32] Wikipedia. Particle filter. https://en.wikipedia.org/wiki/Particle_filter, 2017.

[33] Wikipedia. Precision and recall. https://en.wikipedia.org/wiki/Precision_and_recall, 2017.

[34] Wikipedia. Trello. https://en.wikipedia.org/wiki/Trello, 2017.

[35] Krystian Mikolajczyk Zdenek Kalal and Jiri Matas. Tracking-learning-detection. *IEEE Transactions On Pattern Analysis And Machine Intelligence*, Vol. 34(No. 7), jul 2012.

[36] Zoran Zivkovic and Ferdinand van der Heijden. Efficient adaptive density estimation per image pixel for the task of background subtraction. *Pattern Recognition Letters*, 2006.