UCL Université catholique de Louvain

École polytechnique de Louvain (EPL)

Multiple object tracking combining camera and radar

A real-time solution

Dissertation presented by
Christophe DELEVAL , Maxime LAHY

for obtaining the Master's degree in
Mechanical Engineering

Supervisor(s)
Christophe CRAEYE, Benoît MACQ

Reader(s)
Philippe CHATELAIN, Christophe DE VLEESCHOUWER , Jean LÉGER

Academic year 2016-2017

Acknowledgements

It has been a great pleasure to work on a year-long master thesis at the Université catholique de Louvain, and more precisely, in the Ecole polytechnique de Louvain. Many great moments are to be remembered.

We would first like to express our sincere thanks and our deepest appreciation to both our supervisors, Prof. Benoît Macq and Prof. Christophe Craeye who proposed the subject of this thesis and who guided us throughout the academic year. It was a great honour to work under their supervision.

We would then like to express our gratitude towards Jean Léger for his insight all along the master thesis.

We are also extremely thankful to Prof. Christophe De Vleeschouwer for his indications and precious advice in the field of object detection and pedestrian tracking.

We are also grateful to Thomas Feuillen and Thomas Pairon for their help concerning the selection and use of the radar.

Contents

List of abbreviations and symbols

List of abbreviations

ConvNets	Convolutional Neural Networks
FFT	Fast Fourier transform
FPGA	Field-Programmable Gate Array
fps	Frames per second
GUI	Graphical User Interface
HOG	Histograms of Oriented Gradients
HSV	Color space represented three components: Hue, Saturation, and Value
ICF	Integral Channel Feature
IR	Infrared
JMPD	Joint Multitarget Probability Density
LBP	Local Binary Patterns
LED	Light-Emitting Diode
LUV	Color space represented by three components: $(L*, u*v*)$
PDF	Probability Density Function
RGB	Color space represented by three components: Red, Green, and Blue
SVM	Support Vector Machine

List of symbols

$(p_x; p_y)$	Positions on the image plane along x and y axis	$[pixels]$
(v_x, v_y)	Speed of a particle in the 2D space	$[m/s]$
(x, y)	Position of a particle in the 2D space	$[m]$
α	Parameter defining the covariance of the PDF used at the prediction step ($\boldsymbol{\sigma}_p^2(\Delta T)$)	$[-]$
α_o	Offset angle	$[deg]$
$\boldsymbol{\sigma}_c^2$	Covariance matrix of the camera error model	$[-]$

$\boldsymbol{\sigma}_p^2(\Delta T)$	Covariance matrix of the prediction step	$[-]$
$\boldsymbol{x}$	State of a particle	$[m; m/s]$
ΔT	Time difference between two updates of the particle filter	$[s]$
δ_h	Ratio between two consecutive bounding boxes' heights	$[-]$
δ_t	Maximal time limit between two camera updates before removal of a particle filter	$[s]$
γ_1	Variable quantifying the angle error of the camera detection	$[-]$
γ_2	Variable quantifying the range error of the camera detection	$[-]$
λ	Wavelength	$[m]$
ϕ	Phase shift	$[deg]$
σ	Standard deviation	$[-]$
c_0	Speed of light in vacuum	$299,792,458\,[m/s]$
C_k	Mean colour vector representing the appearance of a pedestrian k	$[-]$
c_{th}	Colour difference threshold	$[-]$
$Col_{diff}^{a,b}$	Normalised euclidean distance between two mean colour vectors C_a and C_b	$[-]$
D	Range of the target	$[m]$
d_o	Mean deviation between the camera and the radar used for calibration	$[pixels]$
f_b	Frequency difference between the sent and received signal (adjusted for the Doppler shift)	$[Hz]$
f_d	Frequency shift due to the Doppler effect	$[Hz]$
f_M	Frequency difference between the highest and the lowest frequencies emitted by the radar	$[Hz]$
I_r	Intensity of radar target points	$[-]$
N_o^i	Number of observations at iteration i	$[-]$
N_p	Number of particles	$[-]$
T_M	Time taken for one period of the sent signal	$[s]$
w_n^i	Weight of particle n at iteration i	$[-]$
$(\theta_x; \theta_y)$	Azimuth and elevation angles	$[deg]$

Abstract

As the demand for accurate real-time multiple object tracking methods increases, the idea of developing a device combining multimodal sensors is proposed. This master thesis proposes an implementation of a pedestrian tracking algorithm based on a camera and a radar and explores the benefits and drawbacks of this type of setup. The intent behind this fusion is to combine the advantages of each of the sensors in order to gain accuracy in situations deemed challenging for methods making use of only one of the two sensors. The final solution uses a state-of-the-art 2D pedestrian detector to extract target positions from images taken by the camera and signal processing to extract observations from the radar. A particle filter is used to combine both information streams by primarily combining the azimuth angle as computed from camera detections with probable ranges observed by the radar. Its accuracy and precision are tested through a series of challenging tests. From the results, it is concluded that, while the camera offers accurate detection, angular localisation and appearance recognition it struggles with too large distance variations, which is why, in these situations, the use of a radar can be highly beneficial. Further improvements to the current method are also suggested for future works.

Keywords Multiple Object Tracking, Particle filter, Multimodal sensors, Data fusion, Pedestrian detection, Camera, Radar

List of Figures

List of Tables

Chapter 1

Introduction

This first chapter starts by introducing the context that this master thesis is based on. The main objectives set for this work as well as the approach that was taken to achieve these goals are then mentioned. The contributions that this study adds to the existing state of the art are also expressed and, lastly, the organisation of this master thesis is put forward.

1.1 Context

In recent years, the rapid development of artificial intelligence and the desire to build machines capable of interacting with the human world has lead to intensive research in the field of autonomous spacial awareness. One of the major aspects that has particularly been worked on is real-time multiple object tracking, which, based on one or more sensors, aims to determine the trajectories taken by specific types of targets and estimate their positions.

Currently, most of the research that is being conducted is based on monocular vision (a single camera) but a number of works exploring different combinations of sensors can also be found. In each scenario, the sensors that are used, and the tracking methods that are implemented largely depend on the type of target that is tracked, the environment in which they move, and how the sensors can be placed to gather information on the environment.

Generally, pedestrians are one of the most popular types of targets around which research is carried out, because being able to track reliably people can have huge impacts in a large number of domains. As such, this thesis is solely based on pedestrian tracking methods. With this choice comes the logical but crucial assumptions that people only move on the ground and that their trajectories can be computed using two dimensions along a surface (the ground). These assumptions are the reason why sensors capturing 2D data on the environment (such as cameras) can be used to perform 3D tracking.

As popular as they are, vision based pedestrian tracking algorithms suffer from a major limitation. Due to the fixed amount of pixels that can be captured in a frame, the distance at which targets are located from the sensor will play a large role on the precision of the information that can be determined. In fact, the amount of pixels representing a same target will decrease quadratically as the distance between the sensor and the target grows [11]. Because the less pixels are used to represent a target, the less data there is to work with, large distance variations can cause monocular vision based pedestrian tracking algorithms to be more susceptible to errors.

On the other hand, radars can also be used for tracking purposes. Although these devices can handle large distance variations with less errors than cameras, they have other flaws. The first difficulty comes from differentiating a proper pedestrian from other moving objects or noise

that is picked up by the sensor. Additionally, even if a target is correctly identified, the next big challenge concerns differentiating two targets from one another. While cameras can be used to compare appearances such as colour and other defining features, radars have much less information to work with to determine which detection belongs to which target.

For these reasons, following the ideas of previous researches [55], [61], [101], the idea came to combine a camera with a radar and analyse how this combination of multimodal sensors performs in situations considered as tricky for tracking methods using only one of the two. The intended purpose of this combination would be to rival, and hopefully surpass, current single-sensor methods, and, in order to do so, have a relatively cheap final product in order to potentially commercialise it.

A scenario deemed challenging for single sensors is one in which they are placed relatively close to the ground and track multiple targets five to fifty meters away walking in intricate patterns. This would make any monocular vision based tracking algorithm struggle by having to deal with a wide distance range, while targets walking close to each other would be very difficult to handle with just a radar. Therefore this thesis is focused on this specific type of scenario, while extensive testing in multiple other contexts is left to future works.

1.2 Objectives and approach

The aim of this thesis is to develop a prototype device composed of a radar and a camera that can track multiple pedestrians in real-time and analyse its efficiency as well as the accuracy of the trajectories it computes. It is hoped that the lessons learned from this first prototype development can be used for future works on this subject, and lead towards the conception of a low-cost product that would be more reliable than similarly priced monocular based competitors. In order to achieve this goal, the following methodology was used;

- The first step consists of the selection of the software platform and the hardware equipment that are going to be used to perform the tracking. This involves choosing the appropriate criteria that have to be met by both appliances and comparing different possible choices.
- Once the hardware is selected, an extensive research is carried out on how compatible data can be extracted from both sensors as well as how to fuse it. Possible tracking methods for this chosen type of data are then explored.
- After considering every possible combination of methods, the most suitable solution is chosen according to its potential performances and a first version of the algorithm is then implemented as the theory suggests.
- A thorough analysis of obtained results is then carried out and improvements are proposed. A second version of the algorithm is then implemented following these suggestions.
- At last, a final analysis of the performances is carried out, and, once more, ameliorations are suggested for future works.

1.3 Thesis contributions

This thesis contributes to the state of the art in two ways.

Firstly, it proposes a new method of multiple target tracking using a camera and a radar as multimodal sensors. The data from the radar is processed using a method developed by RFbeam

[90] while pedestrians' locations are extracted with a pedestrian detector developed by Dollár, Belongie and Perona from images captured by the camera. The data is fused using methods described in [70] and [32]. The pedestrian tracking is originally based on a particle filtering method described in [55] and uses a resampling method inspired by [62] and [1].

Secondly, an analysis of the capabilities and shortcomings of such a tracking method is made and scenarios in which it performs best are identified. Possible improvements are also given, as well as probable solutions to its current weaknesses. This research can be used as a basis for future works that intend to improve the performances reached by the current tracking method developed in this thesis.

1.4 Thesis organisation

The thesis is organised as follows. The device that was developed and the sensors that are employed are described in Chapter 2, along with a justification for their selction. The way usable data is extracted from these two sensors and the way their information is fused is discussed in Chapter 3. Chapter 4 contains a state of the art of the different multiple object tracking methods that were considered. In Chapter 5, the preliminary choices and assumptions that were made before starting the implementation are described. The tests that are chosen to quantify the quality of the algorithm are also detailed in this chapter. The first basic implementation of a single target tracking method, as described by the state of the art, are explained in Chapter 6. This first version already combines data from both sensors and serves as a basis on which new functionalities can be tested and added. The results obtained by this algorithm are also expressed and discussed. An explanation and a discussion on the final implementation of the multiple target tracking algorithm can be found in Chapter 7. This chapter also contains explanations on how the results from the previous version influenced the new choices that were made. The final results are displayed, discussed, and compared to the state of the art in the same chapter. Chapter 8 contains the list of the different limitations that were thought to be the cause of certain shortcomings discussed in the previous chapter as well as proposed ways to fix these issues. Possible applications for such a device and tracking algorithm are also discussed in this chapter.

Chapter 2

Device description

In this chapter, the physical setup used for this project will be described and the choices made will be justified. As stated earlier, the goal of the project is the conception of an integrated device combining a camera and a radar aiming at tracking pedestriansn with one of the main requirements for the equipment being that the final product has to cost less than €100 to be potentially commercialised.

For this first prototyping iteration, the camera and the radar are simply screwed on a wooden platform, shown in Figure 2.1, allowing different relative positions to be easily tested. A more sturdy and efficient platform can custom made for future iterations of the project but in this first phase, the wooden platform was a viable solution.

The sections below goes over the selection of both devices. Firstly, the radar will be described as well as the justification of its choice. Secondly, the same will be done with the camera. Thirdly and finally, a brief section will be dedicated to the calibration between the camera and the radar.

2.1 Radar

In order to achieve the goals of this thesis, an appropriate radar must be selected. Following these goals certain criteria were defined in order to choose the best suited device for this project

Firstly, and most obviously, the device has to be able to deal properly with humans as targets. This means that it has to be sensitive enough to capture reflections caused by people and it will have to be able to detect objects moving at human speeds (from 0 to $5m/s$).

In addition to that, another requirement is to compute the trajectories of pedestrians in an area as large as possible. If the radar is not placed too high above the ground, the azimuth angle range along with the detection distance range will be the main defining elements of the observation area, while the elevation angle range will be less crucial. This is because, in order to detect a person, not the whole body has to fit in the radar's field of view.

Finally, while the output form cameras are usually quite similar to one another, in that they all capture images, depending on the type of radar that is chosen, the outputs form radars can be radically different. The simplest and thus cheapest radars on the market output a continuous analog signal that corresponds to the voltage across the receiving antenna ports. This signal has to be transformed into digital data in order to be compared with the signal sent by the emitting antenna. Because signal processing was not the main focus of this thesis, it was decided that it would be preferable to find a radar unit that would perform these computations internally, even though it would be more expensive. This means the main part of the project could be focused on, without having to spend time on the implementation of a solution for signal processing. Furthermore, the format of the data is not the only difference between radar outputs. The

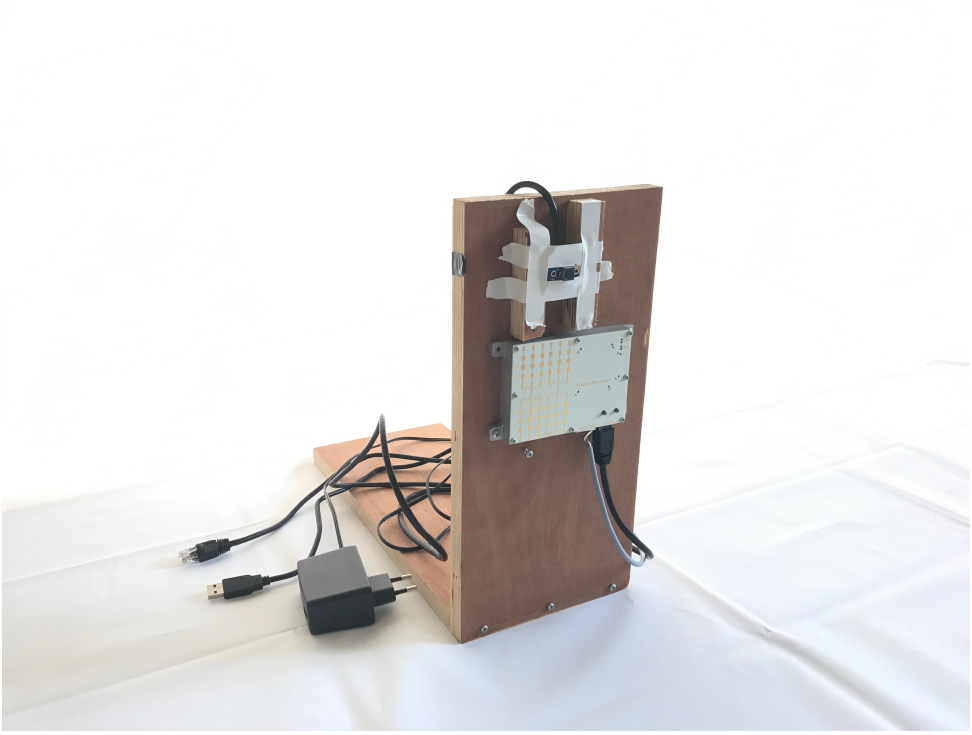


Figure 2.1: Picture of the device that was used for this thesis

simplest and cheapest devices, depending on their characteristics, can sometime only be used to compute speeds, while some more complex and expensive ones can output much richer data that can lead to 3D positioning of the targets. For research purposes, it was deemed wiser to opt for a radar that could retrieve a maximum of information and assess the need for these capabilities after the tests' results analysis.

The product that corresponds the best to this thesis' and that could be found is the K-MD2 radar from RFbeam [90], whose characteristics are listed in the Table 2.1 below.

Frequency range:	$24000 - 24250GHz$
Output power:	$+20dBm$
Number of emitting antenna(s):	1
Number of receiving antenna(s):	3
Output (Ethernet):	Target list (speed, distance, azimuth and elevation angles) for up to 40 objects
Targets:	Pedestrians/Cars
Detection distance:	$300m$ for cars, $100m$ for people. Resolution: $0.7m$
Angle range:	$\pm 10°$ (elevation), $\pm 15°$ (azimuth). Resolution: $0.1°$
Target maximal speed:	$+128km/h$. Resolution: $1km/h$
Price:	€1100

Table 2.1: Characteristics of the chosen radar

This device is the best match for the means of the project. Its three receiving antennas and its capability to modulate frequencies ensure that the targets' positions in the 3D space, as well as their radial speeds can be computed. As desired, this device performs all the signal processing as a black box and outputs all this information directly to a computer through an Ethernet connection at a frequency rate of $10fps$. A MATLAB code is also provided as to control its

parameters. This greatly simplifies the gathering of radar data compared to other devices.

As seen in Table 2.1, the K-MD2 has an azimuth angle range of 30°, which was the largest that could be found for devices of this type. A claimed resolution of 0.1 degrees is sufficient for tracking because, at the claimed range that this device has, that would represent a maximal error of less than $5cm$.

As for the aforementioned range at which it can detect targets, 100 meters is deemed enough for this thesis' objectives. In addition to that, a claimed resolution of $0.7m$ is good enough to track multiple targets close to one another without risking an overlapping of observations.

The price of this device is the only element that does not conform with the desired specifications. Coming at €1100, it is way above the price range that is intended for the finished product. However, a cheaper single purpose device can be purchased in the future following the requirements assessed through testing and the complex K-MD2 device can then be reused for other educational purposes.

The MATLAB code that came with the K-MD2 radar can be interacted with via a graphical user interface (GUI), as shown in Figure 2.2. Many options and commands are available, but for the purpose of this thesis, the "Azimuth Angle Map" with the option "Display Static Objects" unticked was mainly used. With this last option activated, too much noise is present, as depicted in Figure 2.3.

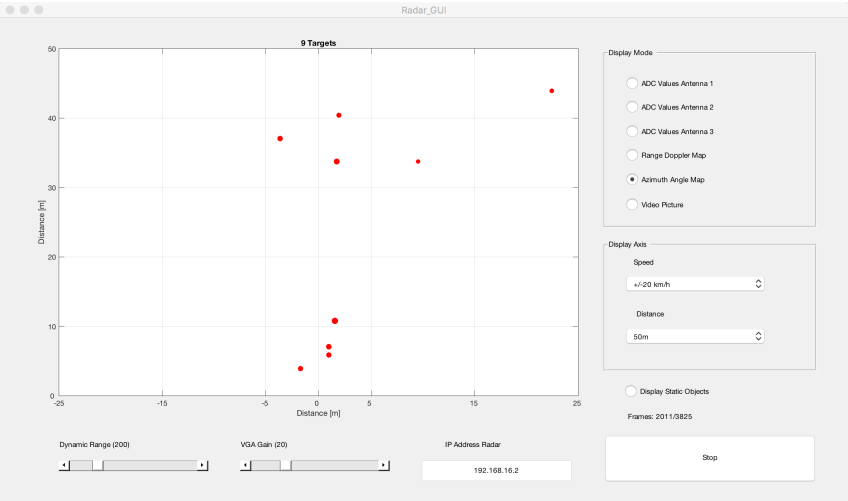


Figure 2.2: GUI of the K-MD2 radar provided by RFBeam. "Display Static Objects" is unticked. The radar is currently pointing statically towards the sky.

In order to use the content of this software without having to go through the GUI, RFbeam's original program had to be decomposed. The connection between the radar and the computer is established by TCP/IP and is initiated when the program starts running. The portion of the code handling this communication protocol was extracted and used in a new MATLAB function whose purpose is to acquire data from the radar and to relay it to the rest of the tracking algorithm.

2.2 Camera

Choosing the correct camera is important for this project as it will have to be extensively used. For its selection, three main criteria are held into account. Firstly, as with the radar, it has to be as simple to use as possible. Secondly, it has to be capable of outputting images with a high enough quality for long range detections but low enough resolution so computation times do not

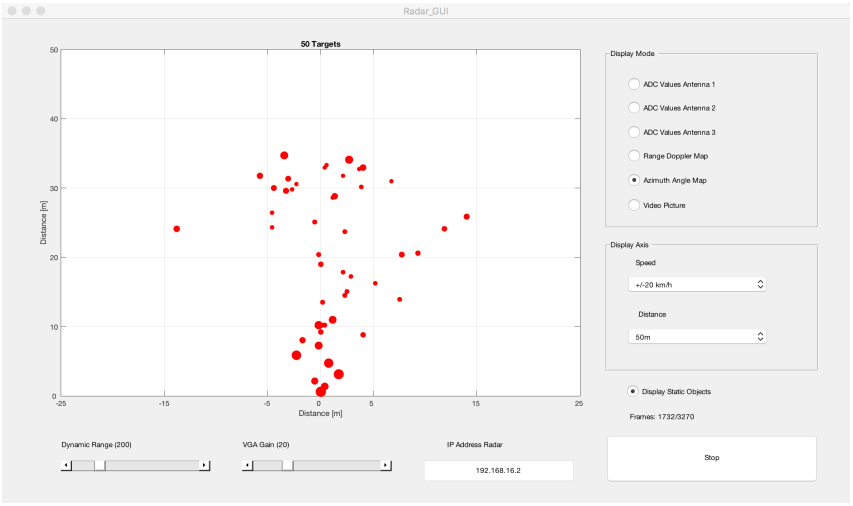


Figure 2.3: GUI of the K-MD2 radar provided by RFBeam. "Display Static Objects" is ticked. The radar is currently pointing statically towards the sky.

Sensor:	1/4" CMOS OV5640
Resolution:	1280×1024 pixels
Angle of view:	$60°$
Autofocus:	Yes
USB protocol:	USB2.0 HS/FS
Auto exposure AEC:	Yes
Auto white balance AEB:	Yes
Adjustable parameters:	Brightness/contrast/color saturation /definition/gamma/WB
Voltage:	$5V$
Current:	$150mA$
Price:	€50

Table 2.2: Characteristics of the chosen camera.

skyrocket. Thirdly, its field of view has to be as close to that of the radar as possible.

For programming and testing computer vision algorithms, USB camera modules are the easiest solution because of their simplicity of use and connectivity. According to the requirements, it was deemed that resolutions between $720p$[1] and $1080p$[2] were a good compromise between a good quality and a low computation time. Considering acceptable shipping time according to the delays, the closest angle of view to the radar's ($30°$) that could have been chosen is $60°$. It was thus determined that the USB camera module whose characteristics are given in Table 2.2 was the best candidate for this project's purposes.

The output of this camera is thus 1280×1024 pixels frames which can be extracted with MATLAB. These frames are then fed to a pedestrian detector in order to locate pedestrians on the image. This step will be described later on.

[1] 1280 pixels wide and 720 pixels high
[2] 1920 pixels wide and 1080 pixels high

2.3 Calibration

As the prototype platform on which the camera and the radar are placed is not ideal, both devices are not perfectly aligned, there is thus a possible offset angle in the azimuth plane between the centre of the vision of these two sensors and it has to be taken into account. This calibration has to be done regularly if the device is disassembled between uses. It is done by detecting an oscillating[3] target with the radar and adjusting its position until the mean radar detections are exactly centred at 0°. A snapshot is then taken with the camera and the amount of pixels between the centre of the target and the centre of the image is computed. As to be as accurate as possible, this computation is performed multiple times with the target at different ranges so a mean deviation d_o, in pixels, can be computed. The offset angle can than be determined using the focal length f (which is in pixels as well) as follows:

$$\alpha_o = \arctan\left(\frac{d_o}{f}\right). \tag{2.1}$$

This value is then saved for future use. Every image detection will thus be adjusted according to the value of α_o.

[3]The target has to be oscillating towards and away from the radar as to stay always at the same azimuth angle.

Chapter 3

Extracting and fusing similar sensor data

When dealing with multimodal sensors, the type of data they can retrieve will influence how they will be used by any algorithm attempting to combine them. Depending on the richness of their outputs, some elements from different sensors might be comparable, and others, complimentary. The aim of this thesis being to analyse the effect of both sensors on a same task, namely multiple object tracking, the first important task is finding a common format in which the target's information will be stored. This also means finding a way to convert the data from both sensors to that format. The decisions made at this stage are important as they define the initial direction in which the work will progress. Indeed, the data fusion module completely depends on the type of data (and its related error) that it is being fed.

In this chapter it is shown that an estimation of the 2D position of a target on the azimuth plane can be extracted from both sensors hereby leading to the choice of representing targets by their position in this space. The methods that can be used to transform the original sensor's outputs to this common format will be discussed below, starting with the camera before moving onto the radar. The different possible data fusion architectures and approaches concerning this particular type of data will then be described.

3.1 Locating a target using a camera

The camera plays a crucial role in the tracking algorithm. Along with a well performing detection algorithm, it brings information on the position of a person on the image, its size in pixels and its appearance. These elements can be used to determine a detected pedestrians angle from the centre of vision, as well as estimate their distance. The pixel representation of a pedestrian can also be helpful when it comes to distinguish them according to their appearance.

Because the goal is to track humans, the main focus of this chapter will be set on the pedestrian detection algorithms. First, a state of the art of the currently best performing detection methods will be presented. This will be followed by the approaches used to estimate the ground position of a person with its detection.

It should be noted that target detection algorithms are usually always linked to camera-based tracking algorithms. However, these two steps will be discussed separately as in this case the camera will not be the only sensor whose data will be used by the tracking algorithm. This section will focus on the detection while Chapter 4 will go over the ways of tracking targets using the multimodal sensors.

3.1.1 Pedestrian detection: the state of the art

This subsection aims to present the best performing pedestrian detectors existing and explain their underlying algorithms. The researches have been based on a multitude of different papers detailing the state of the art [6], [24], [48], [36], [115], [117].

Although significant researches and advancements have been made during the last decades with steadily improving performances [6], it still remains uncertain what the exact key ingredients are for a high-performance detector [115]. However, looking at the best performing detectors from the Caltech-USA pedestrian benchmark[1], two kinds of pedestrian detectors stand out: the integral channel feature (*ICF*) detectors, and the convolutional neural networks (*ConvNets*) detectors. Both of these approaches will be described below.

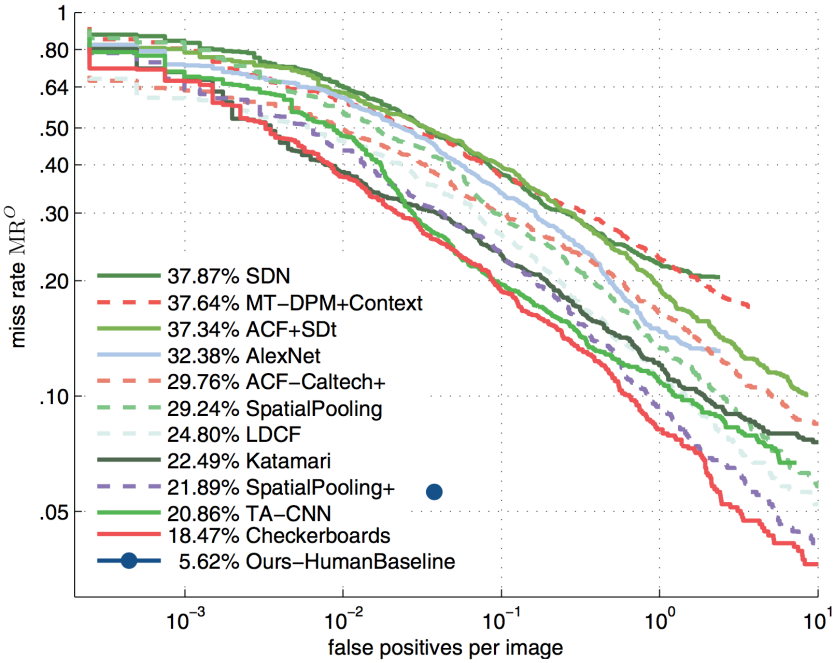


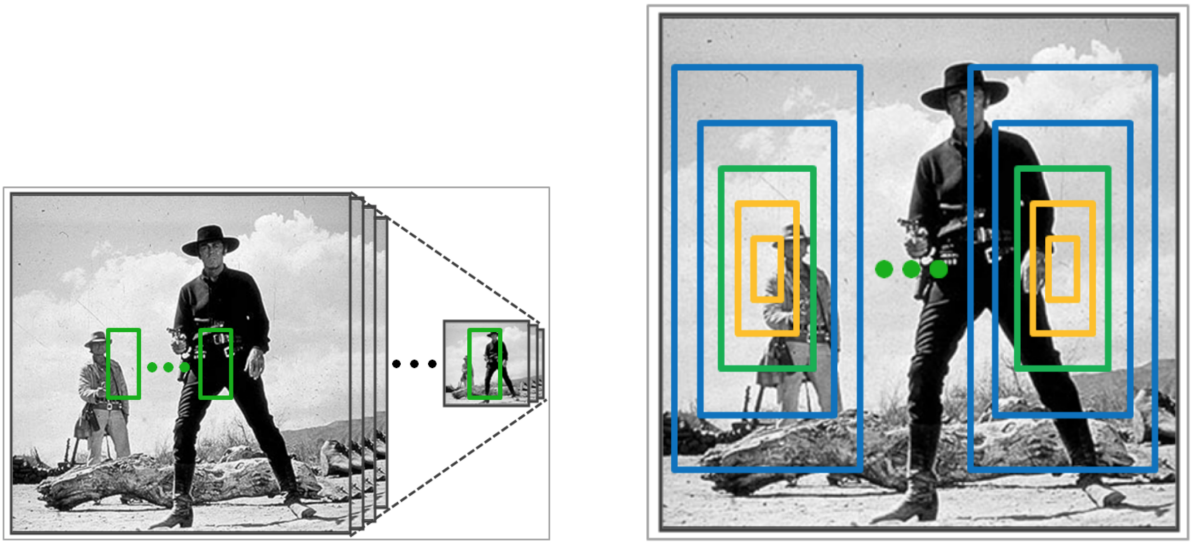
Figure 3.1: Overview of the top performing detectors from the CVPR2015 on the Caltech-USA pedestrian benchmark [117].

The structure of any pedestrian detection algorithm is mainly threefold. First, a candidate generation step is performed which selects locations in the image where it is probable to find a target. These locations are defined by rectangular boxes called "bounding boxes". Secondly, a classification operation is carried out on the generated candidates of the first step that will determine, for each bounding box, if it contains a pedestrian or not. Thirdly, a suppression step is conducted in order to remove similar pedestrian bounding boxes so only one bounding box per target remains.

[1]The Caltech-USA pedestrian benchmark consists of a database of annotated images taken in an urban environment from a car. It contains both training, and testing data sets. Approximately 250,000 frames were taken with 350,000 manually annotated bounding boxes (boxes used to indicate the locations of pedestrians in an image) from 2300 different pedestrians. The testing data aims to quantify and rank detectors in a realistic and unbiased manner. Many choices and evaluation protocols are also provided. It is one of the most used data set for detector evaluation [24], [30].

3.1.1.1 Candidate generation

The most classical and straightforward method of generating candidates consists of an exhaustive scan of the whole image with a variable-scale window (procedure called "classifier pyramid") or a variable-scale image (procedure called "dense image pyramid") [24], as illustrated in Figure 3.2. Although the dense image pyramid method performs a fast detection thanks to not having to scale the image multiple times, its performances are quite low due to the fact that very few features (that will be explained later on) are scale invariant. The classifier pyramid, on the other hand, is slower due to having to scale the image multiple times but is more accurate being able to use a greater diversity of features. It is the most popular method. The exhaustive scan being relatively slow, other faster and popular algorithms have been implemented such as *selective search* [100], *bing* [15], and *edge boxes* [120]. The choice between these candidate generation methods mostly comes down to a compromise between performance and computational speed [31].



(a) (b)

Figure 3.2: Typical exhaustive scan methods. (a) Dense image pyramid. (b) Classifier pyramid.

3.1.1.2 Pedestrian classification

The currently best performing classifiers can be divided, as stated earlier, into two groups: the *ICF* family and the *ConvNets* family.

Integral channel feature detectors The *ICF* family detectors consist of training classifiers with extracted features measured with the *integral image* method over multiple registered image channels computed using transformations of the input image. *ICF* detectors are based on the early work of Viola, Jones and Snow in [104], [102].

Registered image channels are computed with linear and non-linear transformations of the input image. These transformations can be grouped into several categories such as: gradient histograms (HOG [19]), grayscale (Haar-wavelets [103]), colour, texture (LBP [81], co-concurrence [53]), self-similarity [96], and motion [21]. Features are pieces of information (such as scalar values or vectors) extracted from an image channel and intended to be informative and non-redundant. In the case of object detection, they aim to be representative of the particular class of searched objects. Features can then be, for instance, local sums, histograms, and Haar features and

their various generalisations as in [29]. Every effective *ICF* detector uses some form of HOG (Histograms of Oriented Gradients) and the best ones use a combination of multiple features [24]. [117] shows that optical flow [86] and context information [83] are also complementary to image features and can further boost detection accuracy. However, the more sophisticated the image features become, the less useful these additional cues are [117].

The most typically used classifiers are the *support vector machine* (*SVM*) [85] and the *adaptive boosting* [103]. Both of these methods are popular choices due to their "theoretical guarantees, extensibility, and good performance", as stated in [24]. These classifiers are trained using large data sets.

- The *support vector machine* (*SVM*) learning algorithm [85] finds the maximal margin hyperplane between two classes. This maximal margin is defined by computing the smallest distance between the data points and the hyperplane [10], [16], [112]. This results in a non-probabilistic binary linear classifier. An example image is shown in Figure 3.3. The SVM can also be non-linear [93]. The advantage of this method is that the data can be of any type: scalar, vector features, etc.

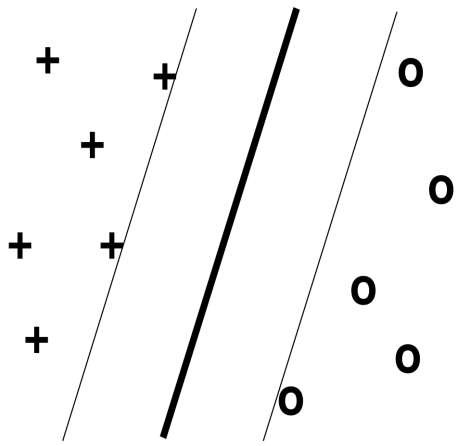


Figure 3.3: Example image of an SVM hyperplane on a 2D data set [92].

- The *adaptive boosting algorithm* [103] is a strong classifier made by combining a multitude of weak classifiers. These weak classifiers are used with different weights depending on their strengths and weaknesses in different conditions. These weights are determined iteratively during the training phase[2] of the algorithm. It has the benefit of automatically performing the feature selection (selecting the more relevant features), being able to use any type of weak classifier (*decision trees*[3] or *SVM*s for example) and having very few parameters to tune with low sensitivity to these [112], [103]. In [6], authors show that the choice of weak classifier has no significant impact on the performance but it is the choice of features that actually influence the performance.

The initial *ICF* pedestrian detector developed by Viola, Jones and Snow consists of an *adaptive boosting* classifier using features defined by sums over rectangular regions. These sums are computed using the *integral image* computation technique that will be defined later. Following this work, Dollár et al. developed the "*Integral channel feature*" in [29], [25]. It consists of an extension of the pioneer work of Viola, Jones and Snow, performing sums over multiple registered

[2]The training phase consists of running a machine learning algorithm through the whole annotated data set in order to create a model that will be used by a classifier.

[3]Decision trees are used for classification and regression. They are tree-like graphs following decision rules.

image channels that are computed using linear and non-linear transformations of the input image instead of directly performing the sums on the image. Since the work of Dollár et al. most the best performing ICF detectors are variants of their work [114], [115], [80], [84], [116].

Convolutional neural networks detectors Despite being present since the 1990s [68], *ConvNets* have only begun to become interesting in the past few years with the improvement of computational power. They are currently revolutionising the world of artificial intelligence and are taking more and more space in the image analysis domain, and more precisely in image classification [97], face recognition [98], and object detection [49]. As it can be noticed in Figure 3.1, a considerable part of the best performing pedestrian detectors are convolutional neural networks.

Convolutional neural networks also reach state of the art performance. In the other methods presented above, features are manually selected by the authors. The choice of features is thus based on their intuitions and visual cues for humans. Conversely, as explained in [99], the key idea behind *ConvNets* is to extract features out of raw image data by performing a sequence of operations such as filtering, local contrast normalization, non-linear activation, and local pooling[4]. The choice of these features only rely a training procedure, based on backpropagation[5], with large data sets, coupled with an optimisation algorithm such as gradient descent. This operation aims to determine the weights of the filters and a suitable classifier. After the training step, the lowest layers of the *ConvNets* will represent low-level features of pedestrians such as edges and details. The higher layers will be a combination of those lower layers and will begin to form more general concepts of pedestrians. The trained binary classifier will then be the last layer. Convolutional neural networks are thus very complex models with millions of fine-tuned parameters that are determined by the training phase.

3.1.1.3 Duplicates suppression

After the classification step, it is likely that multiple correctly selected bounding boxes encapsulate the same pedestrian. Hence, a duplicate bounding box suppression has to be performed. For this operation, *non-maximum suppression* methods with either the *mean shift mode* approach [18] or the *pairwise max suppression* approach [41] are mostly used [24].

3.1.2 Position estimation from bounding box

After finding the location of pedestrians in a frame, the next task is to use this information to compute an estimation of its position on the azimuth plane. There are two major ways in which this can be done . The first method consists of using visual cues to model the shape of the ground on which the pedestrians walk and establishing a transition model that attempts to link the location of their feet on this surface to a position on the actual azimuthal plane. The second method is to use the so-called "pinhole model" that, given an estimation of the target's actual height, estimates its position using the location and size of its bounding box.

Both methods depend on heavy assumptions and are usually not used in the same contexts. The pinhole model is almost exclusively used when dealing with situations with very low vantage points, like that of this project, while the first method performs best when the camera is placed higher above the targets, looking down on them. For these reasons, the pinhole model was chosen for this project and the theory behind it will be explained below.

[4]These operations are typical layers used in *ConvNets* and their explanations are out of the scope of this thesis.

[5]The backpropagation is a method used in neural networks that aims to compute the error contribution of each neuron after running the neural network on a data set.

3.1.2.1 The pinhole model

The pinhole model, shown in Figure 3.4, projects the 3D world on a 2D "image plane" by passing through a point which represents the camera lens. This means that any point on a line passing through the "pinhole" will have the exact same projection. For this representation, the centre of the image is said to be at (0;0) (in the vertical and horizontal axes respectively) and the azimuth and elevation angle of this point are also both equal 0°. From this point, the positions, in pixels, will be represented by $(p_x; p_y)$ and the azimuth and the elevation angles will be represented by θ_x and θ_y.

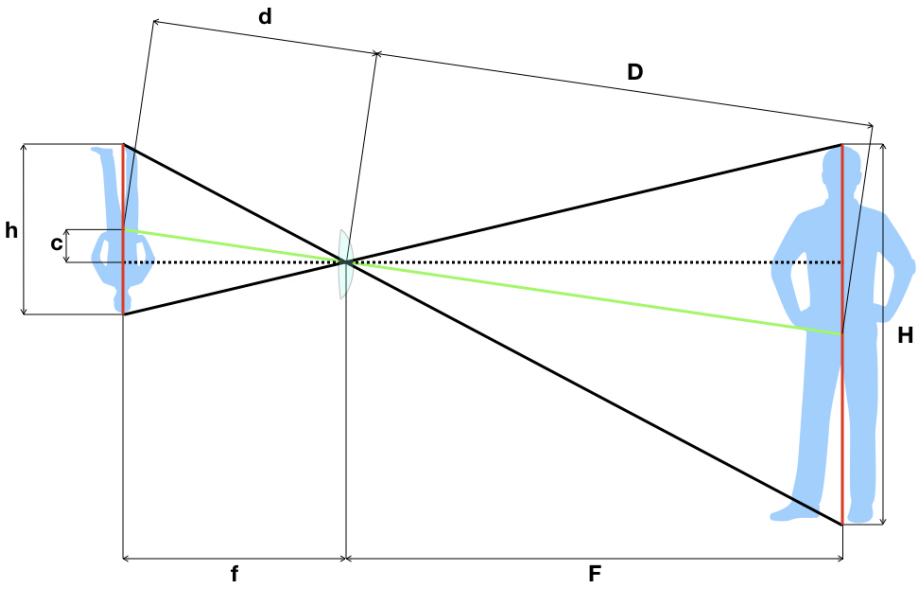


Figure 3.4: Pinhole model used to compute the distance D of a target of height H using the known focal distance f of the camera and the height h in pixels of the bounding box. F, d and c are values used to ease the comprehension of this computation.

The computations needed to obtain the distance of a pedestrian will be described below, based on the schematic in Figure 3.4. The focal length $f = 1251$ pixels has been manually computed using an object with a known size at a known distance. Using the size of that object, on the image, in pixels, the focal length could be calculated with the inverse reasoning of the following calculations.

In order to estimate the distance D of the centre of a person to the lens, it is assumed that pedestrians are always standing up perfectly perpendicularly to the horizontal plane at the height of the camera. We also assume that the distortion of the image, which is extremely low for this particular device, can be neglected[6]. The the focal length f, the height h in pixels of bounding box, and the estimated height of actual area delimited by this bounding box H are used as known variables. The variables d and c used in the following formulas are defined in Figure 3.4. First of all, the variable h is computed with the Pythagorean theorem as

$$d = \sqrt{f^2 + c^2}.$$

[6]If an other camera is used, it should be calibrated so the distortions can be accounted for

Next, using Thales' theorem, the following equation can be extracted

$$\frac{\frac{h}{2}}{d} = \frac{\frac{H}{2}}{D}.$$

Finally, the distance is estimated by

$$D = \frac{H \cdot d}{h}.$$

Once the distance D has been estimated, the azimuth angle can be used in order to compute the position estimation (x, y) in the 2D plane. First, the azimuth angle θ_x is determined by

$$\theta_x = \arctan\left(\frac{p_x}{f}\right).$$

Then the position point (x, y) in the 2D plane is determined by

$$(x, y) = (D \cdot \sin(\theta_x); D \cdot \cos(\theta_x)).$$

3.2 Locating a target using a radar

The K-MD2 device used for this work is expensive and complex and performs the signal processing internally. The output from this sensor, as shown in Figure 3.5, is an array of 2D points representing detected moving targets, as well as the intensity of the signal that bounced off them.

The fact that the signal processing is already implemented means that more time can be spent on the tracking itself. However, given that the long term aim of this project is to use simple and cheap sensors, this processing will have to be taken care of later on. This is the reason why the theory behind it will be discussed here-under.

3.2.1 Target detection

The target detection procedure will be explained below and based on the radar's block diagram shown in Figure 3.6. In order to gather the desired data, the device emits radio waves at known frequencies with the emitting antenna TX in order to analyse the waves reflected by illuminated objects. As it can be seen in the radar's block diagram, analog signals containing the information on voltages on each receiving antenna (RX1, RX2, and RX3) at any given point in time is retrieved. These signals are first converted to a digital format before being sent to a field-programmable gate array (FPGA). Once the FPGA has received the digital data, some additional computations are needed to retrieve information on the different targets. These different steps will be explained in the following sections.

3.2.1.1 Extracting frequencies and phases from raw data

Firstly, in order to gather useful information about possible targets, the different frequencies that are making up the received signal as well as their phases need to be retrieved. This is done by applying a fast Fourier transform (FFT) to the signal intercepted by all the antennas.

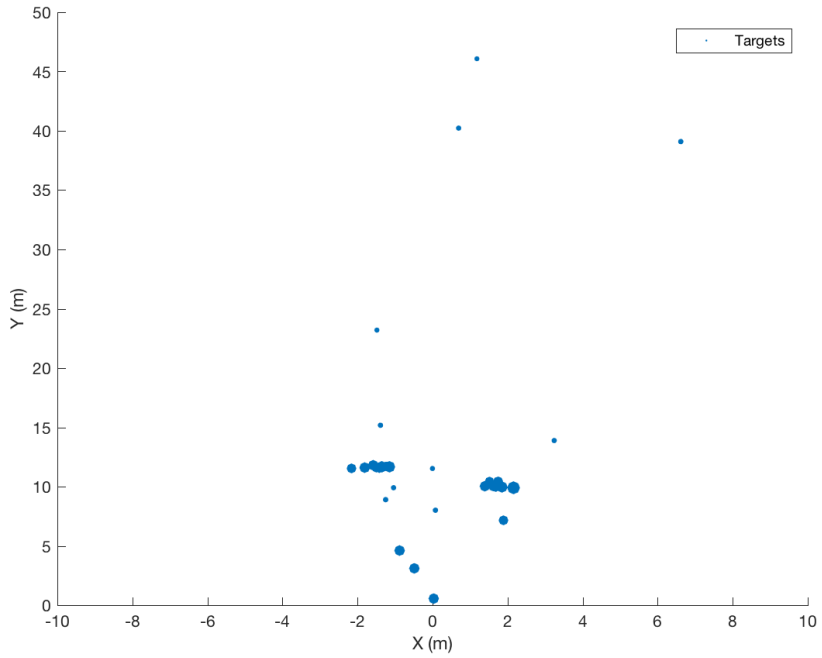


Figure 3.5: Output of the radar's "Azimuth Angle Map". Here, two people are walking side by side: one at approximately $(-1.5; 12)$ meters and the other at roughly $(1.8; 10)$ meters. The radar is at human height and directed towards an empty field. The rest of the points correspond to noise.

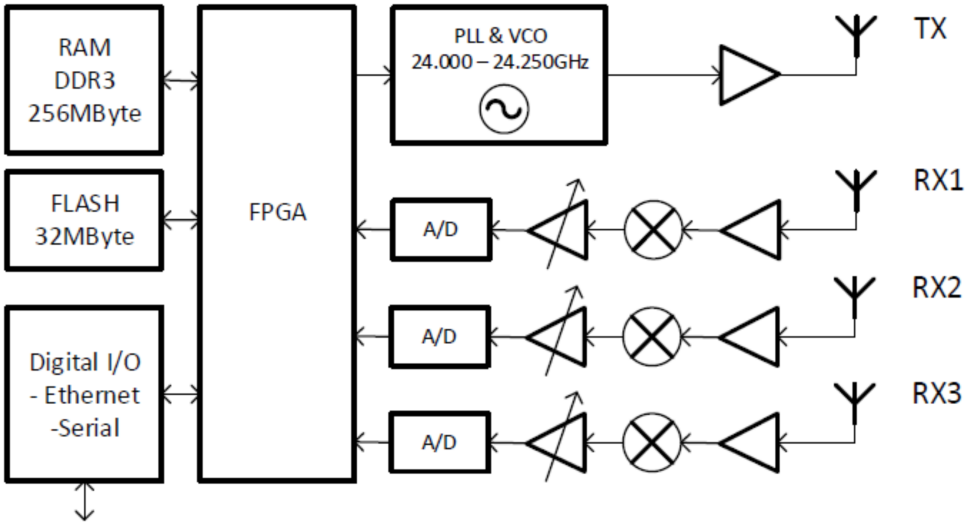


Figure 3.6: K-MD2 radar's block diagram [90].

3.2.1.2 Speed measurement

The Doppler effect is used by the radar to measure the speed at which the targets move relative to the device. The phenomenon that it is based on occurs because, when an object moves towards a radar that emits at constant frequency, it encounters more wave fronts per second than what the source initially emits. Conversely, the opposite happens when the object moves away from the source. By analysing the frequency difference (f_d) between the emitted (f_0) and the received frequencies, the relative speeds (v) of different targets with respect to the source can be computed.

The original formula for the Doppler shift is given by

$$f_d = 2v\frac{f_0}{c_0 - v}, \tag{3.1}$$

Because in this application $v << c_0$, we can say that $c_0 - v \approx c_0$, thus rewriting Equation (3.1) as

$$f_d \approx 2v\frac{f_0}{c_0}. \tag{3.2}$$

From this formula, the speed v can be extracted as follows:

$$v \approx \frac{c_0}{2}\frac{f_d}{f_0}. \tag{3.3}$$

3.2.1.3 Range computation

The method used to get the range of a target is simple: knowing that the speed of light is constant, the time difference between the emission of the signal and the received reflection can be used to compute the distance of the target. For this method to work, the device has to emit at a range of different frequencies. This is due to the fact that when a reflected wave is detected, the time at which the original wave was sent must be known so the delay can be measured. If the emitted frequency is constant, it is impossible to know when the original wave was sent. Therefore, as can be seen in Figure 3.7, the frequency is modulated following a certain pattern. The range can thus be computed by analysing the frequency variation between the signal that was initially sent and the received signal. According to the K-MD2 radar specifications, referenced in [90], the radar sends signals shaped as illustrated in Figure 3.7.

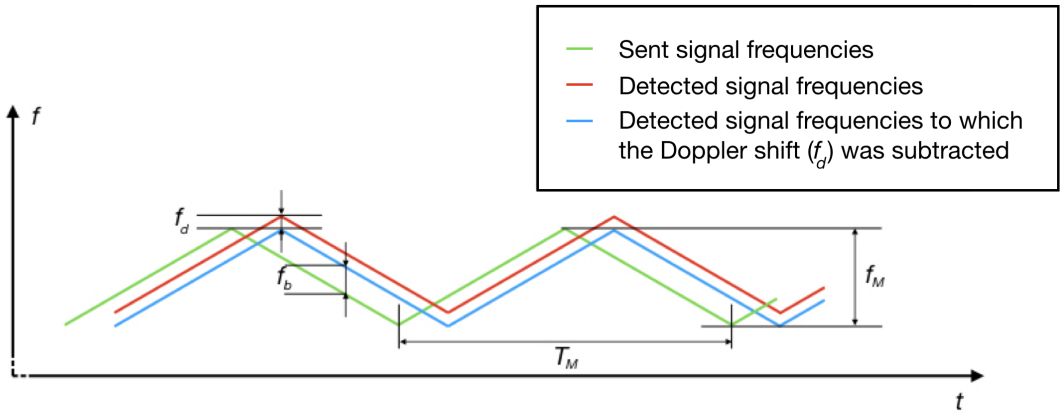


Figure 3.7: Representation of the signals sent by the radar compared to the detected ones.

Although it could be possible to measure the time between peaks, it is typically less efficient than comparing the signal frequencies at any given moment. The frequency difference between the sent and received signals is due to two different elements: the Doppler shift which introduces a frequency variation f_d, and the lag that the received signal has compared to the sent one which induces a frequency difference f_b as shown in Figure 3.7. Because only f_b is used for the range computation, the Doppler shift is subtracted from the received signal which results in the blue curve in the aforementioned figure. This Doppler shift can be computed by comparing the peak frequencies of the sent and received signal.

The range at which the target is from the radar can then be computed with a formula using f_b (the frequency difference between the sent and received signal adjusted for the Doppler shift), f_M (the frequency difference between the highest and the lowest frequencies emitted by the radar), and T_M (the time taken for one period of the sent signal).

$$R = \frac{c_0 \, |f_b|}{2 \, df/dt} \quad = \quad \frac{c_0 \, |f_b|}{2 \, \frac{2 \, f_M}{T_M}} \quad = \quad \frac{c_0 \, |f_b| \, T_M}{4 \, f_M}$$

When multiple targets are observed, the received signal is composed of the different frequencies coming from the various targets. This renders the task of comparing the sent and received signals slightly more arduous because linking each target with their appropriate frequencies is not always possible due to ambiguities. This means that, for each target, there will be a list of possible ranges at which it might be. For this reason, as described in [90], one observation from the radar is composed of 512 chirps that repeat the measurements explained above but varying the shape of the signal by changing the values of T_M and f_M, as well as changing the maximum or minimum frequencies of the signal. This variation will cause the estimated ranges to vary from chirp to chirp but the targets' true ranges will stand out because they will be the ones that will be consistent throughout the chirps.

3.2.1.4 Angle computation

In order to determine the azimuth angle from which the received signal comes from, at least two receiving antennas placed next to each other are needed. When two receptors' signals are compared, a phase shift ϕ can be observed if the target is not exactly in a direction orthogonal to the antennas' baseline. This phase difference can be converted into a distance d_{phase} by knowing the wavelength λ of the incoming signal using the following formula $d_{phase} = \frac{\phi}{2\pi} \lambda$. When the distance between antennas is several orders of magnitude smaller than the distance D separating the radar and the target, the incoming signal can be represented as parallel incident waves. From this assumption, a triangle as represented in Figure 3.8 can be constructed and the azimuth angle θ_x can be computed by applying the following formula:

$$\theta_x = \arcsin\left(\frac{d_{phase}}{d}\right). \tag{3.4}$$

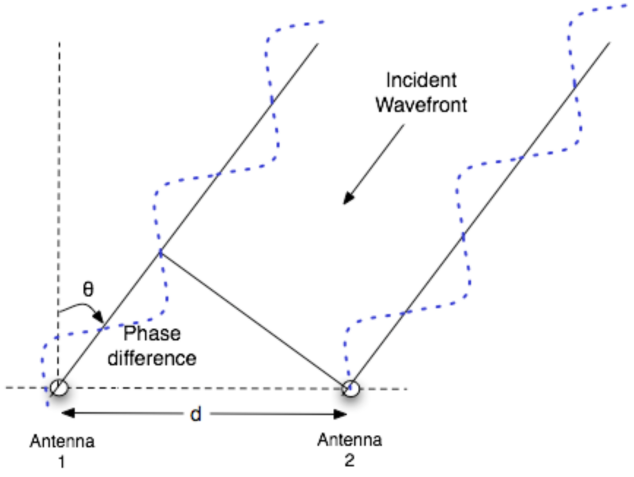


Figure 3.8: Angle computation. Figure from [90].

When the distance d between the two antennas is bigger than $\frac{\lambda}{2}$, the phase shift computed is uncertain and can have the value of $\phi \pm k \cdot 2\pi$ with k having a real value bounded according to the ratio between d and λ. The different possible angle positions are known as "gratin lobes". As the K-MD2 can vary its emitting frequency, the correct angle is usually correctly identified by finding the common computed angle between the detections obtained with different emitted wavelengths. Unfortunately, it was noticed that at some observations of the K-MD2, the radar would make a mistake and output completely erroneous azimuth angles for certain targets.

Finally, the angle of elevation can be computed in the same way when using at least two antennas spaced vertically.

3.2.2 Output analysis

As mentioned earlier, all the computations explained above are carried out by the device itself and then transmitted as arrays of values to a computer. As can be seen in Figure 3.5, returned observations cannot be always fully trusted due to noise from the environment and to the errors the radar makes. But most of the time, a pedestrian's position is correctly represented either by multiple targets grouped in a certain region of the azimuthal domain or by a single target with a large intensity attached to it.

As it will be explained in Section 5.3, the targets may be required to be represented in a common way for both sensors for the data fusion. Using the pinhole model described in Section 3.1.2.1, data from the camera can be converted to represent a target with a single point. All the points from the original radar data might thus have to be grouped in order to represent each detected pedestrians with single points as well. This type of problem falls under the category of cluster analysis which aims to sort data points into distinct groups, called clusters, depending on their location. Clustering is a vast subject with a large number of existing approaches to choose from, but as mentioned in [39], for a specific type of data, not all methods will find the desired clusters. Furthermore, some methods such as the *k-means* algorithm described in [58] require the amount of clusters to be known in advance, which can be restrictive for applications such as this one, where the number of tracked pedestrians can vary from one observation to the next. In order to select the best suited algorithm, inspiration was sought in articles describing clustering algorithms whose purpose also involves grouping multiple retrieved data points together in order separate different detected targets from one another. Papers such as [82] and [94] were found in which the *k-means* algorithm was used on its own or combined with a variation of the *DBSCAN* algorithm described in [38]. The *DBSCAN* method is a deterministic way of finding clusters based on the density of the points, while the *k-means* stochastically separates the points in a fixed number of groups trying to minimise the distance between each point and the centroid of the cluster in which it is located.

3.3 Data fusion

When dealing with multimodal sensors, after data is acquired, the different streams of information must be combined in some manner. Depending on the compatibility of the data types, this fusion can happen on many levels and with various amounts of preprocessing. Because of the immense number of different sensors that exist and the equally large amount of distinct types of data they output, data fusion is a vast subject[7] which is too broad to discuss in its full extent in this thesis. However, once the type of sensors and their data output are defined, only a small number of possible techniques can be chosen to merge them.

[7] [70] is an often recommended book for an in-depth description and analysis of the subject of data fusion.

The first important choice to make is at what stage of the data processing the fusion will take place. The different architectures that can be used are described in [70], [34], and [32] and will be looked at below in the first section. Once the architecture has been chosen, the second important choice that has to be made is the type of fusion itself. This subject will be discussed in the second section.

3.3.1 The different architectures

The first and simplest method is the centralised architecture. This technique consists in combining all the data coming from the sensors in one centralised node. With this approach, the only time information from multiple sensors is used by a same algorithm, is when it is processed by that node. Nevertheless, before being fused, the data streams coming from the sensors can be preprocessed independently so the main node can deal with richer information. As illustrated in Figure 3.9, the amount of preprocessing involved defines the type of centralised architecture used. If no preprocessing is involved, the approach is known as “direct centralised architecture” (Figure 3.9a). If basic feature extraction is performed, it is called “feature based direct architecture” (Figure 3.9b). If the data is completely processed and just has to be compared by the central node, the term “Autonomous architecture” (Figure 3.9c) is used.

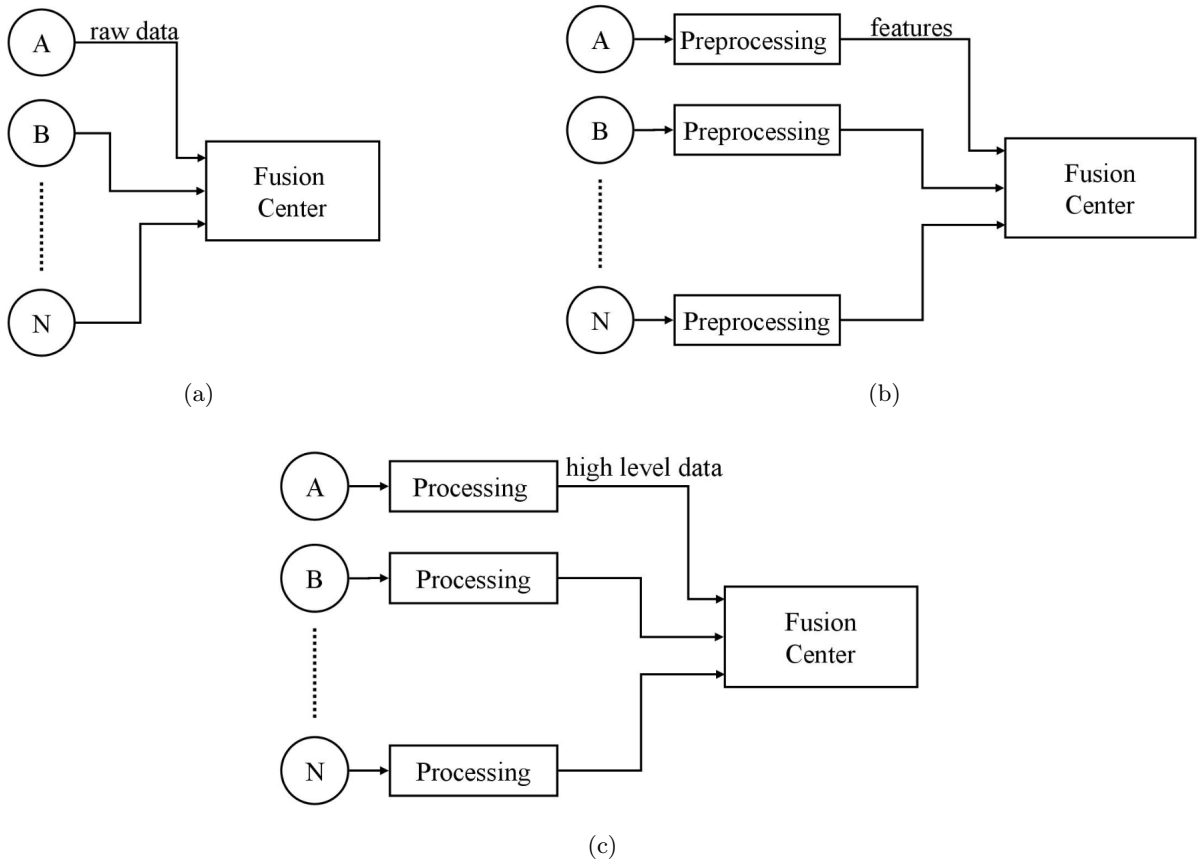


Figure 3.9: (a) Direct centralized architecture. (b) Feature extraction based centralized architecture. (c) Autonomous centralized architecture (figures taken from [32]).

Secondly, a method known as "distributed architecture" can be implemented. This method is based on a principle where each sensor is associated with a node that performs its own independent data fusion. These nodes all communicate with each other in order to constantly

update there state estimation of the global system. An illustration of this approach is shown in Figure 3.10. The advantage of this method is that it is completely scalable as sensors can be added or removed from the network without affecting the networks' operations. However, having multiple nodes working in parallel with global information can be redundant and can cause a loss of precious computation time.

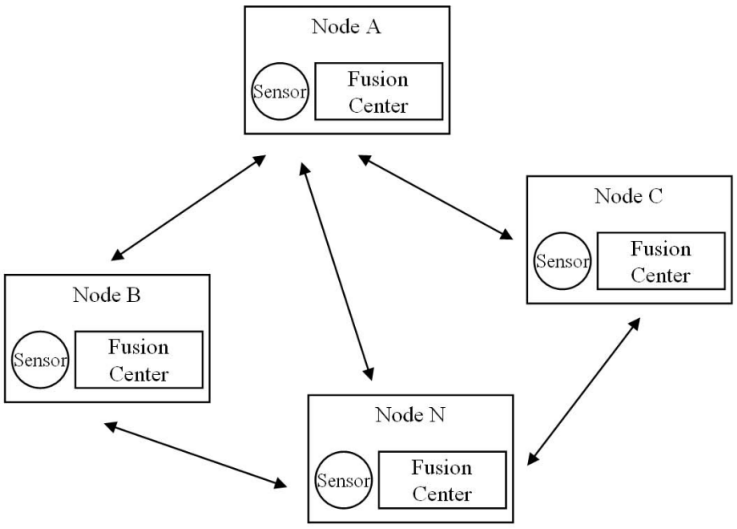


Figure 3.10: Distributed architecture (figure taken from [32]).

Thirdly, the last approach, described by [70], is the hierarchical architecture. With this approach, as with the centralised architecture, the final data fusion is also performed by a central node, but the preprocessing of the data involves already fusing data from multiple sensors in independent nodes whose outputs are then sent to the previously mentioned central node, as shown in Figure 3.11. This approach shares some of the advantages of the distributed architecture described above without its drawback. This architecture is able to deal with added and removed nodes without requiring many modifications of the centralised node's algorithm, and at the same time, redundancy is reduced due to the fact that each node performs different tasks.

3.3.2 Different fusion approaches

Once the architecture has been chosen, the actual approach that will be used to fuse the data is the next problem to tackle. The challenges that can be met during the fusion of data of any sort of sensors can be categorised as imperfect, correlated, inconsistent, or disparate
.

The chosen sensors for this project can trigger two sorts of these challenges: imperfect and inconsistent information. Indeed, the sensor data outputs are not correlated as they are both completely independent and, as 2D coordinates can be obtained from both sensors, sensor data are not disparate either. The two next paragraphs will thus briefly describe both imperfection and inconsistency in acquired data.

Imperfection The imperfection classification have been proposed in the literature in [79], [12] and, [44]. Imperfection is divided in three aspects: uncertainty, imprecision and granularity. When retrieved data is not absolute in that there is the degree of confidence about output, the data is qualified as "uncertain". If the data that is dealt with corresponds to multiple measured elements and it is not known what data point corresponds to which element, the data is called "granular". Finally when one data point corresponds to a measurement of multiple elements, the

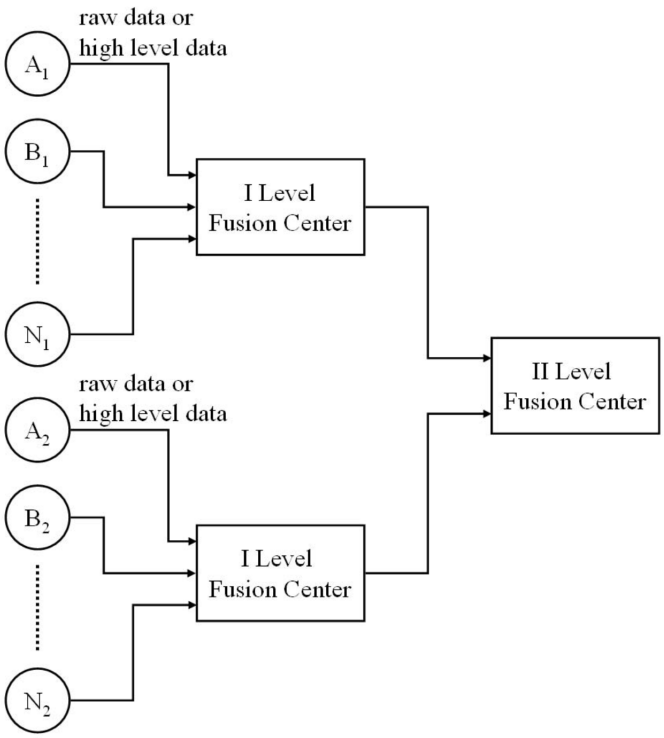


Figure 3.11: Hierarchical architecture (figure taken from [32]).

data is known to be "imprecise".

In this case, the data is uncertain, imprecise and granular. The data is uncertain because, for example, radar noise could be misinterpreted and believed to represent a pedestrian. Or because the visual detection algorithm could provide a false positive detection. The data is imprecise because radar observations or camera detections could represent a group closely packed pedestrians as only one. The data is granular as well because radar observations and camera detections are not referring to specific objects. The data fusion thus has to somehow identify which element of the data(observed positions) belongs to which object (pedestrian).

Imperfection is the most known problem in the field of data fusion and a lot of different techniques have been developed throughout the years. Khaleghi et al., following an analysis of the state of the art, established in [60] a list of different methods that can be used to tackle the challenges .The different methods cited are the following : "Probabilistic" [35], [8], [17], "Evidential" [95], [42], [13], [2], [4], "Fuzzy reasoning" [37], [91], [118], "Possibilistic" [43], [9], [33], "Rough set theoretic" [87], [78], [113], [52], "Hybridization" [109], [111], [118], [3], "Random set theoretic" [74], [50], [72], [73]. All of them have their own capabilities and limitations and are compared in the same paper.

Inconsistency Inconsistency in data can provoke spurious, disordered and conflicting data points [60]. Spurious data is provoked by outliers coming from unexpected failure. Disordered data comes from out of sequence information in the time space. Conflicting data occurs when the sensors disagree on measurements.

Regarding the camera's and the radar's data, only spurious data can be retrieved. Indeed, as stated earlier, due to problem of the grating lobes discussed in Section 3.2.1.4, the radar can sometimes return completely wrong positions. On the other hand, the data is always ordered,

and except from the fact that outliers can occur, which falls under the category of spurious data, the data does not conflict itself.

Most of the works in the literature trying to solve the "spurious data" problem [105], [56], [46] have been about predicting observations and eliminating false ones. But more recently, more promising results have been discovered using stochastic adaptive modeling of sensors with the Bayesian inference[8][65], [66].

[8]The statistical inference is the theory, methods and practice of determining the properties of the probability distribution of the state of a system, typically on the basis of random sampling, by analysing incoming data. The Bayesian inference is a statistical inference method using Bayes' theorem in order to update its probabilistic model of the state of a system as new data comes in.

Chapter 4

Multiple object tracking

After potential targets have been detected from a sensor's observation, their positions have to be matched and linked, if possible, with previous data so they can be tracked in real time. The main difficulty at this point consists of differentiating between targets and being able to track them accurately in every possible condition (such as occlusions, crossing paths, etc.), all of this in real time. The purpose of this chapter is to present the sate of the art of the tracking algorithms compatible to the fusion of a camera and a radar.

A large number of algorithms can be used in order to perform the task of multiple object tracking using various types of sensors, however most of them are developed only for a specific type of target representation. Some approaches use 2D or 3D bounding boxes to represent targets, while others deal with more complex shapes and in some cases targets are only represented by a single point. The choice of this representation largely reduces the amount of tracking methods that can be applied. Because, 2D point-positions in the azimuth plane can be extracted from both sensors targets will be represented as such points. In the case of point tracking, as mentioned in [112], two main types of methods can be found in the literature.

The first one consists of a deterministic approach in which paths are computed using the positions from the current time step as well as a certain number of previous ones. The newly computed paths will then be used to determine to which target the current detections correspond. The second method is stochastic, in that the algorithm deals with the estimated state (position, speed, acceleration, etc.) of targets at all times. The estimations are updated iteratively while the program goes through the frames one by one. These two methods can be used on their own but some more advanced algorithms choose to combine these approaches in order to benefit from the strengths of both. An overview of the current literature on these tracking techniques will be discussed below.

4.1 Deterministic approach

As mentioned above, the idea behind this approach is to round up all the detections from selected frames and attempting to find the most likely paths that the targets have followed. Usually, a graph based formalism is used for this type of data association, as in [59], [108] and [7]. A weighted directed graph is constructed with nodes representing the detections. The edges connecting nodes together are weighted to represent the probability of these nodes corresponding to the same target. The edges are also directed so that they only go from an older detection to a more recent one. Once the graph is built, the best paths through the graph have to be found so trajectories can be given. An example image of a graph, found in [7], is shown in Figure 4.1.

The way in which the weights of the edges are computed can be very complex and varies

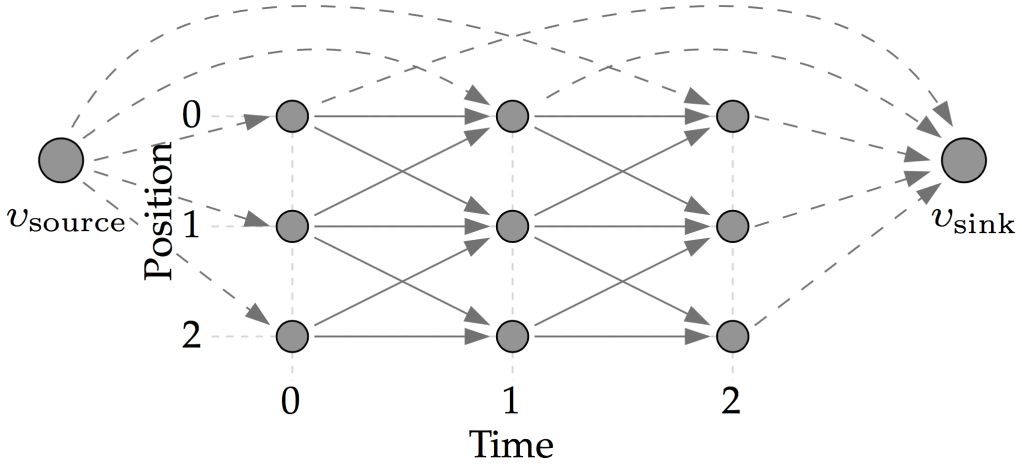


Figure 4.1: Example image of a detection graph using the *k-shortest* path [7].

between algorithms. Many elements are taken into account in order to get a better accuracy. Generally, the dynamics of the targets' movements are sufficiently well-known to set boundaries of speed and/or acceleration, such that edges corresponding to unrealistic movements can be given high weights. On top of that, additional information collected about the targets can be added to the computations as to improve its efficacy. For camera-based algorithms, it is common to add visual cues such as the features presented in Section 3.1.1.2 to the weight computation in order to discriminate between detections from different targets.

In the simplest versions of this approach, such as the one described in [64], nodes can only be connected to each other if they belong to consecutive frames. This means that if a target is not detected in one or more frames, the tracking algorithm could fail to compute a realistic path. This problem can be overcome, as proposed in [59], [108] and [7], by establishing possible connections between non-consecutive frames. Hence, the new challenge consists of deciding how far the edges can reach. This is crucial because the number of edges will grow rapidly with the number of frames that can be connected, which might cause the computation speed to decrease considerably.

Once the graph is built, the best paths have to be found. This can be done in several ways. Some algorithms compute the single shortest path starting at each original detection using methods such as the A^* algorithm [108] while others, more complex algorithms, explore multiple paths before choosing an optimal one. This is the case in the method discussed in [7] which uses the *k-shortest* paths algorithm in order to improve the reliability of the tracking.

4.2 Stochastic approach

The methods using this approach, as introduced earlier, work by estimating and updating the targets' states iteratively. A number of different algorithms following this approach have been developed, but the best known ones are based on recursive Bayesian filtering. This type of filtering recursively estimates an unknown probability density function using a process model and received measurements. This update is composed of at least two steps, a prediction step that estimates the targets' states of the current time step using previous estimation as well as a dynamic model, and the innovation step that adjusts the prediction using the measurements taken at that time step.

Depending on how the environment and the sensors are modelled, one of the two following techniques are used. If the assumption of a linear dynamic model can be made and the target position estimation, as well as all error models, can be modelled by normal distributions, then the *Kalman filter* is often used due to its great computation speed. When these assumptions cannot be made, *particle filters* are the most common alternative. These two filters will be described below.

4.2.1 Kalman filter

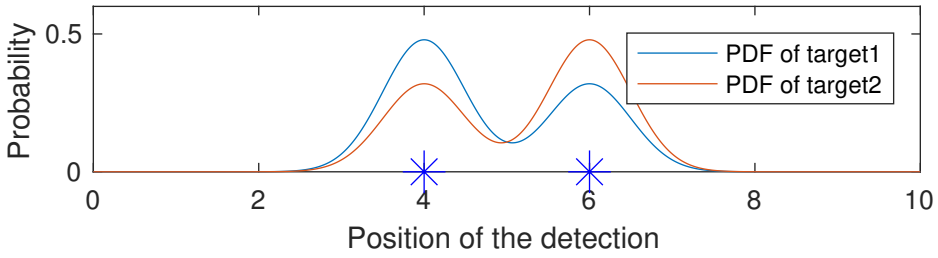
This type of filter, originally described in [57] and further developed in papers such as [69] and [106], recursively estimates the state of a linear system with a multivariate normal distribution. Thanks to this approach, the state estimation of a target can be defined only using a mean and a covariance matrix describing the variance and correlation between the different elements making up the state of the target. The fact that a target is described in such a compact way makes its manipulation extremely efficient compared to other models such as the *particle filtering* method described below.

There are however a few requirements that are needed in order for this tracking method to work. Firstly, the observation noise, that represents the inaccuracies of the sensors must be modelled by a normal distribution whose characteristics (mean and covariance matrix) must be known in advance. The target's dynamic model has to be known or correctly approximated as well to ensure an optimal prediction step. Finally, a process noise, representing the inaccuracy of the dynamic model has to be defined, again, following a normal distribution. With all these known values, when new sensor data is available, the state estimation (mean and covariance matrix) can be updated easily with a series of simple equations. The compactness of this computation makes this method extremely fast. But despite this algorithm's computational speed, it has a few drawbacks.

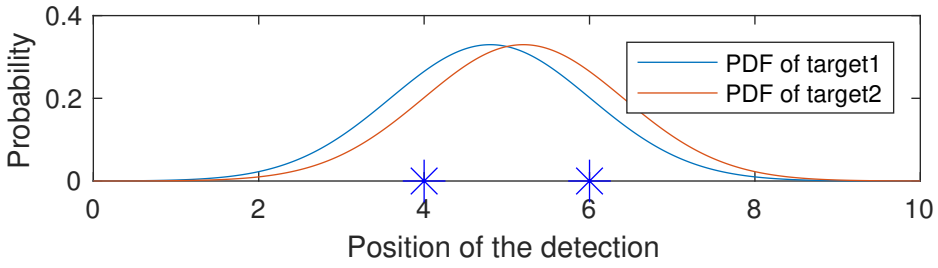
First of all, as mentioned above, the observation noise has to be known and has to be modelled by a Gaussian distribution. Due to this restriction, if the sensor noise is not actually normally distributed, this algorithm will lack accuracy. In addition to that, even if the sensor noise can be represented by a normal distribution, extensive and rigorous testing has to be carried out to find an accurate covariance matrix associated with the sensors. Furthermore, this covariance matrix may depend on the environment in which the experiment is set up and may have to be computed every time the device is placed in new conditions.

Secondly, the fact that the targets' states are represented by Gaussians may work well for single object tracking, but this can cause irregularities when multiple targets are introduced. This is because, when several targets get close to each other, it might not be known for sure what detection correspond to what target. This flaw is represented by Figure 4.2. The appropriate way of representing the probability density function (PDF) of this scenario would be with a weighted sum of normal distributions centred on the detections, as shown in Figure 4.2a. But with a classical *Kalman filter*, only one normal distribution can be used for each target, thus obtaining distributions similar to those in Figure 4.2b, which do not accurately represent the actual probability density function of the targets' positions.

Some algorithms solve these problems by modifying the basic algorithm to handle this kind of situations, which is the case of [106] among others. However, because of these drawbacks, particle filters are often used despite being computationally heavier to run.



(a)



(b)

Figure 4.2: Probability density functions (PDFs) of two targets' positions with measurements in 4 and 6. (a) Using a weighted sum of normal distributions. (b) Using single normal distributions with weighted values.

4.2.2 Particle filter

This type of filters, also called "sequential Monte-Carlo methods", believed to be originally described in [51], estimates the state of the targets using a large number N_p of weighted particles. These particles all have their own estimation of the state of the target that will be updated independently at every time step. Because some states are more probable than others, it is common for several particles to end up sharing the same state estimations. Thus, the target's most probable state will be the one shared by the most particles.

The particle filter is initiated when target is detected. N_p particles are then created to represent it. Every time a new measurement is made, the particle filter will perform a state estimation update of all the particles following three different steps illustrated in Figure 4.3.

Firstly, the prediction step is performed as follows. All the particles are moved using a chosen dynamic model that, as opposed to the *Kalman filter*, does not necessarily have to be linear.

Secondly, the innovation step is performed to give weights to these newly computed particles. The weights are obtained by determining how well the state estimations of the particles fit in a probability density function constructed around the measurements that were taken. This PDF is established by combining normal distributions centred around each measurement. The characteristics of these normal distributions depend on the sensors' error models that have to be defined beforehand.

Thirdly and finally, when the weights have been computed, one last step, called "resampling", is needed to reorganise the particles. The intent of this step is to create N_p new particles that follow the same PDF as the one computed at the innovation step, but where all particles share the same weight. This is important so that the prediction and innovation steps at the next iteration can treat all particles equally. In order for this to work, the particles that are the most likely to represent the target (those with higher weights) are selected multiple times and some less probable ones (those with lower weights) are not selected at all. This is illustrated in Figure 4.3, with the green dots representing the particles that are selected one or more times and the

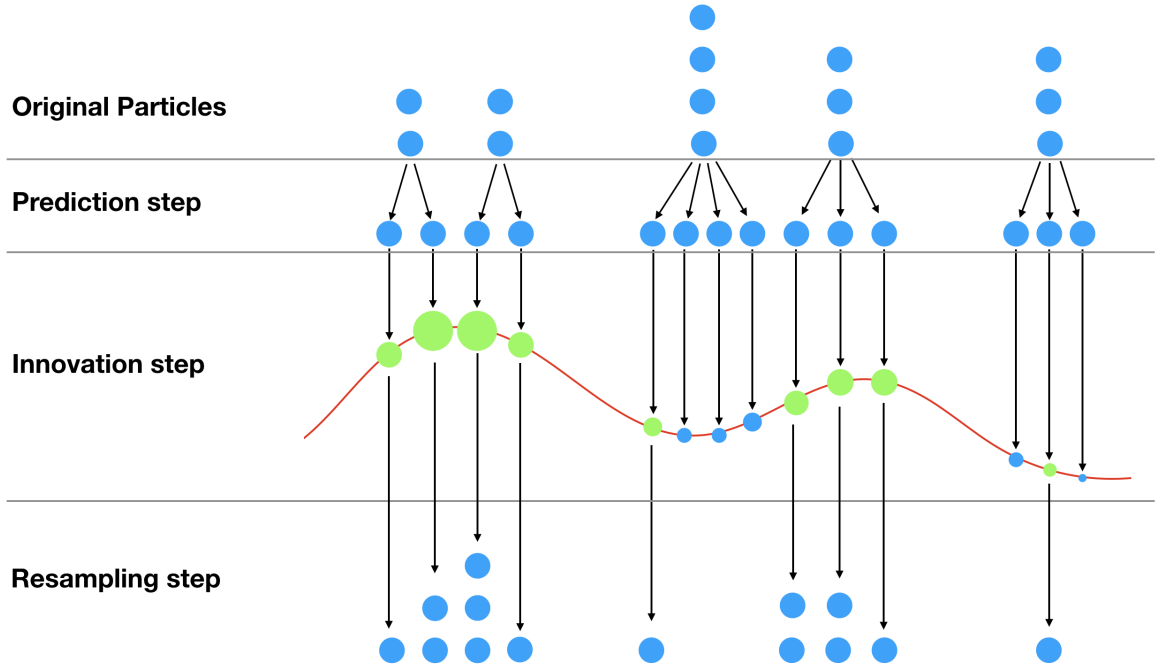


Figure 4.3: Representation of a complete update of a 1D particle filter. The dots are the particles and their size represent the weight associated to them. The sensor measurement PDF is represented by the red curve. Stacked particles indicate that they all share the same state estimation.

ones that remain blue that are not re-selected. There exists multiple resampling techniques but some are more common than others, as described by [54]: “The most frequently encountered algorithms are multinomial resampling [51], stratified resampling [22], [62], systematic resampling [62], [1], and residual resampling [71]”. Although this article was published more than ten years prior to this work, it was found that the claim is still valid to this date.

The explanation above indicates how one target can be tracked, but in order to extend the tracking to multiple targets simultaneously, two main approaches can be considered.

Firstly, the algorithm can be programmed to follow all the targets with a single particle filter with a "Joint Multitarget Probability Density" (JMPD), as in [63], which keeps for every particle a weight for each target. This means that there is a fixed number of particles, and the more targets there are, the less accurate the tracking will be due to the fact that each target will have a certain portion of the particles following it primarily.

The second approach, as discussed in [110], is to create a new particle filter every time a new target is introduced. The detections are then analysed and weighted for each target corresponding to how probable it is that these detections belong to the targets. This method, while more accurate, can largely increase the running time of the program and make real time tracking more complicated.

In both cases the number of particles per filter is crucial and has to be adjusted as a function of the desired computational running speed and the number of targets that have to be handled.

It should be mentioned that having to determine the observation noise is less of a drawback for the *particle filter* than it is for the *Kalman filter*. This is because the *Kalman filter* represents

the target's state with a normal distribution and a potentially inaccurate sensor model could drastically change the shape of this distribution. On the other hand, a particle filter, with a more flexible state representation, will be less influenced by such inaccuracies.

4.3 Hybrid methods

As mentioned earlier, some algorithms combine both approaches to this problem. Usually, as described in [7], this combination works by using a stochastic method in order to create short pieces of paths called "tracklets" that span through a small amount of frames and then creating full tracks by merging these tracks together with a deterministic method. One of the simplest examples of this hybridisation is described in [88], where a *Kalman filter* is used to draw tracklets and the graph search described in [64] is used to join them. The literature contains numerous other examples of combinations of algorithms, such as those described in [7], but the full analysis of these methods is beyond the scope of this thesis.

Chapter 5

Preliminary Choices

After every aspect of the different steps leading to pedestrian tracking have been explained, the appropriate methods to be used in this project for each of those steps have to be chosen. This project joining several large fields of study, it would not be feasible to try every single combinations of the building blocks explained in the previous chapters. Therefore some choices had to be made so that we could have a clear direction in which to start our study. These choices were made based on previous papers describing similar projects, the unique details of the setup and this thesis' authors' scientific intuition. The following sections will go over these choices in detail.

First, the chosen pedestrian detector will be described along with its choice justification and performances. Secondly, preliminary choices concerning the data fusion will be presented. Thirdly, the choice of the tracking method will be explained. Fourthly and finally, the choice of tests and evaluation criteria that will be used to optimise and assess the tracking will be outlined.

5.1 Pedestrian detection

The chosen pedestrian detector will have to fulfil a series of requirements which can be divided into two categories: performance and practicality. Performance of a pedestrian detector mostly comes down to its detection quality and its computational speed while practicality relates to the ease of use of that detector.

Among the potential pedestrian detectors presented in the state of the art of Section 3.1.1, the algorithm best meeting the previously stated requirements has been written by Dollár, Belongie and Perona and is called "The Fastest Pedestrian Detector in the West". It is using *integral channel features* (*ICF*) along with a *boosting* learning algorithm. The related papers can be found in [23] and [29].

The "Fastest Pedestrian Detector in the West", which uses *ICF*, is an image classification algorithm focused on pedestrian detection. The general idea behind it is that multiple registered image channels are computed using linear and non-linear transformations of the input image as explained in [47] and [76]. Afterwards, features such as local sums, (local) histograms [89] and Haar-like wavelets [102] features and their various generalisations [26] are efficiently computed using *integral images*. Some features are approximated to nearby scales in order to avoid the repetition of slow computation. These features are then fed to the boosted classifier [45] which is itself composed of cascade classifiers[1].

[1]Cascade classifiers is a method of concatenating multiple classifiers. Classifiers are run in a consecutive fashion and the information of the previously passed classifiers is used in the next computations.

Despite being from 2010 and of a lesser performance than more recent algorithms, the "Fastest Pedestrian Detector in the West" is still considered as a remarkable work for its efficacy and speed [26], [27], [119], [24]. In 2010, [23] shows that results on the Caltech-USA Pedestrian Dataset [28] show a detection rate of almost 60% at 1 false positive per image compared to a 50% rate for the formerly competing methods and with 1-2 order of magnitude faster. A more thorough study of the performances with the current setup will be described below in Section 5.1.2.

In addition to the good performance of this algorithm, it has multiple practical advantages as well. Firstly, the code is written in MATLAB, which makes it easier for the implementation as the radar's code is in MATLAB as well. Secondly, this code is licensed under the Simplified BSD License which imposes minimal restrictions on the use and redistribution of the code, which suits the potential commercialisation requirement of this thesis' work.

This section will first elaborate on the chosen algorithm and will then make a study of the performances of the chosen detector.

5.1.1 Description of the algorithm

The algorithm is decomposed in multiple steps and has many optimisation tricks such as the *integral image* method and the feature approximations. These will be described below.

Candidate generation The candidate generation step is performed using the exhaustive scan and is a hybrid method between the dense image pyramid approach and the classifier pyramid approach. This hybrid method consists of dividing the image into multiple classifier pyramids at different image scales as shown in Figure 5.1. In each of these pyramid scale, Dollár, Belongie and Perona discovered that features could be approximated at nearby scales. This feature approximation will be explained below. The scale step is of a factor of $\delta_h = 2^{\frac{1}{10}} \approx 1,07$ and the minimal size of a bounding box is of 100 pixels in height.

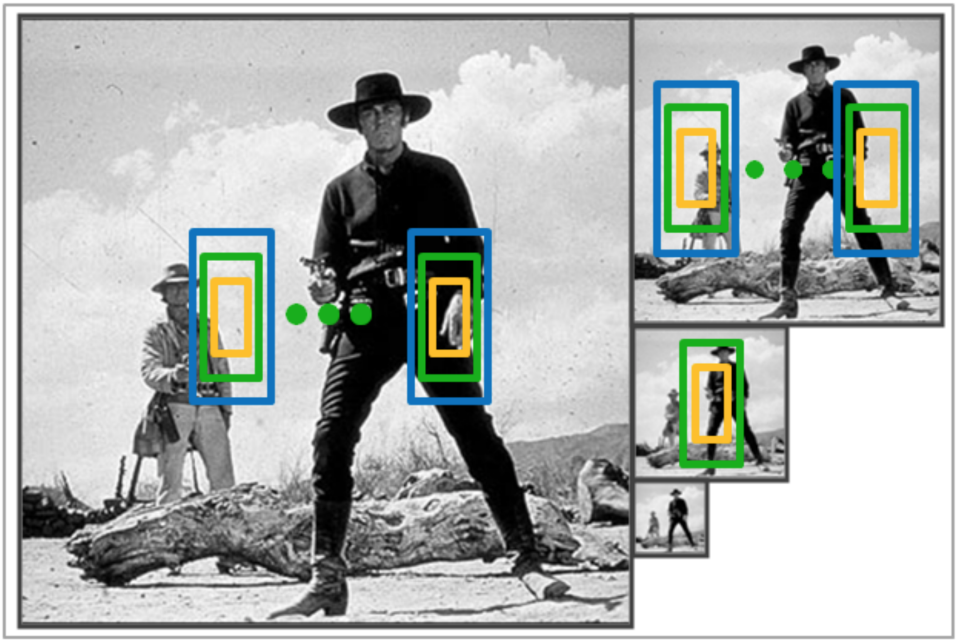


Figure 5.1: Exhaustive scan using multiple scales of windows and images. Figure found in [23].

Image channels Image channels are computed over the whole image. It consists of linear or non-linear transformations that can be performed only once because image channels are translationally invariant and thus do not depend on the position of the sliding window. Image channels examples can be found in Figure 5.2. Image channel computation is even optimised by the use of linear approximations of non-linear transformations. Three types of channels are used: gradient histograms, colour (including grayscale, RGB, HSV and LUV), and gradient magnitude.

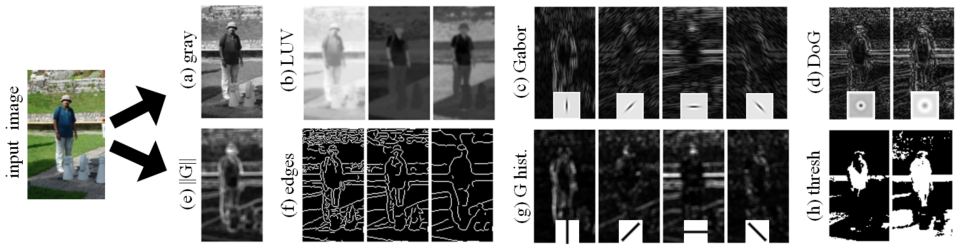


Figure 5.2: Different channel examples found in [29].

Features Two types of features are distinguished: the first-order features and the higher order features. For an analysed window, a first-order feature corresponds to the sum, over a certain portion of the window, of the values obtained through the channel to which the feature is linked to. Th higher-order features are computed through weighted sums of multiple first-order features.

Feature approximations For many detection algorithms, the bottleneck lies in the computation of image channels on every scale of the image [20], [29], [40], [75], [107]. Indeed, most of the detectors use the classifier pyramid for its ability to use many features that are not required to be scale invariant. However, the classifier pyramid needs the image to be computed at each scale which requires a long computational time. The idea of Dollár, Belongie and Perona is to divide the image into multiple classifier pyramids at different image scales, such as in Figure 5.1, and approximate features (such as gradient histograms) at nearby scales. The proposed approach achieves nearly the same accuracy as using densely sampled image pyramids, with nearly the same speed as using a classifier pyramid applied to an image at a single scale [23]. This approximation leads to a minor loss of detection accuracy (1-2%) but yields a 10-100 times faster detection process. More information on this feature approximation can be found in [23].

Integral image *Integral image* is a method allowing to perform fast local sums of pixels in rectangular areas within an image by performing only one main computation at the beginning of the process. The principle is that every point on the image is represented by the sum of all the the pixels positioned before it along both axis. The formula [14] giving the value $I(X, Y)$ at each pixel is thus

$$I(X, Y) = \sum_{x=1}^{X} \sum_{y=1}^{Y} f(x, y)$$

with $f(x, y)$ being the value of pixel at a (x, y) position. The sum of a rectangle defined by $(x_1, y_1; x_2, y_2)$, as shown in Figure 5.3, is then

$$S(x_1, y_1; x_2, y_2) = I(x_2, y_2) - I(x_1 - 1, y_2) - I(x_2, y_1 - 1) + I(x_1 - 1, y_1 - 1).$$

An example image is shown in Figure 5.3.

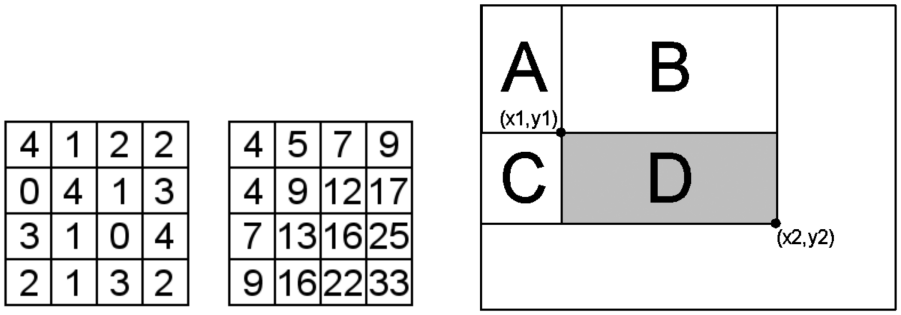


Figure 5.3: *Integral image* example image found in [14].

Adaptive boosting The principle of adaptive boosting is explained in Section 3.1.1.2. However ways the iterative training phase is performed in this case needs to be explained, and works as follows. Random training data is picked from the training data set and a weak classifier is built. This weak classifier is then tested on the whole data set in order to detect where its major strengths and weaknesses are. After that, another weak classifier will be trained, again using randomly selected training data, but where data points that the previous classifier got wrong are more likely to be selected. This procedure is then repeated a large amount of times. In the end, the boosted classifier is actually composed all the weak classifiers that together form a strong one. Because the strengths and weaknesses of all the weak classifiers are known, when an image is tested, the boosted classifier determines which weak classifiers should have more weight in the final descision.

Duplicates suppression In order to suppress nearby detections, a *non-maximal suppression* algorithm, with the *pairwise max suppression* approach [41] is run. This approach has been chosen for its simplicity of use as only one parameter is needed.

5.1.2 Detection performance

The tested performances of the algorithm will be described below. The performances can be divided in two categories: the detection quality and the computational speed.

Detection quality As mentioned earlier, the chosen detector is said to have 60% detection rate at 1 false positive per image on the challenging Caltech-USA data set. However, testing the detector on easy-to-detect images such as the one in Figure 5.4 taken by the setup, better results were discovered. Indeed, a detection rate of almost 100% at 0 false positive per image was found. On more challenging data, the detection rate is lower and expected to be close to the detection quality described by the detector's authors.

The detection range is another criterion for the detection quality. The chosen detector is trained in order to be able to detect people of a minimal height of 100 pixels. Using the pinhole model, this value corresponds to a range of approximately $25m$. By manually testing the detector in good conditions as well, the detection rate is, as expected, at approximately $25m$. As the original goal was to track pedestrians up to fifty meters, this maximal distance is very restrictive.

Computational speed The computational speed of the detector on a typical computer is about $0.2s$. Coupled with the time needed to save the image which is significant, the usual detection frequency is of approximately $4fps$.

Figure 5.4: Example image with the detector's detected bounding box.

5.2 Data Fusion

A preliminary choice that has to be made before that of the data fusion architecture and approach is how the different sensor firing rates will be dealt with. As stated earlier, the radar outputs observations at approximately $6fps$ while the camera outputs at a frequency of approximately $4fps$. Thus, the sensors outputs arrive in a disordered sequence. Two different possibilities could be chosen from, either the data from both sensors can be directly treated independently by the tracking algorithm as it arrives, or it could be fused with previous information from the other sensor before sending it an updated detection. The second option was deemed as not optimal because past data from the sensors would have to be extrapolated in order to estimate the conjoined detection. Because certain tracking algorithms are already designed handle this type of data fusion, it was considered useless to perform additional computations to fuse the data prior to the tracking.

5.3 Tracking method

In this section, the justification of the tracking algorithm choice will be first explained.

After establishing a list of existing tracking techniques that could be used for point tracking, a choice had to be made as to what method should be implemented. In order to decide on the tracking algorithm, a search for previous works with similar goals to this thesis' was performed. After a thorough look through the existing literature, two most relevant[2] articles were selected. These are [61] and [55]. They describe algorithms used to track multiple objects using both a camera and a radar. Although these papers propose solutions to tracking pedestrians with a device mounted on a vehicle, which is not the primary goal of this thesis, the methods implemented can also be of use for a static setup. The algorithms developed in both cases work using a stochastic approach, with [61] choosing a *Kalman filter* and [55] a *particle filter*. Even though deterministic methods have been proven to give great results for camera based tracking (as in [59] and [7]), they are not popular when multimodal sensors are used. This is one of the reason a deterministic approach is not selected for this project but the major reason concerns the way observations are made by the radar.

As mentioned earlier, the radar's output is made up of multiple observation points, with a

[2]An additional work should also be mentioned. [101] describes a multiple object tracking solution developed during the *EU FP7* (the Research and Innovation funding programme established by the European Union from 2007 to 2013) project SPENCER, which aimed to track people using a camera and a 2D laser. The obtained results were impressively accurate, but because the radar is not as accurate as their lasers and due to the complexity of their algorithm, it was decided that it would not be feasible to develop a similar algorithm in this thesis.

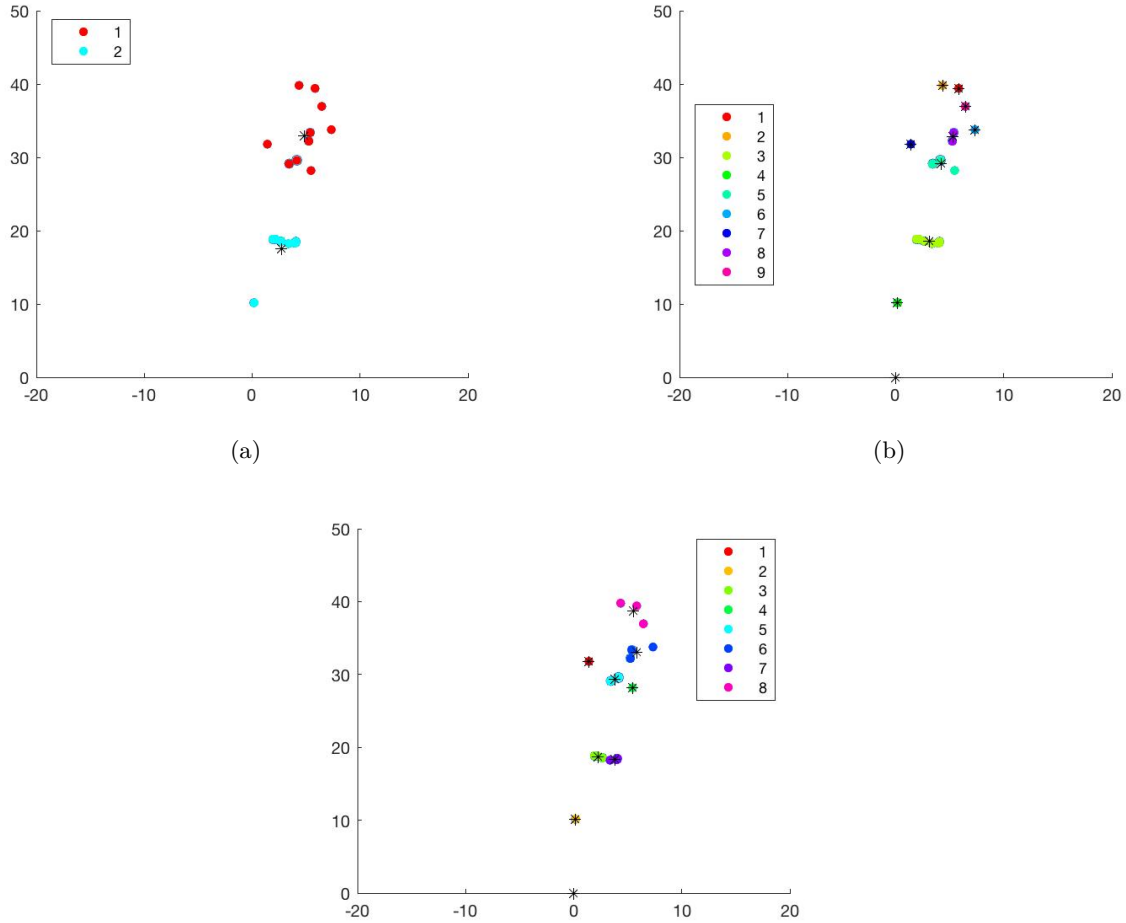


Figure 5.5: Different outputs of the same *k-means* algorithm ran on the same data set. The colours represent the clusters the points were assigned to.

large number of false positives, but, most importantly, with multiple observations belonging to the same target. These multiple observations per target cannot easily be used directly as input for deterministic method, nor for a Kalman filter. The data has to undergo a cluster analysis first in order to obtain well defined positions for each potential target detection. Thus, the clustering methods described in Section 3.2.2 were put to the test.

The first clustering method that was tested is the *k-means* algorithm. The first challenge that this method imposes concerns the number of clusters that have to be found. As discussed previously, for the simplest versions of this algorithm, this number has to be known in advance which can be challenging when the number of targets is unknown. There are methods that can be used to determine this number, such as discussed in [58], but as it can be seen in Figure 5.5, due to its stochastic nature, this algorithm is not consistent and can be unreliable at times. For these reasons, this cluster analysis method was dismissed.

The second method analysed was the *DBSCAN*. While this method is deterministic and computes more consistent results, its parameters have to properly be tuned in order to obtain usable results. The two main parameters that have to be defined are the minimal number of points needed to represent a cluster and the maximum distance between two points. Due to the inconsistency of the radar's observations, tuning these values has proven to be a hassle. There is

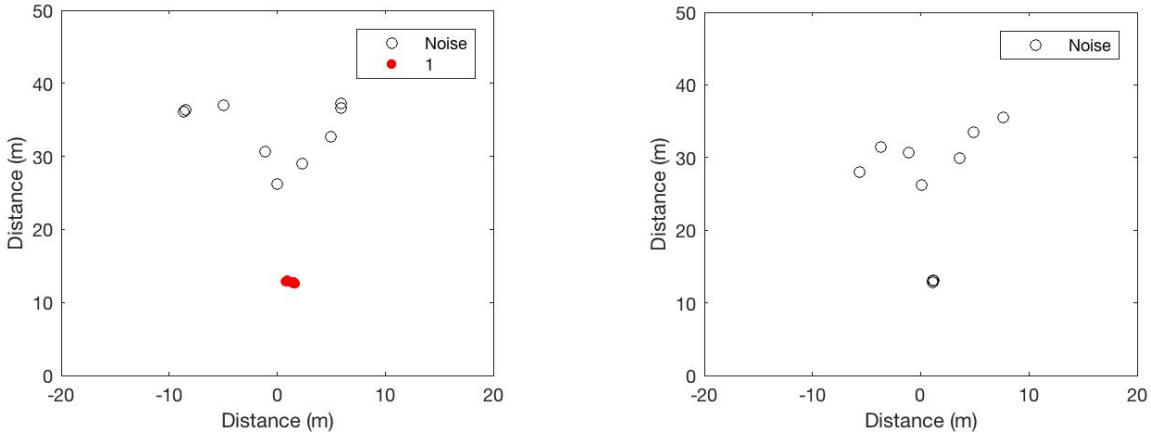


Figure 5.6: Results of the first version of the *DBSCAN* algorithm. (a) At iteration I. (b) At iteration $I + 1$.

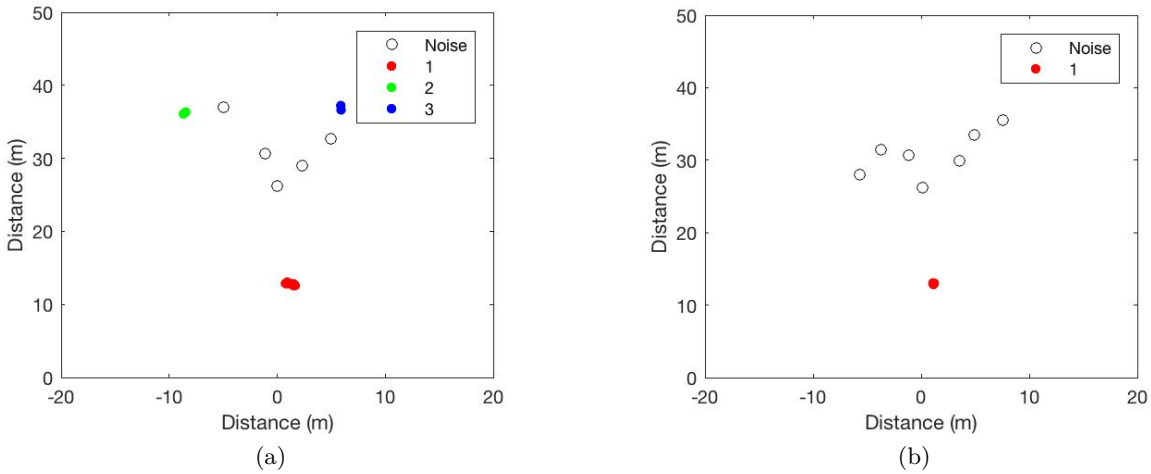


Figure 5.7: Results of the second version of the *DBSCAN* algorithm. (a) At iteration I. (b) At iteration $I + 1$.

no real consistency in the way that the radar detects targets and this means that these targets can appear as completely different patterns of observations from one time step to another. This behaviour can be observed in the following example.

As can be seen in Figure 5.6a, the *DBSCAN* is tuned so it can rightfully detect a target at iteration I of the radar's update. At this time step, the target is detected as a large amount of points are separated by at most three meters around the pedestrian location. At the next iteration shown in Figure 5.6b, the *DBSCAN* algorithm was applied with the same parameters and the target was not found. This is because at that next iteration, the output from the radar, for this target, is composed of only two points positionnedd extremely close to each other. The *DBCSAN* variables could be trained to work for case resembling the one at iteration $I + 1$, as in Figure 5.7, but as it is shown, that can create false positives at other iterations. The *DBSCAN* algorithm, applied on the radar's inconsistent data, outputs unreliable information that was deemed too poor to be used for the tracking.

Therefore, because of the poor results obtained by clustering algorithms it was decided to follow the method used in [55] and implement a *particle filter* algorithm that would not require a cluster analysis to work. In order to handle multiple targets, as described above, either a JMPD can be used, or a new particle filter can be created for each target. Because no particular advantages were found by using one or the other, it was decided that multiple filters would be implemented in the first version and if it was to be shown that this method would not perform as desired, the algorithm would be changed to use a JMPD.

5.4 Testing methods for algorithm evaluation

Before implementing the complete algorithm, the way it will be evaluated has to be determined. The developed device is made to track pedestrians' positions and draw their trajectories in the the 2D azimuth plane, while being set at the same height as said pedestrians.

During the development of the algorithm, a decision had to be made on how to rate the accuracy of the tracking such that improvements' effects could be measured. The usual methods used to rate tracking algorithms are made specifically for visual tracking only. The universal databases that are used to perform benchmark tests, like those in [77] and [67], do not contain any radar data, which is an integral part of this project. It was thus decided to create a data set that would be used as a benchmark.

This section's structure is as follows. First, challenging situations that the tracking algorithm might encounter will be described. Secondly, there will be a description of the tests that were imagined to simulate these challenging situations independently. Thirdly and finally, the tracking quality evaluation methods will be explained.

5.4.1 Challenges

There are two distinctive challenges that will have to be faced. Firstly, the algorithm should be able to accurately estimate a pedestrian's position even during noisy situations. Secondly the algorithm must be able to differentiate between the targets it tracks and precisely know which target represents what pedestrian. In situations where pedestrians cross paths or occlude one another, the algorithm must be able to correctly converge back on the right target.

It is important to note that accurate pedestrian tracking and handling of multiple targets are considered as independent challenges and are thus tested as such. This is because the first challenge aims at looking at how the algorithm deals with inaccurate data, in the form of sensor noise as well as occlusions and how well it can converge on actual targets. The handling of multiple targets is different in that the challenge will not come from the sensors or the environment but rather come from the way that each target is handled by the particle filter. The difficulty here lies in updating each filter with the right information and ignore detections of other targets. An extremely accurate tracking algorithm that can find the exact location of a target would be useless if it could not differentiate between various detected targets, and, conversely, an algorithm capable of perfectly differentiating between targets would not be interesting if the estimations of their positions were totally off.

5.4.2 Tests

In order to be as accurate and precise as possible, the ideal tests that could be performed would have the device capturing data on pedestrians in different situations and manually record the exact positions of the targets at every time step. By comparing the results of the tracking algorithm with the real-world values, the accuracy of this algorithm could easily be computed. Unfortunately, such an in-depth test was not feasible with this project's resources, thus alternative

methods were found and will be described below. As expressed in the introduction, this thesis aims at testing challenging situations for both sensors and, therefore, all the tests were performed with the device placed at no more than one meter above the ground.

Accurate target tracking In order to check the quality of the tracking without having to record real world positions, simple tests were set up. For each of them, circular patterns of known radii were drawn on the ground at known positions. The test participants were then instructed to follow these paths while the device captured the data. The paths computed by the algorithm are then to be compared to the actual circular pattern that the pedestrians followed.

In an effort to address multiple challenges, several tests were created. These tests attempt to influence sensor's data and its accuracy with occlusions and challenging appearances and light exposures for the camera and by adding noise for the radar. These tests are listed in Table 5.1.

Challenge tested	**Test situation**
No challenge(ideal situation)	5m radius circular path 20m away from the device
Occlusions	5m radius circular path around a 40cm radius tree 20m away from the device
Lighting variations	5m radius circular path 20m away from the device partially in the sun and partially in the shade
Radar noise	5m radius circular path partially among plants moving with the wind, 20m away from the device
No radar data	5m radius circular path 20m away from the device
No camera data	5m radius circular path 20m away from the device

Table 5.1: Tests to assess the accuracy of the tracking.

Handling of multiple targets This challenge will be tested by having multiple people walking in different patterns with clothes of the same and different colours. The corresponding tests are listed in Table 5.2.

Challenge tested	**Different colours**	**Same colours**
People crossing paths from far away with constant speed and direction		
People crossing paths with constant speed and direction		
People changing direction as they meet		
People changing direction and crossing each other as they meet		

Table 5.2: Tests to assess the handling of multiple targets. Distinct colour arrows mean a different clothing appearance while same colour arrows mean a same clothing appearance.

5.4.3 Evaluation

Tracking quality is difficult to assess. As there are no existing benchmark for similar experiments, no baseline can be used in order to properly compare the results. Two evaluation criteria are thus proposed and will be cited below. Each one of them will be discussed after the presentation of each implementation of the tracking algorithm.

5.4.3.1 Accuracy

For monocular data, the most general way in which the accuracy of tracking algorithms is assessed is through benchmarks such as the "MOT Challenge" [77]. Although these benchmarks are mainly used for object tracking on the 2D image space, some databases can be used to assess tracking on the azimuthal plane. The difference between these two types of tracking is crucial. The errors between ground truth and the tracking results are compared using measurements in the image space for the first one and in the azimuthal plane for the second one. The interesting benchmarks for this thesis' results are thus those evaluating tracking errors in the azimuthal plane. Such benchmarks are discussed in [67]. The accuracy of these algorithms is generally determined using the $MOTP_{3D}$ that computes an average value of the distance between the computed position and the ground truth. Unfortunately, this measurement could not be used because, as stated earlier, it was not feasible to capture the exact position of a target at every time step.

An alternative solution was thus found. Targets would walk following a predetermined path passing by known points so deviations from this path can be measured. Unfortunately, the time aspect of the measurement is lost, all that is known is the shape and position of the path originally taken by the pedestrian, and the positions of all the particles throughout the test. Nevertheless, a circular path was chosen to measure the deviation between these two elements. This shape is ideal because it makes the target walk in every direction and the deviation from the path can easily be computed by taking the difference between the radius of the circular path and the distance between the target's computed position and the centre of the circle. This deviation can be computed for every time step of the test and an average value of the error can be taken as the average tracking error.

However, determining the exact position of a tracked target with a particle filter is not straightforward. Indeed, as explained in Section 4.2.2, after each prediction, innovation and resampling step, the particles represent a probability density function for the position of the tracked pedestrian. All the particles having the same weight, the probability increases whenever multiple particles are at the same location, but no exact location can be given with a 100% certainty. This is why the absolute value of the deviation of each particle from the circular trajectory is computed independently before an average is computed for every time step. The final computation of this mean error , $\bar{\Delta}_{\boldsymbol{x}c}$, depends on the number of particles N_p, the number of time steps N_t, the position of the centre of the circular path and its radius ((X_0, Y_0) and R respectively), and the position of each particle at each time step $(x^i_{x,n}, x^i_{y,n})$. This is computed using the following formula:

$$\bar{\Delta}_{\boldsymbol{x}c} = \frac{\sum_{i=1}^{N_t} \sum_{n=1}^{N_p} \left| \sqrt{(x^i_{x,n} - X_0)^2 + (x^i_{y,n} - Y_0)^2} - R \right|}{N_t \cdot N_p}. \tag{5.1}$$

The absolute value is crucial to make sure that the deviations from the desired trajectories add up instead of cancelling themselves out when being opposite sides of the path. In the quality assessment and in the parameter tuning that will have to be performed, the aim naturally is to minimise this $\bar{\Delta}_{\boldsymbol{x}c}$.

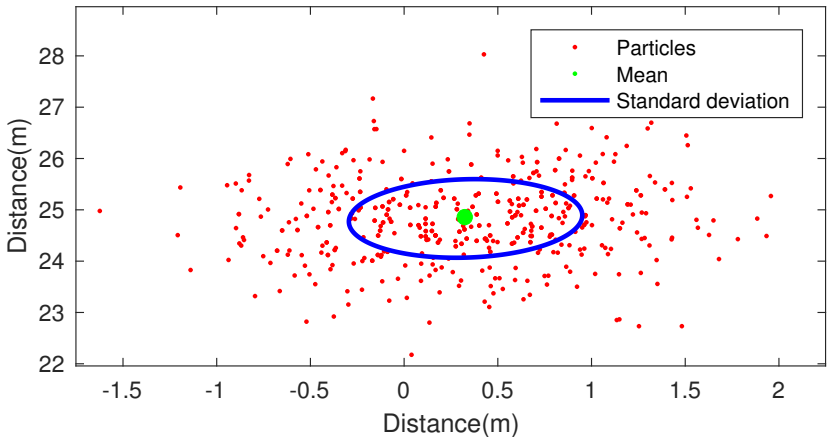


Figure 5.8: Representation used to show the PDF defined by the particles. The green dot represents the mean centre position of the particles while the blue ellipse represents the standard deviation around that centre. This representation is showing a particle filter after the resampling step and is thus composed of identical weight particles that might be overlapping each other.

In order to have a comprehensive visual representation of the behaviour of the particle filter's tracking, it was decided that, because only one target was involved in the accuracy tests, the probability density function shaped by the particles could be approximated by a normal distribution without loosing too much information. The mean position of the particles represents the most likely position of the target and the standard deviation is used to analyse how spread out the particles are. These elements will be graphically represented by a point and an ellipse, respectively, as shown in Figure 5.8. It is important to note that this ellipse representing the standard deviation, by the definition, contains approximately 68.2% of the particles.

5.4.3.2 Multiple target handling

The evaluation of a multiple target tracking is a more complicated task. With the available data, no tangible value could be found in order to exactly and systematically judge the accuracy and precision of the multiple object tracking. This is due to the fact that when a single target is tracked, all the particles are supposed to follow it, and thus, all the particles should be located around the tracked target, making the mean position of all the particles a good approximation of the target position. On the other hand, when multiple particle filters are tracking multiple targets, the particles can sometimes be divided into multiple clusters around multiple targets, making the mean position of all the particles a wrong approximation of the position of the tracked target. Thus, the assessment of the ability to cope with multiple targets will be done by manually looking at the evolution of the particle filters.

If the algorithm converges towards a single target, the particles will be mostly clustered around one position. If the algorithm is not capable of making the difference between targets, the particles will be equally distributed in multiple clusters around these multiple targets, or in the worst case, converge on the wrong target. By manually analysing these clusters, it can be determined if the algorithm correctly distinguish between the targets.

Chapter 6

First algorithm developed for single target tracking

In order to perfect the target tracking, it was decided to start by implementing a functioning single target tracking that would establish a base to work on when stepping up to multiple targets. The goal was to implement the simplest and most bare-boned version of a particle filter tracking algorithm that could easily be modified so new elements could be added and tested in a simple way. In order to do this, the most basic theory on multimodal data fusion and particle filtering was followed as to introduce as less assumptions as possible. The algorithm was developed using MATLAB, as it is the interface used to communicate with the radar and the camera detector, along with the "Computer Vision System Toolbox" dedicated to image processing.

There are multiple steps that have been followed to implement this algorithm and these will be cited below.

1. Retrieve the data from both sensors and allow them to communicate with the algorithm.
2. Choose a data fusion scheme adapted for different types of processed data.
3. Establish sensor error models for the camera and the radar so their observation error can be estimated.
4. Implement a particle filter to track the targets.
5. Optimise the various parameters.

After the workings of this first version of the program is explained, a series of tests discussed in the previous chapter will be performed. The results from these test will then be analyse and discussed as to identify the elements that could be improved by adding of modifying elements from this version.

6.1 Retrieving the data

The first challenge that is faced concerns the handling of the data coming from multiple sources simultaneously. The radar connects to a computer using a TCP/IP connection that, once established, cannot be interrupted nor delayed. This means that a processor core has to be dedicated to handling this connection because it always has to be able to receive and handle incoming signals from the radar instantly. Thus, this dedicated core can not perform any type of computation other than handling the radar's data and communicating with the other cores using a shared file system and multicore locks.

The camera connection is different, once a core establishes a connection, it can be left open while other computations occur and, when data is needed, a frame can be sent upon request with little delay. This means that, if only two cores are available for use in total, one can be dedicated to the radar connection, and the other can handle the camera connection as well as all the other computations such as the tracking. In an ideal scenario, three cores would be required: one for the radar connection, one handling the camera connection as well as the pedestrian detection and finally one core working on the tracking. In the current setup, a computer consisting of four cores was available. Thus, the three core implementation choice has been chosen.

6.2 Data fusion

In this section, all the possible data fusion architectures will first be explored and their possibility of implementation with the current setup and requirements will be assessed as well. Secondly, the justification of of the particle filtering choice related to the data fusion challenges will be outlined.

6.2.1 Data fusion architecture

For this thesis' purposes, the distributed architecture did not seem useful owing to the fact that it is better suited for large networks of sensors and only two sensors are used in this project. There were thus no clear benefits of implementing this method.

The direct centralised architecture was also rapidly dismissed because it would cause a significant bottleneck. This is due to the fact that there are two steps in the algorithm that take a significant amount of computation time, these are the visual pedestrian detection and the tracking itself. Having these two elements run in just one unique node was considered as a waste of computing time because although the visual pedestrian detection and tracking update at iteration I_n, for instance, have to happen sequentially, the pedestrian detection of iteration I_{n+1} can be done in parallel with the tracking update of iteration I_n, which would make the program much faster.

The autonomous centralised architecture was also abandoned after the choice of the tracking algorithm: the *particle filter*. Indeed, using this approach would mean performing separate tracking with the radar observations and the camera detections before fusing the results in the final node. This would mean that two tracking algorithms would have to run in parallel, which takes time, and would require finding a way to combine results in an effective way. This approach would also require to transform radar observations into detections via clustering methods, which was proven to give poor results. Running a single particle filter that would be able to fuse the data to perform the tracking all on its own was deemed a better solution.

Two possible options remain for the data fusion: the hierarchical architecture and the feature extraction based centralised architecture. Having only two sensors, it seemed simpler to use the feature extraction based centralised architecture only and thus to preprocess the data from the camera and the radar in order to return 2D coordinates and send these to the final node: the particle filtering. The data fusion scheme is depicted in Figure 6.1.

6.2.2 Data fusion approach

By examining the data fusion challenges that can be met during the fusion of a camera and a radar for the purpose of multiple target tracking in Section 3.3.2, two types of data challenges stand out: imperfections and inconsistencies. These two categories of data errors each have their own way of dealing with fusion. However, only one solution of data fusion is identical for

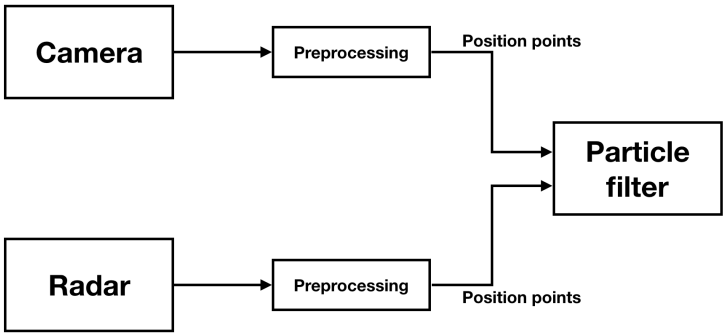


Figure 6.1: Data fusion scheme of the first developed algorithm.

both types of errors, namely the probabilistic data fusion approach based on the Bayesian filtering.

The fact that the previously chosen tracking and data fusion method, namely the *particle filtering*, falls into the category of common solutions for both data fusion challenges is confirming the good choice of this approach.

6.3 Sensor model

Defining observation error models of each sensor is the first step that has to be undergone to implement a particle filter. The sensor error model will influence how the incoming data is going to influence the particles. Although having a good model will increase the tracking precision, it is not crucial for it to be perfectly accurate because there are other ways to ensure the good functioning of the particle filtering. If the sensor model is slightly off, setting a higher process noise or increasing the number of particles are two methods that can be used to compensate for inaccurate estimations. Nevertheless, these methods do not come without inconveniences. Increasing the process noise will make the PDF more spread out, so it will be more difficult to pinpoint the position of the target and increasing the number of particles will slow down the computations. For these reasons, even if it is possible to have a working program with a basic sensor model, it was decided to be as accurate as possible with the sensor models.

6.3.1 Camera model

The camera's observation error only applies to the pedestrian's estimated position, because this is the information that will be used by the tracking algorithm.

When a target is detected, as described in Section 5.1, the *non-maximum suppression* method computes bounding boxes that are positioned as to be accurately centred on the target. Even with targets far away, the bounding box is mostly always correctly centred on the pedestrians. The angle at which the target is positioned compared to the centre of the frame can thus be computed with great precision and does not depend on how far the target is from the sensor.

The range error is completely different. This distance is computed using the pinhole model described in Section 3.1.2.1 by using the height of the bounding box. This can cause inaccuracies because (1) an assumption of the target's height has to be made and (2) even though the bounding boxes are centred properly, their sizes compared to the targets' can vary from frame to frame which hinders efforts to link bounding box sizes to targets' ranges.

The strong hypothesis was made that the bounding box around a pedestrian is always of a

height of $H = 2m$ because the average human height is on average $1.70m$[1] and the bounding boxes often overestimate the true size of pedestrians. This model thus assumes a $30cm$ overestimate by the bounding box. This has the negative effect that the distance of significantly taller or shorter people would be quite difficult to estimate with this approach.

The bounding boxes do not always start at the target's feet and stop at their heads, but often miss these boundaries by a few pixels. This is due to the fact that the *non-maximal suppression* is not constant and sometimes less accurate bounding boxes get selected. It was also noticed that the bounding boxes' sizes are increasingly overestimated as the distance of the pedestrians increase.

In the algorithm developed by Dollár, Belongie and Perona, as explained in Section 5.1.1, there is only a limited number of bounding box sizes that can be tested. The smallest bounding box is of a height of 100 pixels and this height is scaling up with a scale of $\delta_h = 2^{\frac{1}{10}} \approx 1,07$. This means that, with a fixed estimate for a target's bounding box height H, the computed range can only take a limited amount of values due to the limited number of bounding box sizes. The range difference Δ_D between two consecutive bounding boxes (of size h and $\delta_h \cdot h$) can be computed using the model described in Section 3.1.2.1 and Figure 6.2.

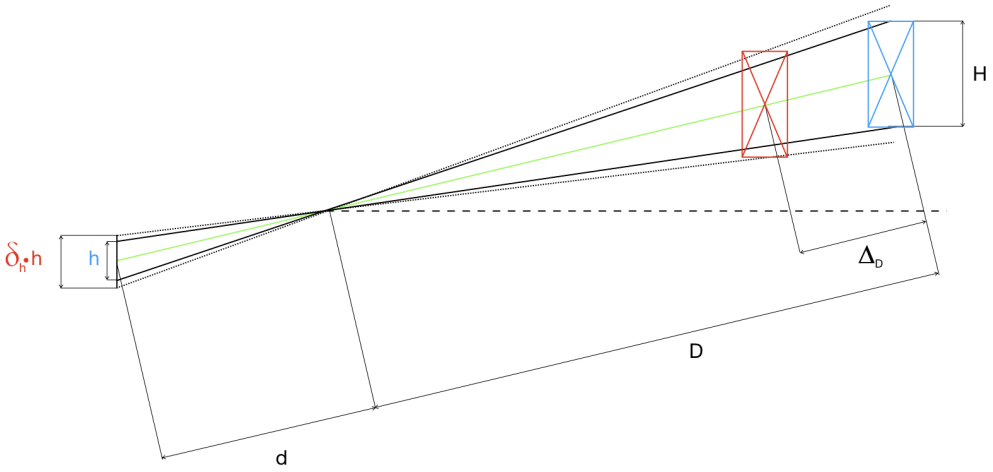


Figure 6.2: Distance difference (Δ_D) between two bounding boxes (in red and blue) of size h and $\delta_h \cdot h$, representing a target of height H.

The assumption that the height of the centre of the bounding box (c) is the same for both detections is made. If c is constant then $d = \sqrt{f^2 + c^2}$ is too. With the equations from the pinhole model and knowing that H is constant, it can thus be written that:

$$D = \frac{H \cdot d}{h} \qquad D - \Delta_D = \frac{H \cdot d}{\delta_h \cdot h} = \frac{D}{\delta_h}. \tag{6.1}$$

And from this we have

[1] In Europe.

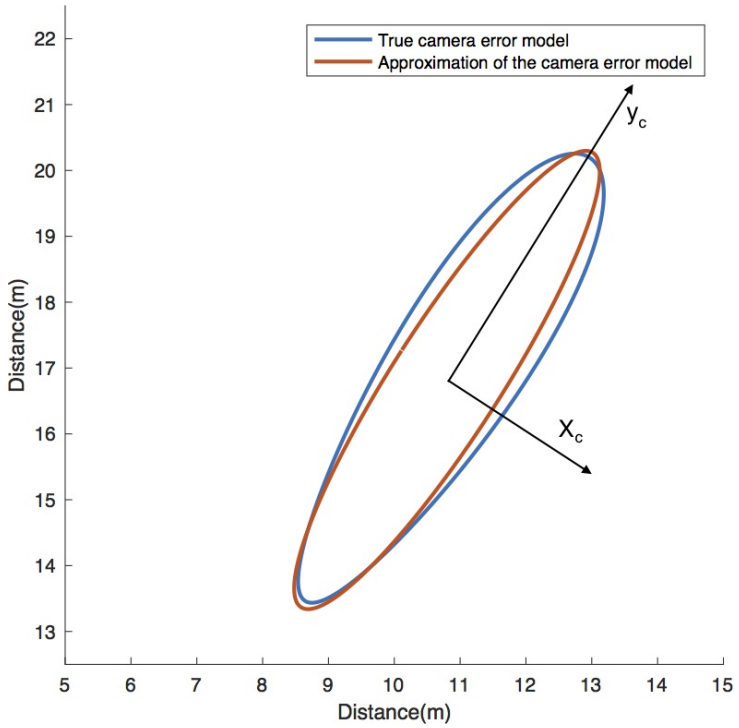


Figure 6.3: Representation of the actual and estimated PDFs for the camera error model, with the axes of the ellipses representing the standard deviations of the independent elements of the covariance matrix. As it can be noticed, the true camera error model is greater the further it is from the centre of the map.

$$\Delta_D = D\left(1 - \frac{1}{\delta_h}\right) \approx 0,067 \cdot D. \tag{6.2}$$

In the case Δ_D would be computed for a scaled down bounding box of size $\frac{h}{\delta_h}$ the new distance would be $D + \Delta_D$ and the formulas above can be used to find that $\Delta_D = D\,(\delta_h - 1) \approx 0.072 \cdot D$. It could thus be assumed that these formulas would dictate the maximum error from the distance estimation, however after this model was tested, much greater errors were observed. The reason behind this difference is that the pedestrian detector is not perfect and sometimes outputs bounding boxes that are multiple increments bigger than they should be. Nevertheless, the Formula 6.2 states that the error in distance should theoretically increase linearly following D. The factors 0.067 or 0.072 are not really adequate due to the errors mentioned above, but it is reasonable to assume that the actual error still evolves linearly with the distance.

The camera's inaccuracies are measured in the angle and range measurements, and thus, the error will likely follow a multivariate normal distribution in the polar space, which is not how the data is represented in the program. Nevertheless, because the angle inaccuracy is very small compared to the range inaccuracy, the shape of this distribution will closely resemble a regular multivariate normal distribution in the Cartesian space, as can be seen in Figure 6.3. It was thus decided that a conversion to polar coordinates was not useful and the estimation in the Cartesian coordinates (x_c,y_c) was used.

As explained previously, the azimuth angle determined by the camera remains constant as the azimuth distance grows. This is why it was decided to model the error in the tangential direction, x_c, with a constant called γ_1 which represents the variance of this error along that axis.

As for the range inaccuracies, it was decided to model the radial error using the standard deviation $\sigma_d = \sqrt{\gamma_2} \cdot D$, where the value of γ_2 is given through the tests described in Section 5.4. γ_2 is put under a square root here because Δ_d represents the standard deviation of the error but the final error model is described with a covariance matrix which means that the value σ_d^2 will be used to define the error model.

Taking all the information above into account, a final camera error model can be established. It follows a multivariate normal distribution whose covariance matrix is be determined along the axes x_c and y_c as shown in Figure 6.3. This covariance matrix is defined as follows:

$$\boldsymbol{\sigma}_c^2 = \begin{bmatrix} \gamma_1 & 0 \\ 0 & \gamma_2 \cdot D^2 \end{bmatrix}. \tag{6.3}$$

6.3.2 Radar model

The radar model is simpler to estimate because it was noticed that, in the range of distances that are used, the noise and precision are almost completely independent of the target's range. Although it is stated in the device's manual [90] that the azimuth angle has a resolution of 0.1°, which should theoretically influence the precision of the targets positioning in the azimuth plane, it was noticed that this was not the case with the actual data. However, errors of up to 4 meters in all directions were seen when performing various tests, but in the ranges that were explored ($10m$ to $50m$), no significant correlation could be observed between the distance of the target and these errors. The precision of the observations depends rather on factors such as the environment, where the tests are made, and various obstacles in the field of view.

The radars error model, as mentioned is thus quite straightforward, in that it can be represented by a symmetrical normal distribution. Although this distribution does not vary as a function of any quantifyable element, it was decided to make the weight of the observations a factor in the error model's covariance matrix as represented in Equation 6.4. The elements of the covariance are inversely proportional to the intensity I_r of the recorded signal from the radar. The effect of this is that radar detections with high weights will only and highly influence particles close to them while lower weights will have less direct influence. The parameter δ will be fine-tuned in Section 6.6.

$$\boldsymbol{\sigma}_r^2 = \begin{bmatrix} \frac{\delta}{I_r} & 0 \\ 0 & \frac{\delta}{I_r} \end{bmatrix} \tag{6.4}$$

6.4 Implementation of the particle filter

As mentioned, the tracking was chosen to be done using a particle filter, which is essentially works with group of particles, each containing a state estimation in the form of attributes, that will be updated recursively. These updates occur whenever one of the sensors outputs new data. This data is then independently used as a new state observation and causes the particles' state estimations to update in a certain way depending on the sensor. In the case of this implementation, the filter in itself is a structure that stores the time stamp of the last update, the last time stamp at which the camera detected a target, and all the particles along with their state information.

The particles contain two attributes: $\boldsymbol{x}_P = [x_x, x_y]$ and $\boldsymbol{x}_v = [v_x, v_y]$. These are the particles' positions and their speeds and they are used to define the particles' states. This data is expressed following the coordinates illustrated in figure 6.4.

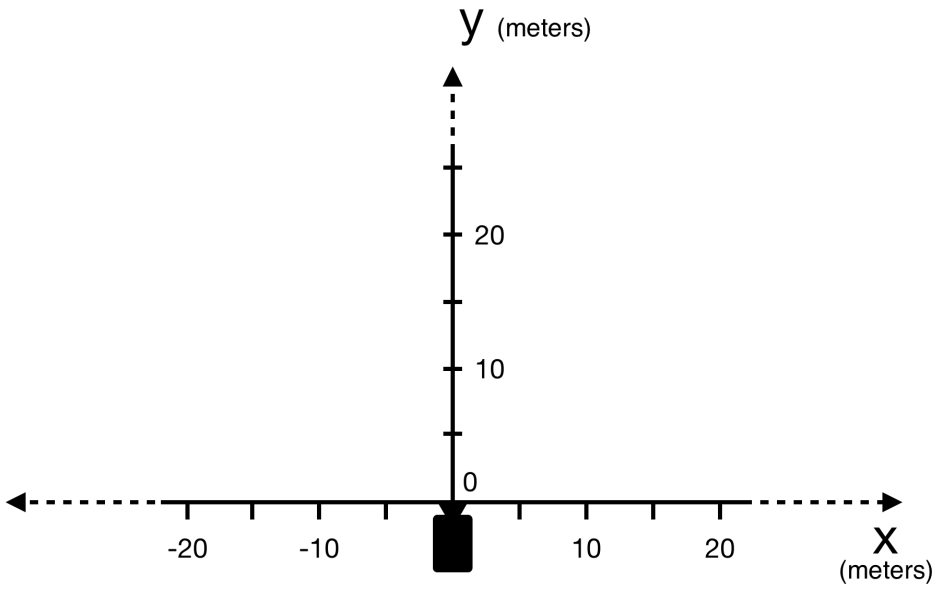


Figure 6.4: Axis used to measure targets' positions with the device located in $(0, 0)$ meters and pointing in the positive y direction.

6.4.1 Introducing a target to follow

The tracking algorithm only starts when a target first enters the space scanned by the sensors. Due to the extremely large numbers of false positives obtained by the radar, it was decided to wait for the pedestrian detection algorithm applied on the camera data to detect a new target. When a pedestrian is found, in order to track it, a new particle filter is created as follows. The pedestrian's position is estimated using the pinhole model and a multivariate probability density function is created using the camera's error model. N_p particles are then created taking random points generated following this PDF as their positions $\boldsymbol{x}_P$. Their speeds $\boldsymbol{x}_v$ are initially estimated at $0m/s$ because no information is available on it at the first iteration. The state of the n-th particle of iteration i is thus represented as

$$\boldsymbol{x}_n^i = [\boldsymbol{x}_{P,n}^i \, , \, \boldsymbol{x}_{v,n}^i] = [x_{x,n}^i \; x_{y,n}^i, \; v_{x,n}^i \; v_{y,n}^i]. \tag{6.5}$$

6.4.2 The different updating steps

At each iteration, the particles will go through the different steps discussed in Section 4.2.2 in order to update. The implementation of these different steps will be detailed below.

6.4.2.1 Prediction step

The prediction step is the first operation to be performed every time a sensor sends data. This detection comes with a time stamp that can be compared with that of the last filter update. This time difference, ΔT, can then be used to predict all the targets' new positions, $\boldsymbol{x}_{P,n}^i$, using the common assumption that targets have constant velocities between iterations.

However there is a certain level of uncertainty that comes with this dynamic model, and it is added to this step by introducing a process noise, $\boldsymbol{\epsilon}_n^T$, which as described formally below, is a normally distributed random error that is added to the estimated position computed with the dynamic model. The mean of this distribution is $(0, 0)$ and its covariance matrix is $\boldsymbol{\sigma}_p^2(\Delta T)$ which is described below. This error distribution exists for two reasons. The first is that the speed computed might be inaccurate at the time of the measurement, which can be partially compensated with this error distribution. The second reason comes from the constant speed assumption, which is a simplification of the real dynamics of the targets. The target's speed can change between measurements and this variation has to be accounted for. Because the targets have a finite acceleration, it is reasonable to say that the smaller the time gap ΔT between observation, the smaller the error can be. For this reason it was decided that the independent elements of the covariance matrix would be proportional, by a parameter α, to ΔT. The value of α is set empirically through rigorous testing that will be discussed later in Section 6.6. Thus, $\boldsymbol{\sigma}_p^2(\Delta T)$ is defined by:

$$\boldsymbol{\sigma}_p^2(\Delta T) = \begin{bmatrix} \alpha\Delta T & 0 \\ 0 & \alpha\Delta T \end{bmatrix}. \tag{6.6}$$

With all the parameters defined, the position prediction is thus performed as follows.

$$\forall\ n \in [1, 2, ..., N_p] :\ \boldsymbol{x}_{P,n}^{i|i-1} = \boldsymbol{f}_n \left(\boldsymbol{x}_n^{i-1}\right)^T + \boldsymbol{\epsilon}_n^T \qquad \boldsymbol{\epsilon}_n \sim \mathcal{N}\left((0,0), \boldsymbol{\sigma}_p^2(\Delta T)\right) \tag{6.7}$$

$$\boldsymbol{f}_n = \begin{bmatrix} 1 & 0 & \Delta T & 0 \\ 0 & 1 & 0 & \Delta T \end{bmatrix} \tag{6.8}$$

The last computation that is made during the prediction step is the one regarding the speed measurements. Under the constant speed assumption, this is quite straightforward as shown next.

$$\forall n \in [1, 2, ..., N_p] :\ \boldsymbol{x}_{v,n}^i = \frac{d\boldsymbol{x}_{P,n}^i}{dt} = \frac{\boldsymbol{x}_{P,n}^i - \boldsymbol{x}_{P,n}^{i-1}}{\Delta T} \tag{6.9}$$

No error model has been applied to the computed speed, because it will only be used for the prediction step of the next iteration. As shown in Equation (6.7), an error model is already applied at that step, and adding one for the speed would not have much sense.

6.4.2.2 Innovation step

This step consists on weighting the particles proportionally to how close the prediction is to the target's position or, more precisely, to the observations' positions. In order to do so, a PDF $\mathcal{P}(\boldsymbol{x}_n^i|\boldsymbol{z}_k^i)$ is established around each observation $\boldsymbol{z}_k^i$ under the form of a multivariate normal distribution whose covariance matrix's $(\boldsymbol{\sigma}_s^2)$ characteristics depend on the type of sensor and its error model. The weighting function, for each particle, sums up the probabilities $\mathcal{P}(\boldsymbol{x}_n^i|\boldsymbol{z}_k^i)$ of each of the measurement. It is thus defined, with N_o^i being the number of observations, as

$$\boldsymbol{w}_n^i = \sum_{k=1}^{N_o^i} \mathcal{P}(\boldsymbol{x}_n^i | \boldsymbol{z}_k^i) \tag{6.10}$$

The probability $\mathcal{P}(\boldsymbol{x}_n^i | \boldsymbol{z}_k^i)$ introduced in Equation 6.10 will be different if the update comes from the camera or from the radar. This is where the sensor models discussed in Section 6.3 come into use.

6.4.2.3 Resampling step

The goal of this step is to select, with replacement, N_p particles from the previously weighted ones so that, by giving all of them the same weight, they reproduce the original PDF as accurately as possible. Various methods of resampling exist but although they do not all output the exact same particles, the differences between them are not significant enough to justify the choice of a method other than by its computational speed. As mentioned in [54], systematic resampling is the fastest method, especially when the number of particles gets larger. This method is implemented as shown in Algorithm 1.

Data: Particles P^i arranged in ascending order of weights $\boldsymbol{w}_n^i$ which are normalised as to add up to 1.
Result: Resampled particles R_i.
$CS = \left[0,\ \boldsymbol{w}_1^i,\ \boldsymbol{w}_1^i + \boldsymbol{w}_2^i,\ ...,\ \sum_{n=1}^{N-p} \boldsymbol{w}_n^i = 1\right]$;
$V \sim U\left[0, \frac{1}{N_p}\right[$;
$n = 0$;
$k = 0$;
while $n < N_p$ **do**
  if $V > CS_k$ && $V \leq CS_{k+1}$ **then**
    $R_n = P_k$;
    $n = n + 1$;
    $V = V + 1/N_p$;
  else
    $k = k + 1$
  end
end

Algorithm 1: Systematic resampling algorithm for N_p particles described in [54].

This algorithm is cleverly designed as to select multiple times highly weighted particles while also selecting a sufficient amount of less highly weighted ones in order to keep a balanced distribution that, in the end, will resemble the original PDF.

6.5 Parameter optimisation

The particle filtering algorithm described above depends on multiple parameters that have to be calibrated in order to perform an optimal tracking. The parameters that can be adjusted were the following:

- α: linked to the propagation of the prediction step (Equation 6.6).
- γ_1: fixed element of the covariance matrix in the camera error model (Equation 6.3).

- γ_2: linked to the range dependant element of the covariance matrix in the camera error model (Equation 6.3).
- δ: linked to the radar's error model (Equation 6.4).
- N_p: the number of particles.

In order to optimise each of these parameters, the data from four different tests detailed in Section 5.4 are used as to minimise the chance of calibrating them for one specific scenario to the detriment of all the others. From the six tests in Table 5.1 the ones that were not used for the optimisation were the tests using only one of the two sensors.

The defining element that is used throughout the calibration is $\bar{\Delta}_{\boldsymbol{xc}}$, whose computation is detailed in Equation (5.1). This value reflects how far off-course the particles of the filter are throughout the tracking, thus minimising this value is a way to ensure an optimal tracking. To eliminate risks of over-fitting, the final $\bar{\Delta}_{\boldsymbol{xc}}$ is computed as the mean of all values computed throughout all the iterations of the four tests.

In all the tests that were conducted, the resulting $\bar{\Delta}_{\boldsymbol{xc}}$ obtained are very uneven, this is due to the probabilistic nature of particle filtering, in some cases the target is lost and the error thus increases dramatically. These mishaps show up in the data as spikes in the values of $\bar{\Delta}_{\boldsymbol{xc}}$. It was chosen to show them in the graphs because it was not practical to show only the best case scenarios for each variable. By keeping these outliers, regions can be found where these errors are less likely to happen. The results of the optimisation of all the variables listed above will be discussed below.

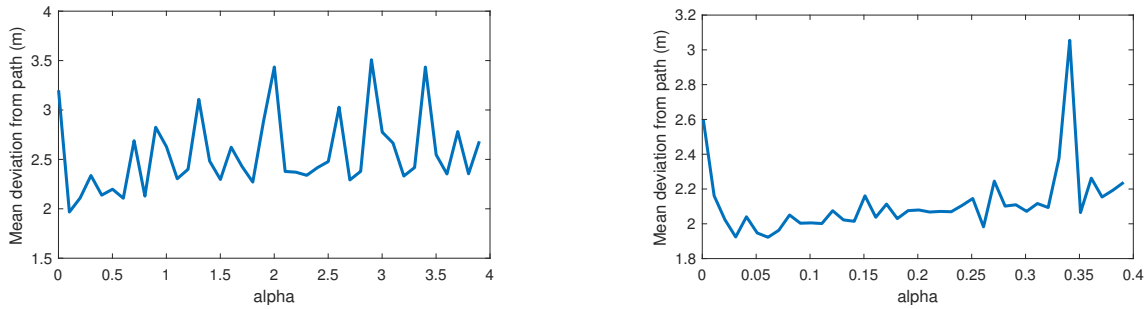
6.5.1 Optimisation of α

Finding a good value for α is important because it will reflect the accuracy of the dynamic model. As described in Section 6.4.2.1, independent diagonal elements of the covariance matrix representing the error in the prediction step are equal to $\alpha \cdot \Delta T$. Before launching the tests, it was noticed that, in general, when α was set to a value higher than 4, the particles would not converge on any target and essentially be placed randomly. For this reason $\alpha = 4$ was the highest value set for the tests. The minimum value was set at 0.0001 to represent a very small error and an equally small variation of the dynamic model. Results of this test can be seen in Figure 6.5a. From this, it can be noticed that small values of α tend to give better results so a test on smaller values was performed and results are shown in Figure 6.5b. Although it may seem from this second graph that values of alpha between 0.02 and 0.06 are optimal, it was later revealed that for these small values of α, in the scenario with occlusions, the target was completely lost by the algorithm. This loss of target meant that the particles stayed close to where the target was last detected, which was correctly positioned on the circular path. Thus $\bar{\Delta}_{\boldsymbol{xc}}$ did not get too significantly big and because the other tests performed really well with these values, the error was ultimately not visible in Figures 6.5a nor 6.5b. After making a controlled test with the occlusion test, a value of $\alpha = 0.25$ was finally selected.

This value means that, with a mean value $\Delta T \approx \frac{1}{10} = 0.1s$, using the formula expressed in Equation (6.6), it yields an independent standard deviation of $0.16m$.

6.5.2 Optimisation of δ

Having an appropriate value for δ is important so that the radar's error model can be estimated correctly. Because it will be divided by the weights coming from the radar, who range from 20 to 50, values of δ between 1 and 100 were taken such that standard deviations ranging from $\sqrt{1/50} \approx 0.14m$ up to $\sqrt{100/20} \approx 2.24m$ could be tested. These values were chosen so multiple



(a) (b)

Figure 6.5: Variation of $\bar{\Delta}_{xc}$ as a function of α. (a) α ranging from 0 to 4,. (b) α ranging from 0 to 0.4

values starting at a very small radar error[2] up to high radar errors could be experimented. Results, shown in Figure 6.6, indicate slightly and quite steadily increasing values of $\bar{\Delta}_{xc}$ as δ increases above 30. It is also visible that, below a value of 5, the error increases as well. The value of this parameter was chosen to be set to $\delta = 30$ which, on average tends to give more stable results than smaller values.

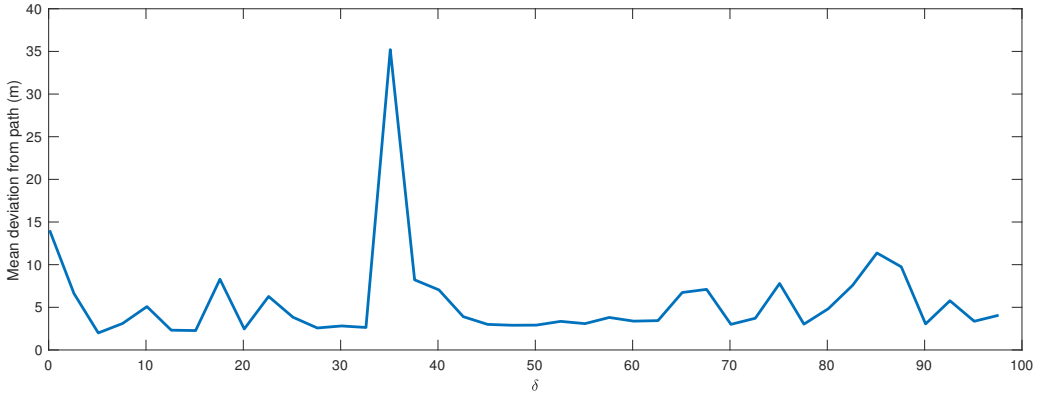


Figure 6.6: Evolution of $\bar{\Delta}_{xc}$ as a function of δ

This value defines the covariance of the radar's error model and yields a standard deviation ranging from $0.77m$ to $1.22m$ depending on the radar intensity signal.

6.5.3 Optimisation of γ_1 and γ_2

The two variables γ_1 and γ_2 are linked to the camera's error model and their value will influence the effect of the camera's detections on the tracking of the pedestrians. Because both of these values influenced the same factor, namely, the camera error model, it was decided to optimise them together. The range that was chosen for both values was defined as to test standard deviations ranging from $0.1m$ to $2.5m$ in both axes, which, again, represent respectively a very precise sensor and an imprecise one. The range of γ_1 is thus $0.1^2 = 0.01$ to $2.5^2 = 6,25$. As for γ_2, because it is not the only factor influencing the standard deviation, the range D has to be taken into account. It was decided that values of γ_2 would be chosen such as the standard deviation would also vary from $0.1m$ to $2.5m$ for a distance of $D = 20m$. With the standard deviation being computed as $\sigma = \sqrt{\gamma_2} \cdot D$, this means that γ_2 should range from $\gamma_2 = \left(\frac{0.1}{20}\right)^2 = 2.5 \cdot 10^{-5}$

[2]Approximately 5 times more precise than what the supplier claimed the radar was capable of.

up to $\gamma_2 = \left(\frac{3.5}{20}\right)^2 = 0.0306$.

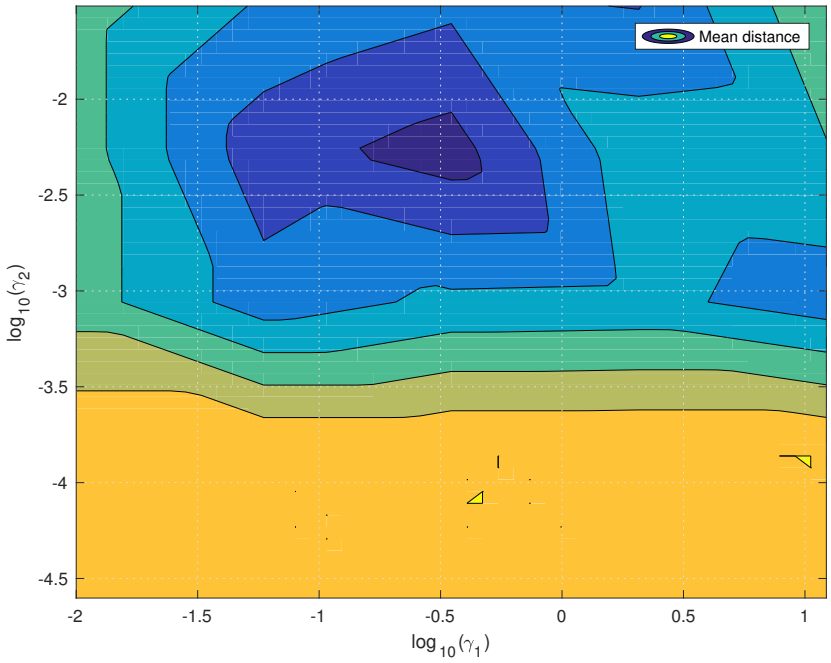


Figure 6.7: Evolution of $\bar{\Delta}_{\boldsymbol{xc}}$ as a function of γ_1 and γ_2. Dark blue represents small values, and yellow large values.

After the first tests, it was noticed that results varied the most for lower values of γ_1 and γ_2, such that, as shown in Figure 6.7, a logarithmic scale was chosen to represent them best. The values $\gamma_1 = 10^{-0.5} \approx 0.316$ and $\gamma_2 = 10^{-2.3} \approx 0.005$ gave the best results.

These parameters at a $20m$ distance, yield a covariance matrix for the cameras error model of

$$\boldsymbol{\sigma}_c^2 \approx \begin{bmatrix} 0.316 & 0 \\ 0 & 2 \end{bmatrix} \tag{6.11}$$

This finally give standard deviations of $0.56m$ and of $1.41m$ along the tangential and the radial axis (with respect to the camera) respectively.

6.5.4 Optimisation of N_p

Choosing the number of particles is be very important for both the speed and the accuracy of the algorithm. While a high number of particles may be beneficial for the tracking algorithm's accuracy, it may require too much computing power to run efficiently. For this reason, the effect of this number is observed in Figure 6.8. As it could be expected, the computing time increases linearly with the number of particles. However, the accuracy reaches an asymptote relatively quickly. As a matter of fact, once the number of particles reaches 500, the precision stops increasing and levels out. This is why the value of $N_p = 500$ was chosen.

As the computational time of an update with 500 particles is approximately $2ms$, this number of particle does not influence at all the real-time aspect of the tracking as the sensor data comes

in at an average of $10fps$ which allows the algorithm to perform the tracking during an average of $0.1s$.

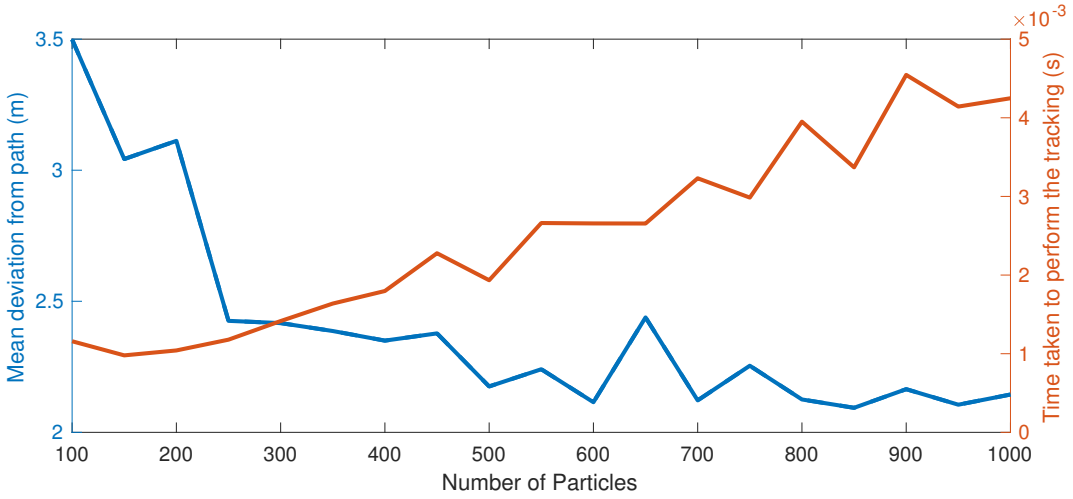


Figure 6.8: Evolution of $\bar{\Delta}_{\boldsymbol{xc}}$ and time taken as a function of the number of particle N_p.

6.6 Results and observations

This section aims to present and analyse obtained results with the optimised parameters. These results will be discussed in the next section. Every test presented in Section 5.4 have been applied to the tracking algorithm. Trajectory results of these tests can be found in Figure 6.9. Each of these figures will be analysed below. On Table 6.1, the mean deviation $\bar{\Delta}_{\boldsymbol{xc}}$ of the trajectory described by the Equation (5.4.3) is calculated for every test. Also, as an example, Figure 6.10 shows all the particles of a trajectory drawn on the "no challenge" test.

In these figures, the frequency at which the PDFs are drawn are not representative of the frequency of the updates. For better visibility, not all the PDFs were drawn, but depending on the figure, only one in two to four updates are displayed.

It is important to note that the tests (a), (e), and (f) are all done with the same data. The test without radar data is the "no challenge" test without the radar observations and the test without camera detections is the "no challenge" test done by taking into account the first detection by the camera (in order to start the tracking) and then only using radar data to track it.

(a) No challenge This test is the most basic one representing an ideal situation. It can be observed that this version of the algorithm generally follows the ground truth trajectory. However, some small deviations are present as well as one significant deviation on the top left corner yielding $\bar{\Delta}_{\boldsymbol{xc}} = 0.8254m$.

(b) Occlusions The same small deviations as in the first test can be observed here as well. Additionally, the path is completely deviating from the desired trajectory behind the obstacle but finds the right trajectory after the occlusion, which causes the mean deviation to be relatively high: $\bar{\Delta}_{\boldsymbol{xc}} = 1.9623m$.

(c) lighting variations The path is over a shadowy and sunny exposition which causes the same small deviations as in the first test can be observed here and $\bar{\Delta}_{\boldsymbol{xc}} = 0.6884m$.

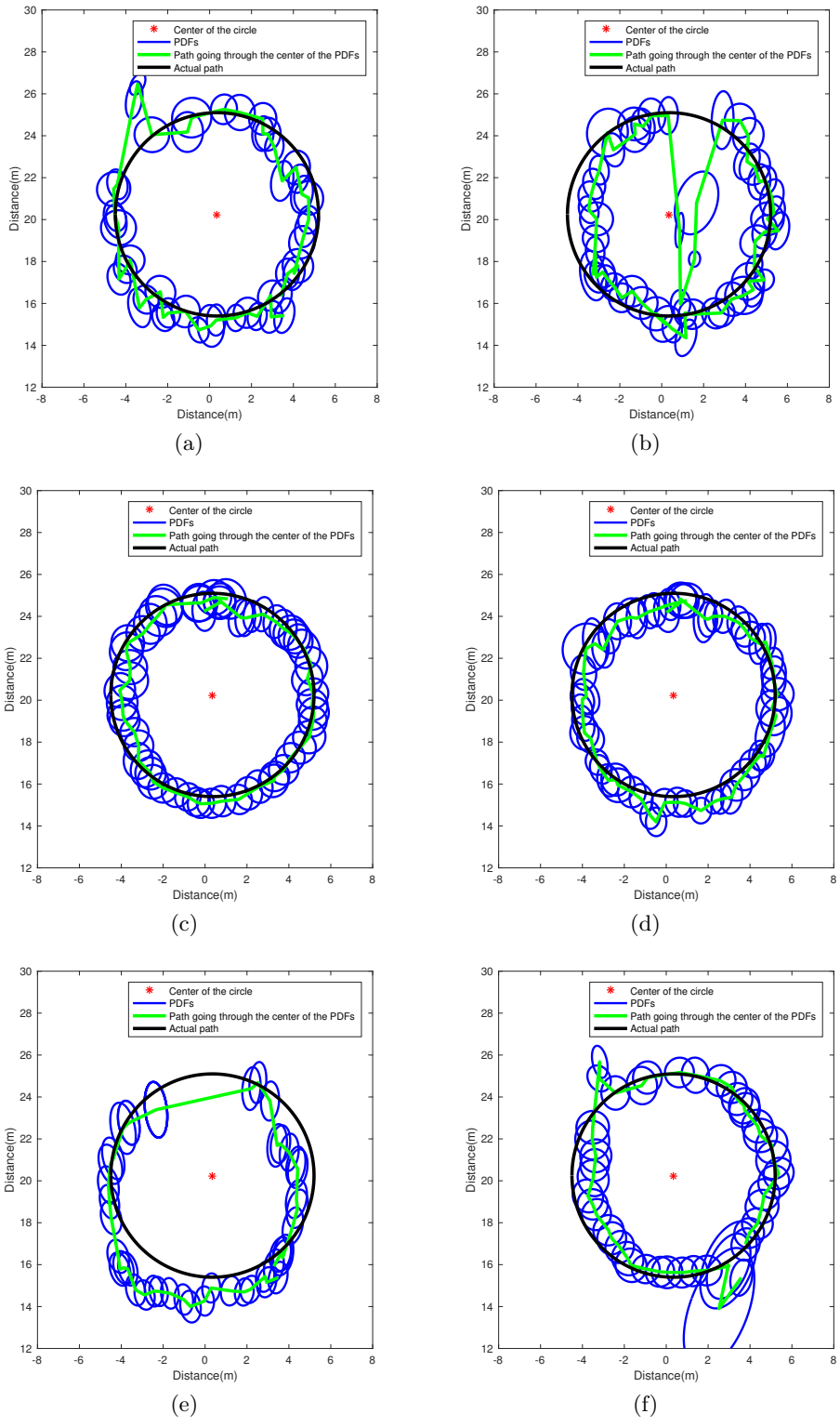


Figure 6.9: Figures of the different tests. (a) No challenge (ideal situation): 5m radius circular path 20m away from the device. (b) Occlusions: 5m radius circular path around a 40cm radius tree 20m away from the device. (c) Lighting variations: 5m radius circular path 20m away from the device partially in the sun. (d) Radar noise: 5m radius circular path next to moving bushes 20m away from the device. (e) No radar data: 5m radius circular path 20m away from the device. (f) No camera data: 5m radius circular path 20m away from the device.

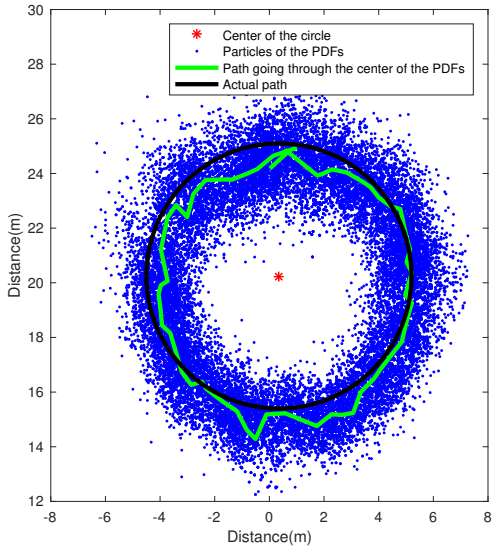


Figure 6.10: Particles from the basic test (no challenge) of the first algorithm.

(d) Radar noise Again, the same small deviations as in the first test can be observed here. Additionally, the majority of the ellipses are vertical and $\bar{\Delta}_{\boldsymbol{xc}} = 0.6789m$.

(e) No radar In addition to the small deviations, two observations can be made. First, surprisingly, no camera detections were found at the top of the circle. Second, it can be observed that, in the lower part of the figure, the path takes a slightly squared shape. These observations lead to $\bar{\Delta}_{\boldsymbol{xc}} = 1.1256m$.

(f) No camera Camera data was used only for the first detection, and left unused for the rest of the test. The same small deviations as in the first test can be observed here and $\bar{\Delta}_{\boldsymbol{xc}} = 0.7328m$.

Test	$\bar{\Delta}_{\boldsymbol{xc}}$
No challenge	0.8254
Occlusions	1.9623
Lighting variations	0.6884
Radar noise	0.6789
No radar data	1.1256
No camera data	0.7328

Table 6.1: Tests results of the first algorithm.

6.7 Discussion

This section aims to discuss the results and the abnormalities found in the tests after the parameter optimisation. Two main observations have arisen from this analysis: (1) sudden large deviations from the ground truth and (2) a squared shaped path as well as vertical ellipses on the "camera only" test.

6.7.1 Sudden deviations from the path

The first thing that seizes the eye when looking at the results in Figure 6.9 is the fact that, sometimes, the path seems to jump off course without warning. This is mainly the case in test

(b). It is thought that these errors are mainly caused by one element: the relatively large value of α.

Between two usual particle filter updates, a person walking at an average human speed of $1.4m/s$ is expected to move $14cm$. As stated above, for a usual step, the dynamic model's error is as high as $16cm$ in both axes. This means that the dynamic model must not be able to estimate correctly the next position of the target, thus needing such a large error to hope that some particles end up in a correct location. This is illustrated in all the tests by the fact that the ellipses representing the standard deviation of the PDFs are always very large, showing standard deviations of the order of one meter. The fact of having large predictions makes the particles more sensitive to radar noise. Indeed, if the radar outputs a false positive, even distant from the trajectory, it can cause a few particles to end up close to this erroneous observation and, through the resampling, force a large portion of the particles to converge towards it.

This is especially visible in test results (b) in Figure 6.9b. The occlusion (a tree) causes the target to be undetectable by both sensors, but, because of the radar updates, particles are continuously moving towards the source of the radar noise, which is induced by the tree. Fortunately, when the camera performs the update, it sets the particles back on track.

A counter-example can be found on test (e) and, thus, in Figure 6.9e. When only the camera tracks the pedestrian, a loss of detection for a certain duration caused the particle filter not to be updated for a moment. An interesting behaviour compared to when the radar is used can be noticed here. When the particle filter did not receive any camera detections at all for a moment, the particle filter did not receive any other data. This had the beneficial effect of keeping the particle filter where it was last updated instead being led towards false positive observations causing sudden deviations from the trajectory.

6.7.2 Squared shaped path and vertical ellipses

On the tracking results of the camera only, three observations have been made: a loss of detections on the higher part, a squared shaped trajectory on the lower part, and vertical ellipses representing vertical PDFs. The loss of detections is already mentioned above. Thus, the squared shaped path and the vertical ellipses will be discussed here-under.

Squared shaped path On the lower part of the trajectory, the drawn path seems to follow a horizontal line with two sharp angles on both edges, thus giving a squared shape to the lower part of the path. This effect was attributed to the poor range estimation of the pinhole model, and the fact that the bounding boxes come only in specific predetermined sizes which makes the task of estimating a precise distance even more challenging.

Vertical ellipses The vertical ellipses happen because of the error model of the camera and this effect is to be expected by looking at the influence of γ_1 and γ_2. As mentioned previously, the values found for these parameters cause the error model to have a much larger standard deviation in the radial than in the tangential direction.

Chapter 7

Improved algorithm for multiple target tracking

After the development of the first algorithm several improvements have to be made. Indeed, the first algorithm was developed based on the preliminary choices and implemented following theoretical models. However, obtained results, as discussed in the last chapter, were not satisfactory enough. Moreover, handling multiple targets being a necessary feature, is an element that has to be added. A new version of this algorithm was thus developed based on this thesis' authors' scientific intuitions, knowledge, and researches.

As intended, the first algorithm was a basic implementation that could easily be modified and on which new elements could be added. In this chapter, the modifications that were made are justified and explained. Every other aspect of the algorithm that is not mentioned in this chapter remains the same as in the first developed algorithm. Finally, results and observations will be presented and discussed.

7.1 Handling a varying number of pedestrians

The first algorithm developed is only able to track one person. This choice had been made for programming simplicity. One of this improved algorithm's goal is to handle multiple targets This includes being able to differentiate between multiple pedestrians, as well as handling the addition of new targets to track and the removal of old ones.

Handling multiple pedestrians is a challenging operation. Bearing in mind that only the camera is able to positively detect pedestrians, the simplest solution would to create a new target to follow whenever a new person is detected by it. This target, along with the others would be tracked using observations from both sensors leaving the particle filters to keep track of them[1]. A particle filter belonging to a target would then be deleted whenever it is lost by the camera. However, this naive method has many flaws.

Firstly, some pedestrians are detected multiple times by the camera on the same frame which would cause the creation of unnecessary particle filters.

Secondly, ambiguities of sensor data might occur. These ambiguities often happen whenever radar noise is intense and drives the particles along the wrong trajectory. This noise can be caused by surrounding movements but also by other pedestrians. This last mentioned perturbation can, for instance, drive multiple particle filters to converge on the same trajectory thus leading to the loss of the initial targets.

[1] As stated earlier, it is in the *particle filter*'s nature to filter out non-corresponding observations (the furthest ones) and to take into account the corresponding ones only (the closest ones).

Thirdly, pedestrians are not necessarily detected by the camera at each and every frame. The removal of a particle filter would then be inappropriate in case of non-detection by the camera.

In order to keep track of the pedestrians, it is figured that the most robust method is to be able to identify them independently at each possible update, based on their appearance. With the radar updates, this operation is not feasible as it only gives weighted detections with no other information than their locations. On the other hand, camera updates can be useful. Some features can be used in order to determine an appearance model of a pedestrian. In this iteration, the mean RGB vector (defined below) of a certain part of the detected window was chosen to represent the appearance of a pedestrian. This area has been chosen in order to represent the pedestrian's appearance model with relatively high certainty. By manually analysing detections, this area was established to be the portion of the bounding box defined by $[\frac{1}{3}; \frac{2}{3}]$ horizontally and $[\frac{1}{5}; \frac{7}{10}]$ vertically, with box axis pointing towards the right and downwards. An example image is shown in Figure 7.1. The mean RGB vector of a pedestrian k (particle filter k) is represented as $C_k = [C_k^r, C_k^g, C_k^b]$ and is calculated by performing the mean computation of each channel r, g, b (red, green, blue) for every pixel of the area. Each channel is given a value between $[1; 256]$.

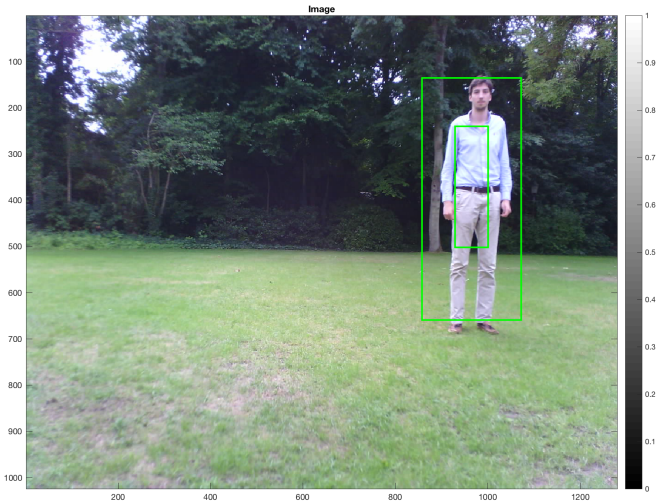


Figure 7.1: The outer green bounding box is the detected bounding box. The inner green rectangle is the area that is taken into account for the RGB mean calculation.

In order to distinguish between appearance models, a comparison function is created to determine if two camera detections belong to the same pedestrian or not. For this purpose, a basic euclidean distance between RGB mean vectors is calculated and normalised. The colour difference $Col_{diff}^{a,b}$ between two colour vectors C_a and C_b is defined as

$$Col_{diff}^{a,b} = \frac{\sqrt{(C_a^r - C_b^r)^2 + (C_a^g - C_b^g)^2 + (C_a^b - C_b^b)^2}}{\sqrt{(3 \cdot 256^2)}}.$$

A threshold c_{th} is then defined in order to determine if two vectors can be considered as similar or different. Thus, two colour vectors C_a and C_b are treated as similar if $Col_{diff}^{a,b} < c_{th}$ and are considered different if $Col_{diff}^{a,b} > c_{th}$. The value of c_{th} will later be optimised and discussed.

Addition The addition of a person is carried out whenever a camera-detected pedestrian is considered different than every other targets. The new target is represented as a particle filter

and given its attributes as in the first algorithm. This particle filter is then added to the list of tracked targets. The newly added particle filter also receives a new attribute: its mean RGB colour vector. This attribute will remain the same as the particle filter evolves.

Tracking If a camera detection is considered as similar to one or more already tracked pedestrians, this detection will be sent to the corresponding particle filters only and will not be influencing targets with different appearances.

Removal A person is removed from the list of targets by deleting it's corresponding particle filter whenever no similar camera detections are found within a time limit of δ_t.

7.2 Updated data fusion

In order to improve the quality of the tracking, a new data fusion model is used. This updated approach is ddescribed below.

7.2.1 Camera range perception correction

As previously explained in Section 6.3.1, the visual detection algorithm uses a scale change factor of $2^{\frac{1}{10}}$. The distance estimation between a larger bounding box and the next smallest one (Δ_D) is thus estimated as $\Delta_D \approx 0,07 \cdot D$. This causes the visual detection position estimations to only have a finite number of distances to chose from. Furthermore, these estimations grow more separated as the distance increases. This effect is shown in Figure 7.2.

It is assumed that the use of these camera position estimations for the particle filter updates were greatly and unfavourably influencing the tracking performances. This is thought to be due to the fact that the error of the camera position estimation in not normally distributed. It was noticed that at a constant distance from the camera, there are usually 5 consecutive sizes of bounding box that can be given as output from the camera detector. These sizes are comprised of the optimal bounding box size, two larger ones and two smaller ones. This means that when a bounding box is given as output from the detection algorithm, the target can usually be anywhere between the distance estimated by a bounding boxes two increments smaller and two increments larger. Using the formulas derived in Section 6.3.1, if the distance D is computed for the detected bounding box, the only thing known from the detection is that its actual distance is somewhere between $(\frac{1}{\delta_h^2} D) \approx 0.87 \cdot D$ and $\delta_h^2 D \approx 1.15 \cdot D$ meters.

Because of this poor range estimation from the camera, it was initially thought that the error model from the camera could simply be modified from a normal distribution to a uniform distribution, bounded by $0.87 \cdot D$ and $1.15 \cdot D$, along the radial axis. Unfortunately this lead to particles significantly diverging at each camera update.

It was then imagined that previous radar observations could be of use to get better estimations of the range, under the assumption that, in the time between the last radar update and the camera update, the target has not moved significantly. The idea is to use the camera detection position estimation of a target to define an area in the azimuth plane in which it is possible that the detected pedestrian is located, as shown in 7.3. The radar observations from the previous radar update that are located in this area will then be used to determine a probable distance.

Instead of receiving one detection from the camera and updating the tracking algorithm with its corresponding error model, the particle filter is updated using a set of points combining the distance estimation of the radar and the angle estimation of the camera. As shown in Figure

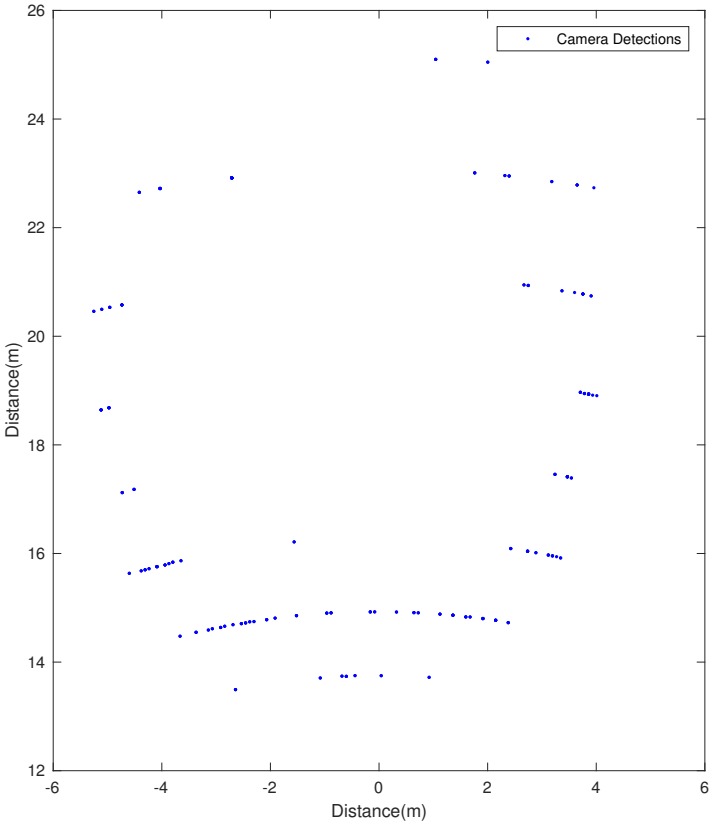


Figure 7.2: Visual detection position estimations on a circular path.

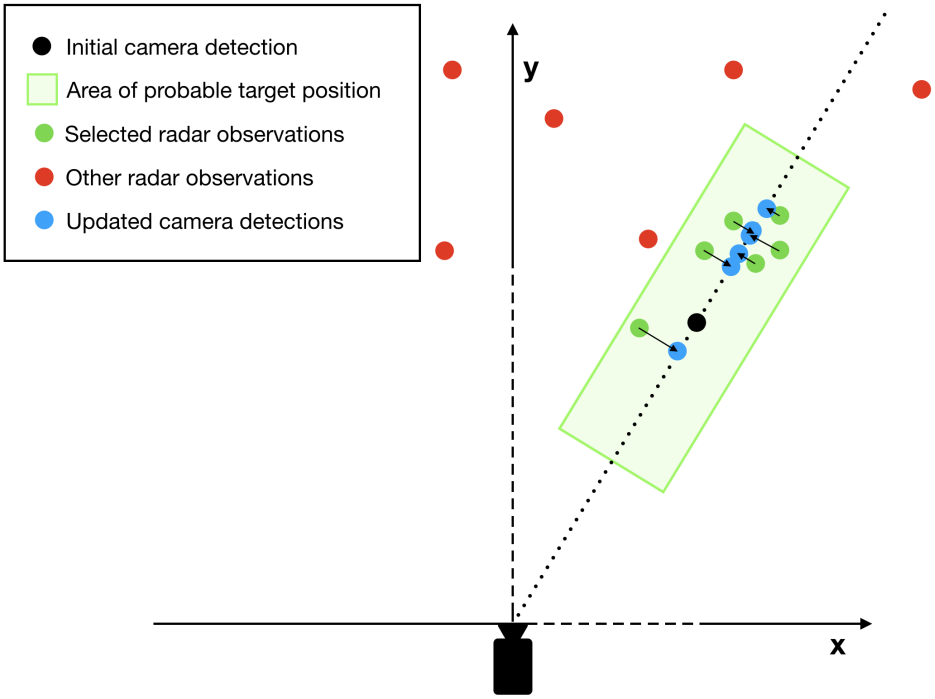


Figure 7.3: Representation of the camera range perception correction.

7.3, this set of points essentially consists of the selected radar observations which have been projected on the line passing through the camera and the camera detection. This way of fusing the data was chosen because, from the previously found values of δ, γ_1, and γ_2, it was shown that the camera's azimuth angle estimation is significantly more accurate than the radar's, and, conversely, the radar's distance estimation outperforms the camera's.

The selected area's dimensions were chosen as follows. Firstly because small angles are being dealt with, it was assumed that a rectangle in the Cartesian space would be a close enough estimation of a region delimited by a large range difference and a small angle variation in the polar space. For the length of this rectangle, it made sense to use the previously computed values $0.87 \cdot D$ and $1.15 \cdot D$ because it corresponds to the observed variations. For the width, inspiration was taken from the previously computed radar error model that showed, after optimisation, a standard deviation of up to $1.22m$ in all directions. It was thus decided that the width of the rectangle should span over 2.44 meters (1 standard deviation in each direction). This value was chosen because the radar's angle computation is not very accurate, thus having a too small width could mean not selecting the observations that concern the intended target.

The new way of defining the camera-based update to the particle filter means that a new corresponding error model has to be defined. Because the update is essentially using radar data, it was decided to use a similar model to the radar's for this new type of update. Because camera data is also available at this iteration, additional information about the colour difference $Col^{a,b}_{diff}$ between the original target and the detected one is known. Therefore, this information can be used in the error model. It was incorporated by dividing the radar observations' weights by $Col^{a,b}_{diff}$. The idea behind this is to minimise the influence of the observations if the colours do not match well. The new covariance matrix linked to this error model is thus given by

$$\boldsymbol{\sigma}_c^2 = \begin{bmatrix} \dfrac{\delta \cdot Col^{a,b}_{diff}}{I_r} & 0 \\ 0 & \dfrac{\delta \cdot Col^{a,b}_{diff}}{I_r} \end{bmatrix}. \tag{7.1}$$

Selecting the last radar position points in a particular area is deterministic as opposed to the particle filter, which is stochastic. This approach is unusual but brings the benefit of accurate angle computation from the camera and accurate range computation from the radar.

7.2.2 Data fusion architecture

The introduction of this preliminary data fusion operation before the particle filtering changes the whole data fusion scheme. As a reminder, the previous data fusion scheme was only based on the feature extraction based centralised architecture. But in Section 6.2, another choice of data fusion architecture was possible: the hierarchical architecture. The final designed scheme is a combination of both architectures. The feature extraction based centralised architecture is kept in order to feed the particle filter and the hierarchical architecture is used in order to fuse the camera and radar data as explained above. The new data fusion scheme is represented in Figure 7.4.

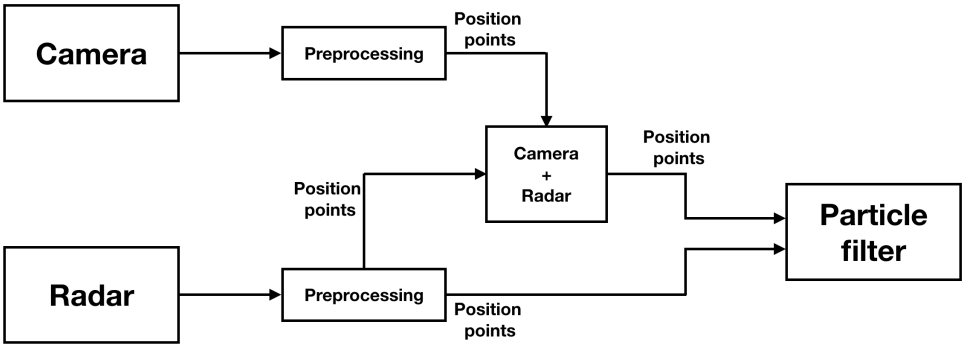


Figure 7.4: Data fusion scheme of the improved algorithm.

7.3 Selective resampling

While looking at the evolution of the particles iteration per iteration, as explained in Section 6.7, it was noticed that unpredictable false positives and false negatives coming from the radar cause the particle filter to converge towards wrong locations, which throws the particles off course. This happens most often when the tracked pedestrian remains undetected by the camera. Indeed, when the camera does not detect the pedestrian for a moment, the particle filter still updates with data coming from the radar. If the radar data is very noisy, the particle filter might be mislead towards incorrect observations. And when the pedestrian is detected again by the camera, the particles may be too far from the detection's position to converge back on the target.

In order to avoid this kind of error, it was decided to modify the resampling method when radar updates are performed. To do so, less importance is given to the resampling step by choosing to keep a certain proportion C_P of the particles computed at the prediction step as final particles. This is done by performing the usual resampling step with all the particles and then randomly swapping $C_P \cdot N_P$ particles with random non-resampled ones. This is defined as the "selective resampling". If the value of C_P is zero, then the particle filter updates as it did in the previous algorithm, but if $C_P = 1$ it is equivalent to not performing a particle filter update with data from the radar.

It can be argued that the original resampling method already selected particles with poor weights by design, but because its goal is to estimate the PDF computed by the innovation step, the amount of these particles that it selects cannot be chosen.

7.4 Parameter optimisation

As with the previous algorithm, there are variables that have to be configured as to optimise the tracking quality. These are listed below.

- α: linked to the propagation of the prediction step (Equation 6.6).
- δ: linked to both the radar's and the camera's error model (Equation 6.4 and Equation 7.1).
- C_P: the resampling coefficient.
- c_{th}: the colour difference threshold.
- δt: maximal time limit between two camera updates before removal of a particle filter.

It can be noted that γ_1, γ_2, and N_p, although still present in the algorithm, are not part of the list above. In the case of γ_1 and γ_2, this is because, as previously mentioned, they only influence the creation of a new filter, which has only a minor effect on the tracking and therefore the values computed for the previous algorithm were reused. As for N_p, the shape of the graphs were not expected to change significantly and therefore the same number of particles was used as previously.

The same tests as the ones described in Section 5.4 were performed to define these variables.

7.4.1 Optimisation of α and δ

As the use of α and δ are similar for this algorithm as for the previous one, their boundaries were set in a similar fashion. For α, the initial boundaries were set between $\alpha = 0$ and $\alpha = 4$, but, again, only small values of alpha gave good results and a more precise analysis was made with values only going up to $\alpha = 0.5$. For δ, values between $\delta = 0$ and $\delta = 100$ were again selected. The results from these test are shown in Figures 7.5 and 7.6.

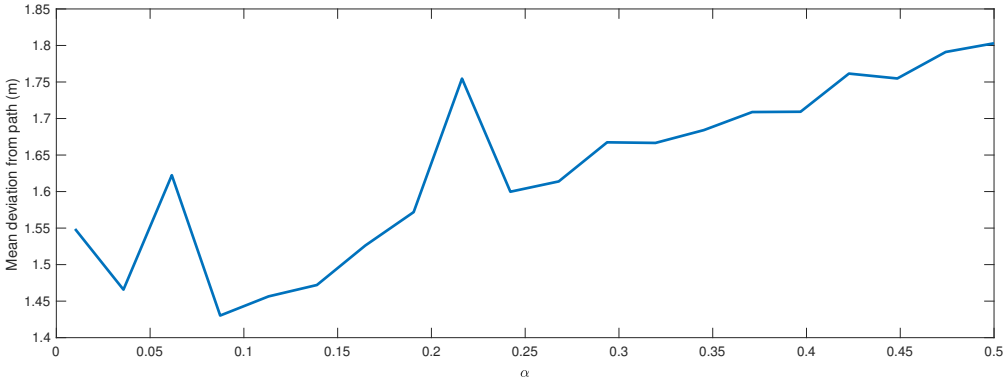


Figure 7.5: Variation of $\bar{\Delta}_{\boldsymbol{xc}}$ as a function of α.

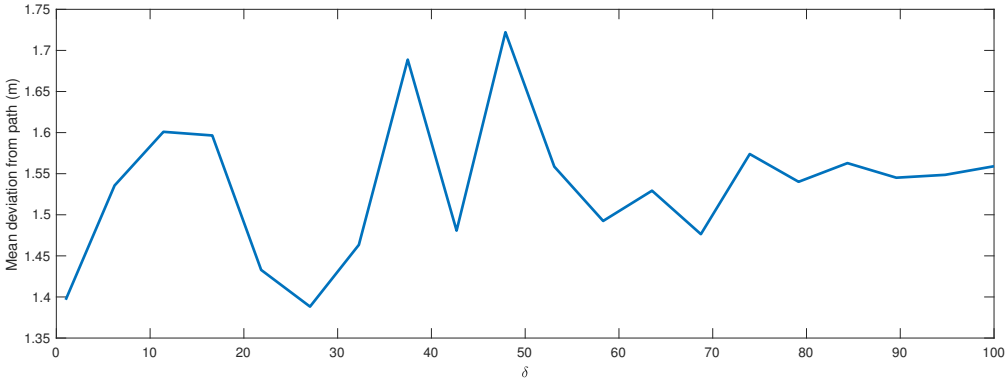


Figure 7.6: Evolution of $\bar{\Delta}_{\boldsymbol{xc}}$ as a function of δ.

In Figure 7.5, the minimum error happens to be close to $\alpha = 0.1$. This value means that, with a mean value $\Delta T \approx \frac{1}{10} = 0.1s$, using the formula expressed in Equation (6.6), it yields a standard deviation independent value of $0.1m$.

In Figure 7.6, the value of $\delta = 28$ has proven to give the best and most stable results. This value defines the covariance of the radar's error model and yields a standard deviation, symmetrical in both axis, ranging from $0.75m$ to $1.18m$ depending on the radar's signals intensities. As for the new camera model, a maximum standard deviation can be found at $0.25m$ for $c_{th} = 0.2$,

for instance.

These values have proven to work well across all tests and were thus adopted as default values for this algorithm.

7.4.2 Optimisation of C_P

This parameter refers to the amount of particles that will skip the resampling step during a radar update. This parameter can range from 0, meaning the resampling is done as normal, to 1, meaning the resampling step is completely skipped, and thus, no data information is held into account for this update. The results of the tests on various values of C_P are shown in Figure 7.7. The value of $\bar{\Delta}_{\boldsymbol{x}c}$ tends to decrease steadily as C_P rises until $C_P = 0.6$. At this point, the curve seems to level out and reaches an asymptote. It can be seen that the minimum value of $\bar{\Delta}_{\boldsymbol{x}c}$ is achieved when $C_P = 0.75$. For this reason, this value was adopted for the use of this algorithm.

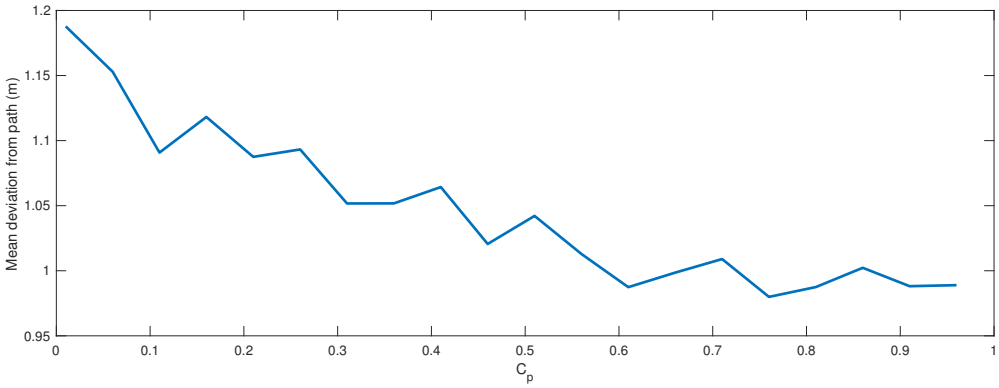


Figure 7.7: Evolution of $\bar{\Delta}_{\boldsymbol{x}c}$ as a function of C_P.

7.4.3 Optimisation of c_{th}

This parameter determines, as explained in Section 7.1, if (1) a camera detection should create a new tracked target, (2) which camera detection should update which particle filter(s), and (3) which tracked pedestrian should be deleted from the list of tracked targets.

If c_{th} is too small, the algorithm will see differences for small variations of colours. It will thus create useless new particle filters when a tracked pedestrian's RGB mean colour vector changes a little bit. When tracking pedestrians, camera detections will less likely update their corresponding particle filters and particles filters will more likely be removed because no corresponding camera detection will happen for a certain duration δt.

Conversely, if c_{th} is too great, the algorithm will not see much difference even between great variations of colour. Particle filters will thus less likely be created. When tracking, particle filters will likely be updated by many different camera detections, which will hinders the differentiation between particle filters. And the removal of particle filters will be less frequent because particle filters will often be updated by non-corresponding camera detections.

In order to define the optimal value of this parameter, the tests explained in Table 5.2 will have to be performed. These situations involve two targets of similar or different appearances crossing paths. By looking at how many filters were created and if the filters converge onto the right targets, an optimal value of c_{th} can be determined. No simple automatic and objective way of quantifying the results obtained by different values of c_{th} could be found. Therefore, the assessment was done subjectively by hand while looking at the positions of the various particles throughout the tracking scenarios. It was found that a value of $c_{th} = 0.15$ is the value that

gives the best compromise between separating the targets correctly and not creating too much superfluous particle filters.

7.4.4 Optimisation of δ_t

As for the optimisation of c_{th}, the calibration of δ_t also has to be done manually. In particular, the removal of the particle filters of the pedestrians has to be analysed. If δ_t is too small, particle filters will get too often deleted from the list of tracked pedestrians. On the other hand, if δ_t is too great, particle filters will remain on the image for δ_t seconds after disappearing from the field of view, and superfluous particle filters will be more likely to remain in the list of tracked pedestrians. Thus a middle ground has to be determined. After a thorough analysis, a value of $\delta_t = 2s$ has been found.

7.5 Results and observations

Results from the various tests explained in Section 5.4 will be presented below. The two main elements that will be analysed are the accuracy of the algorithm and its capacity to handle multiple targets.

7.5.1 Accuracy

(a) No challenge This test is the most basic one representing an ideal situation. It can be observed that this version of the algorithm follows the ground truth trajectory with great accuracy. The mean deviation from the ground truth is $\bar{\Delta}_{\boldsymbol{x}c} = 0.5067m$. Additionally it can be noticed that the particles, whose distributions are represented by ellipses, are mostly densely packed during the trajectory except briefly on the top left corner and during the bottom right quarter of the circle. These larger ellipses also correspond to zones where the centre of these ellipses do not exactly follow the ground truth.

(b) Occlusions Before and after the occlusion occurs, the trajectory is well followed by the algorithm. It can be observed that the particles get more dispersed as the occlusion blocks the pedestrian. The top portion of the trajectory indicates that the target was not precisely tracked, which was to be expected due to the occlusion. Nevertheless, the error during this portion is not as significant as with the previous algorithm. The mean deviation from the ground truth is $\bar{\Delta}_{\boldsymbol{x}c} = 0.8377m$.

(c) lighting variations Although the ellipses follow the estimated trajectory reasonably well, deviations from the ground truth can be observed on the right part of the circle. The ellipses are also larger in general for this test than for the others. The mean deviation in this test is $\bar{\Delta}_{\boldsymbol{x}c} = 0.7448m$.

(d) Radar noise The same general observations of the first test can be applied here with a $\bar{\Delta}_{\boldsymbol{x}c} = 0.6248m$ mean deviation. Larger deviations can be seen on the right side of the circle as well as on the bottom left. These deviations coincide with regions where ellipses grow larger.

(e) No radar The trajectory deviates a lot from the ground truth by tracing a squared shape path. The ellipses are significantly smaller than in other tests even if the tracking is off course. The resulting mean deviation is of $\bar{\Delta}_{\boldsymbol{x}c} = 2.2351$.

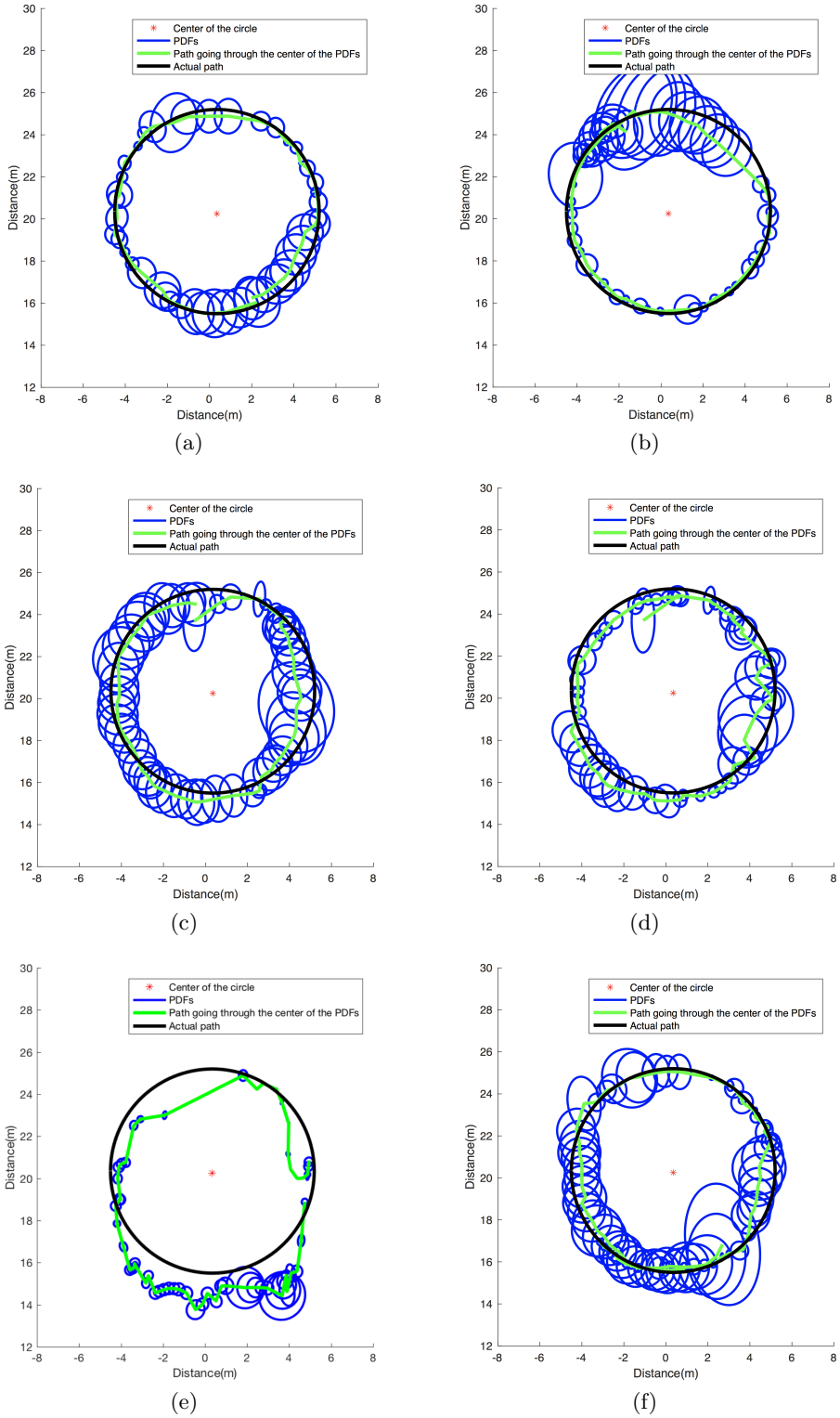


Figure 7.8: Figures of the different tests. (a) No challenge(ideal situation): 5m radius circular path 20m away from the device. (b) Occlusions: 5m radius circular path around a 40cm radius tree 20m away from the device. (c) Lighting variations: 5m radius circular path 20m away from the device partially in the sun. (d) Radar noise: 5m radius circular path next to moving bushes 20m away from the device. (e) No radar data: 5m radius circular path 20m away from the device. (f) No camera data: 5m radius circular path 20m away from the device.

(f) No camera Camera data was used only for the first detection and left unused for the rest of the trajectory. The same general observations as with lighting variations can be made. In particular, the trajectory, although accurate in the y-axis, lacks stability in the position estimation in the x-axis, as can be seen on the left and right sides of the circle. The resulting mean deviation is of $\bar{\Delta}_{\boldsymbol{x}c} = 0.6754$.

Test	$\bar{\Delta}_{\boldsymbol{x}c}$ **for the first algorithm**	$\bar{\Delta}_{\boldsymbol{x}c}$ **for the second algorithm**
No challenge	0.8254	0.5067
Occlusions	1.9623	0.8377
Lighting variations	0.6884	0.7448
Radar noise	0.6789	0.6248
No radar data	1.1256	2.2351
No camera data	0.7328	0.6754

Table 7.1: Tests results of the first and second algorithms.

7.5.2 Handling multiple targets

The second part of the testing analyses the way multiple targets are dealt with and what the limits of this algorithm are. When performing the various tests described in Section 5.4, some scenarios caused different behaviours of the particle filters. These behaviours will be discussed next. The representations of the following behaviours are depicted in Figures 7.10, 7.11, and 7.12.

Firstly it can be observed that, in Figure 7.10, when pedestrians wear clothes of opposite colours, the algorithm tracks them with great accuracy, even when they are less then a meter from each other and walking in the same direction. The particles are tightly grouped around the right targets and mostly do not overlap. The only element that is noticeable in such a scenario is the creation of surplus filters. These excess filters follow targets that were already followed by another filter.

Secondly, in Figures 7.11 and 7.12, when targets wear similar colours, excess filters can also be noticed, but they are not the only issue. The tracking of these similar targets is very unstable. When targets get closer to one another, particles from all filters can be seen around both targets locations which indicates the fact that the algorithm cannot differentiate properly between them. Moreover, when the targets get close to one another, the particles representing them tend to agglomerate in the space between them instead of forming one cluster on each target. It can also be seen that filters jump from a target to another between iterations even if targets are more than five meters apart (both in the x and y axis). These large variations in the positions of the particles can occur both with camera and radar updates.

7.6 Discussion

By analysing results obtained in the previous section and comparing them to the results obtained by the first iteration of the algorithm, certain elements stand out and will be explored. Two main aspects can be discussed in this section, namely the accuracy of the tracking and the way multiple targets are handled. These two elements will be examined in this section by analysing the values of the parameters computed in Section 7.4, as well as the results of the various tests made in the previous section. Observations from this algorithm will be compared to the previous version in order to assess whether the modifications that were made were justified.

7.6.1 Accuracy

Three main observations concerning the accuracy of the tracking will be discussed below. Firstly, the tracking with occlusion is much better handled. Secondly, the ellipses' sizes vary much more than for the first algorithm. Thirdly, although the radar seemed to give good results for the tracking, the selective resampling has a coefficient of $C_P = 0.75$, meaning that the radar updates are barely used on their own.

7.6.1.1 Occlusion handling

The "occlusion" tests are depicted in Figures 6.9b and 7.8b for the first and second algorithm respectively. As can be seen on these figures but also in Table 7.1, the tracking is much better handled by the second algorithm in the case of an occlusion. With the first algorithm, the particle filter is attracted towards the radar noise while with the second algorithm, the particle filter remains in place until a new camera detection occurs. That is believed to be caused by two factors. Firstly, the newly optimised value of α, which is smaller, and secondly, the introduction of the selective resampling.

Concerning the value of α, in the first algorithm, a value of $\alpha = 0.25$ was found. This value leads to a standard deviation of $0.16m$ within a time difference of $\Delta T = 0.1s$, while the estimated walking distance by a pedestrian during $\Delta T = 0.1s$ is of $0.14m$. In the second version of the algorithm, a value of $\alpha = 0.1$ has been optimised. This value generates a standard deviation of $0.1m$ which is considerably smaller than the previous one. This evolution of α means that, after the modifications explained above, the prediction updates are less spread out, which shows that the updates of the particle filter can be more trusted. It is believed that this is mostly due to the new camera data update. An advantage of this small value of α is also that, when the camera does not detect the target anymore, the particles spread less broadly and, thus, follow the radar noise less.

Concerning the selective resampling, the particularly high value of $C_P = 0.75$ should be noted. This means that when the radar performs the tracking on its own, only 25% of the particles get resampled whereas 75% of them stays spread out. This is the reason why, in Figure 7.8b for instance, as soon as the camera looses the pedestrian because of the occlusion, the particle clouds start to inflate until the camera performs another detection. Thus, the advantage of that selective resampling is that, instead of following the noise like with the first algorithm, the particle filter simply remains near the last camera detection and grows at each radar update in order to anticipate the position of the next camera update.

7.6.1.2 Ellipses' size variation

In the test presented in Figure 7.8, it can be noticed that the ellipses representing the distribution of the cloud of the particles vary much more than for the first algorithm. These clouds often grow when the tracking leaves the supposed-to-be-followed path while they usually shrink when they get back on the path.

Smaller particle clouds can be explained by the fact that a smaller value of α is used (0.1 against 0.25), along with the more precise camera position estimation. It means that during the prediction step, particles spread out less further and often arrive close to the detections. On the other hand, bigger particle clouds, can be caused by two different factors. Firstly, as explained above, when the camera does not detect the pedestrian anymore, the selective resampling makes the particle clouds grow incrementally. This happens in the "occlusion" (b) and the "lighting variations" (c) tests when the camera looses the pedestrian. Secondly, radar noise can also enlarge

the particle clouds because they bring ambiguities. This effect can be noticed on the "radar noise" (d) test for instance.

7.6.1.3 The resampling coefficient C_P

As stated above, the value $C_P = 0.75$ is particularly high. This means that radar data on its own has a small impact on the particle filter updates. This can seem counter-intuitive because from the results displayed in Figures 6.9f and 7.8f and in Table 7.1, it can be seen that the radar on its own can perform a reasonably accurate tracking, which, more importantly, is significantly better than what the camera can do on its own. This can be be explained by the fact that the updated range computation explained in Section 7.2.1 (combining the radar range and the camera angle computations) reaches a much greater precision than any of the two sensors on their own. This data fusion is so accurate that the radar data, that was previously proven to be accurate enough to track a single target with good accuracy on its own, is barely used alone anymore.

7.6.2 Handling multiple targets

In Section 7.5.2 and on the basis of Figures 7.10, 7.11, and 7.12, two observations have been made.

First, is has been noticed that apart from the fact that superfluous particle filters were created, the tracking of dissimilar pedestrians is much better handled than with similar appearances. This difference in performance is a good indication that a robust target discrimination with the appearance could lead to great performances when tracking multiple pedestrians.

Secondly, if the appearance of pedestrians is similar, particles filters are more likely to make errors due to the radar data. This observation, once more, reinforces the idea that a stronger pedestrian appearance discrimination needs to be implemented.

7.7 Comparison with the state of the art

After assessing the performance of this algorithm, its quality can be compared to other currently used tracking algorithms. As mentioned previously, [61] and [55] also developed multiple object tracking algorithms using a camera and a radar, and it thus seems relevant to compare the results we have obtained with theirs. Unfortunately, their code is not available and not much details were given on the accuracy of these algorithms. For these reasons, it was decided to look at other, more common, tracking methods only using cameras with monocular data.

As previously mentioned, the accuracy of monocular vision based algorithms is determined using the $MOTP_{3D}$. The best algorithms give average values of $MOTP_{3D} = 0.533m$, which means the standard deviation between the estimated position and the ground truth throughout the databases of [67] is $0.467m$. Which at first sight seems to be better than this algorithm, however, it must be noted that these data sets are completely different to this thesis'. Indeed, by looking through the "MOT Challenge" database, the 3D tracking situations are either (1) a camera at eye-level, situated in a crowd, tracking pedestrians in close proximity, or (2) a camera placed high above a certain location, tracking people below. None of these two situations correspond to the intended use of the algorithm developed in this thesis (having a camera placed at human height tracking pedestrians at long ranges). Thus these 3D tracking tests aim to track pedestrians at a range proportional to the camera's height. That could be justified by the fact that most 3D tracking algorithms with a single camera primarily use the detected position of the pedestrians' feet in order to estimate the range of these pedestrians. Thus the main element that plays a role in the accuracy of the detection is the angle between the ground and the line passing

through the targets' feet and the camera. In Figure 7.9, this angle corresponds to ω. The bigger this angle is, the more accurate the detection can be. This angle shrinks when the distance between camera and pedestrian grows, and increases with the height at which the camera is placed. It is thus not surprising that the tests of [67] are done either with a low camera for close proximity targets or with a high camera for further ones.

The first comparing option is to evaluate this thesis' algorithm with benchmarks' data sets. However, none of these data sets contain radar data, which makes the evaluation impossible.

The second comparing option is to evaluate the benchmarks' top performing algorithms with this thesis' data set. Although the comparison of the results would be biased in such situations, because this thesis' algorithm's precision intrinsically does not depend on the height of the camera, such an assessment would be interesting. However, once more, the comparison is impossible because none of these codes could be obtained.

Although no direct comparison with existing algorithms could be established, an estimation can be made as to how precise a monocular tracking algorithm should be in order to be equivalent to the one developed in this thesis. This can be done by modelling the testing scenarios with Figure 7.9. The device is placed at a height h_{cam} above the ground and the target is at an average distance D from it with an error margin of $\pm\Delta_D$. The intent is to find the amount of pixels that correspond to a certain precision $\pm\Delta_D$ in the azimuthal plane. When looking at the case of $+\Delta D$, via Thales' theorem, $\frac{\beta}{D} = \frac{h}{f}$, and in the following figure it can be seen that $\frac{\beta}{\Delta_D} = \frac{h_{cam}}{D+\Delta_d}$. Putting these together gives

$$h_+ = \frac{f \cdot \beta}{D} = \frac{f}{D} \cdot \frac{\Delta_D \cdot h_{cam}}{D + \Delta_D}. \tag{7.2}$$

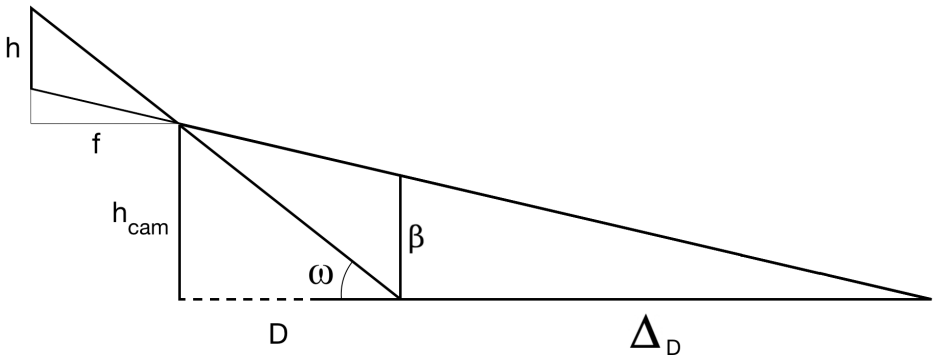


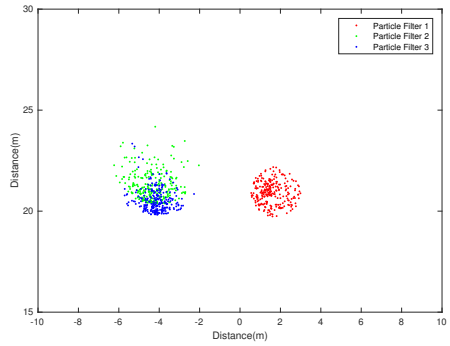
Figure 7.9: Testing scenario following the same notations as the pinhole model.

As for the case of $-\Delta D$ the same computation as above can be made by simply replacing D by $D - \Delta D$ and $D + \Delta D$ by D. Thus obtaining:

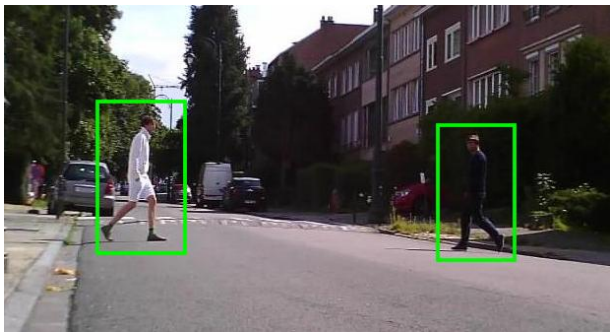
$$h_- = \frac{f \cdot \beta}{D} = \frac{f}{D} \cdot \frac{\Delta_D \cdot h_{cam}}{D - \Delta_D}. \tag{7.3}$$

With a camera placed at $h_{cam} = 0.8m$ from the ground, the computation of the range of a target, at a distance of $D = 20m$ with the requirement of an average accuracy of $\Delta_D = 0.6m$,

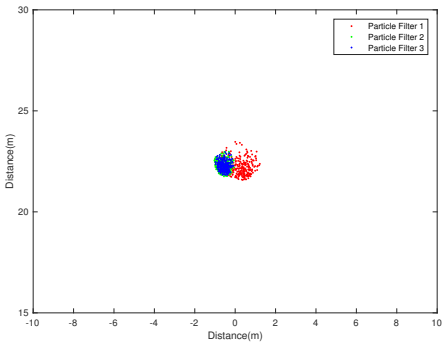
requires a precision of $h_+ \approx 1.45$ pixels and $h_- \approx 1.55$ pixels. It is thus believed that, with the setup of this thesis, a tracking algorithm only using the camera would have to be able to track the pedestrian's feet with an error of less than two pixels in order to match the performance of the developed algorithm.



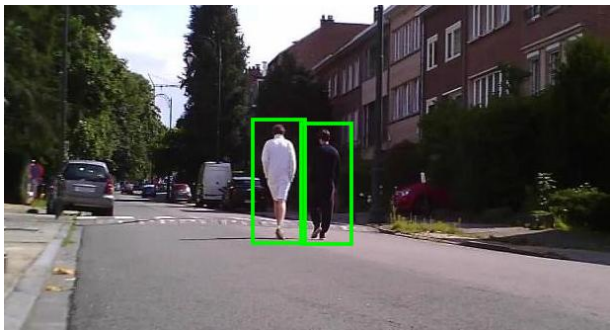
(a)



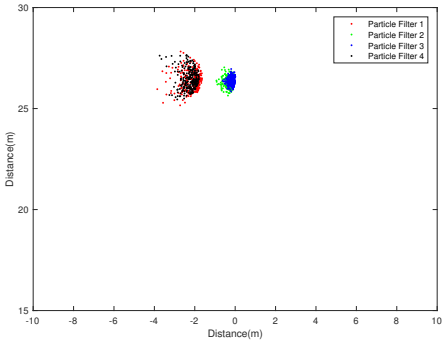
(b)



(c)



(d)



(e)

(f)

Figure 7.10: Two pedestrians with different appearances are walking towards each other, walking side by side and then crossing path to different directions. It corresponds to the test of the fourth row and first column of Table 5.2.

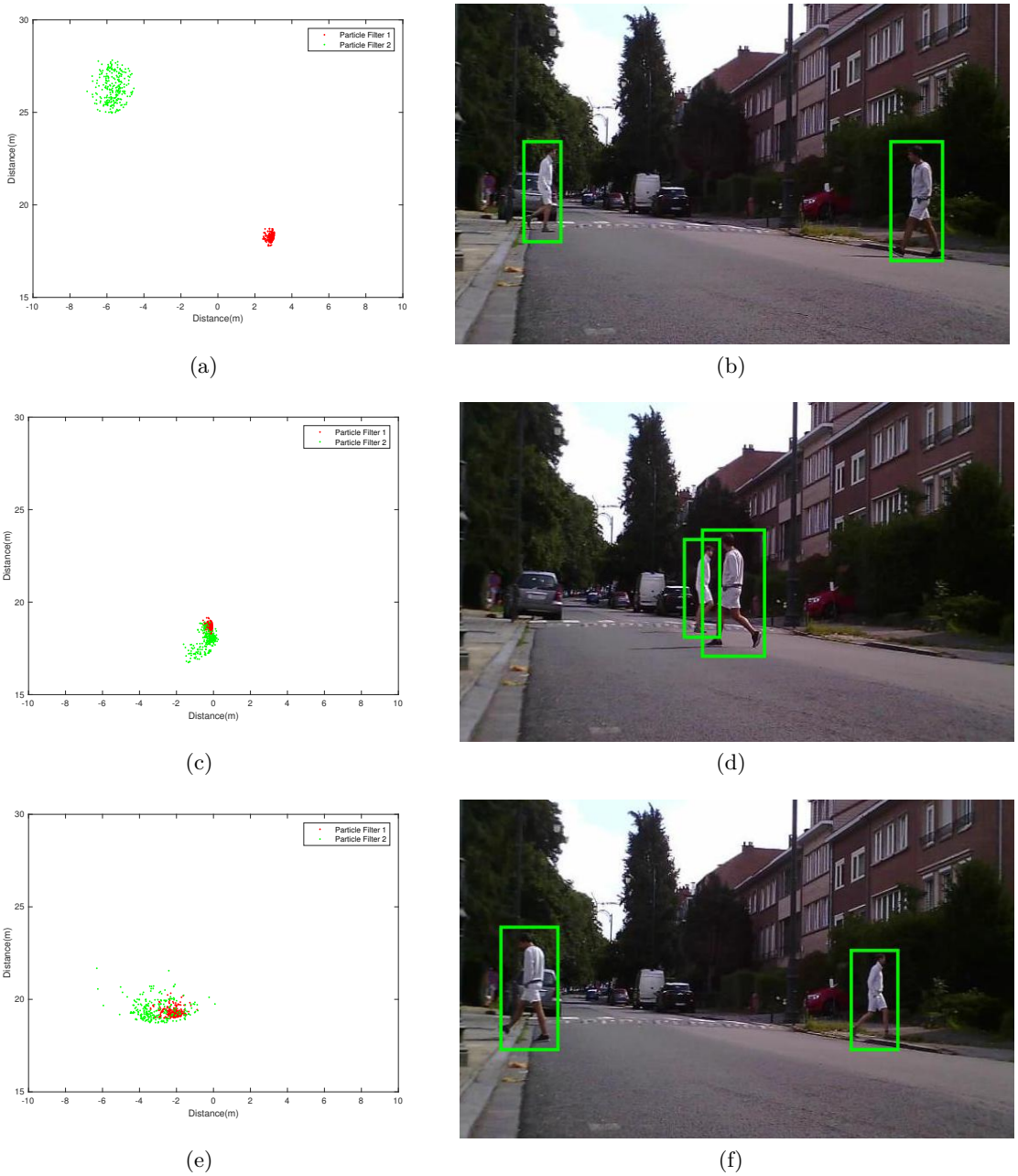


Figure 7.11: Two pedestrians with similar appearances are walking in two parallel straight lines six meters apart. It corresponds to the test of the first row and second column of Table 5.2.

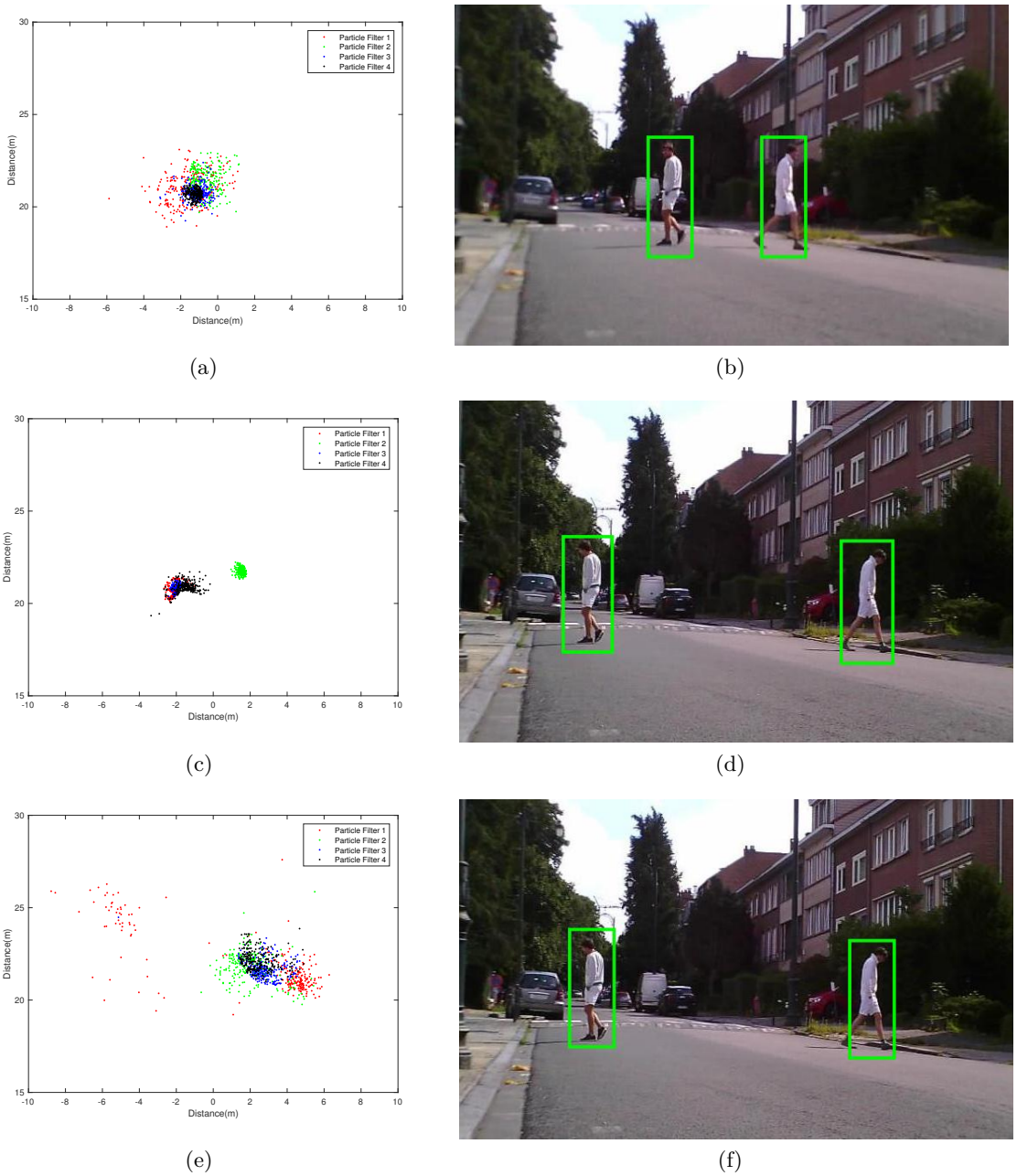


Figure 7.12: Two pedestrians with similar appearances are walking in two parallel straight lines six meters apart. It corresponds to the test of the second row and second column of Table 5.2.

Chapter 8

Limitations, future works and applications

Now that the whole implementation process, the final algorithm, and the test results have been thoroughly explained and analysed, limitations of the current solution and future works that can be conducted in order to improve it will be presented. These observations will be useful for future researchers working on the continuation of this project. After analysing the performances, possible applications of the device will be presented.

8.1 Limitations and future works

Ideas concerning the implementation can be categorised according to one of the next objectives: improving the visual detector performances and efficiency, improving the distance estimation by the camera, and better handling multiple objects. Ideas also concerning the equipment will be cited.

8.1.1 Improving the visual detector performances and efficiency

These suggestions aim to improve the the performances as well as the efficiency of the visual pedestrian detector.

8.1.1.1 Lowering the computation power requirements

The current algorithm is able to run a real-time tracking but its speed and accuracy highly depend on the available computing power. Indeed, as said earlier, the bottleneck of the program is the visual detection. Conversely, the computational time of the tracking operation is negligible and the radar signal processing is performed externally with a communication time lag that is also negligible.

During tests, it was noticed that the algorithm gave extremely good results on a 2.7GHz quad-core Intel Core i7 processor machine with 16GB of ram and solid-state drive. However, testing on an older 2,4 GHz bi-core Intel Core i5 with 8GB of ram and a solid-state drive gave no conclusive results at all because of a too low frequency of visual detections.

Thus a solution to improve both the speed and the accuracy of the tracking is necessary. One could be to replace or modify the visual detection algorithm to a faster one, naturally with the same or better performances. This is the reason the idea proposed in the next subsection was imagined.

8.1.1.2 Candidate generation for pedestrian detection

The high computational time of the pedestrian detector is partially due to the exhaustive scan performed on the image before the pedestrian classification. As explained in Section 3.1.1.1, candidates are obtained by selecting every possible bounding box's size at every possible position in each frame. With large image formats, this number can get extremely large. It is thus thought that the number of these candidates could be considerably decreased with a new candidate generation algorithm using previously acquired data from the radar. By reversing the pinhole model described in Section 3.1.2.1, each radar observation could generate a set of possible candidate bounding boxes for pedestrian classification. It is believed that, depending on the environment where the device is placed (depending on the radar noise), the use of this method could reduce the number of generated candidates by up to two orders of magnitude.

8.1.2 Improving the distance estimation by the camera

This section aims to improve the distance estimation by the camera by proposing new features to the already existing pinhole model.

8.1.2.1 Pinhole model

The pinhole model explained in Section 3.1.2.1 assumes that each bounding box constructed around a pedestrian corresponds to a $2m$ high box around it. This is quite a strong hypothesis considering the fact that, as stated in Section 6.3.1, the bounding box around a pedestrian increasingly overestimates its real size the further it is. It is thus believed that a solution giving a more correct distance estimation would be to test, for each bounding box's size, the average height overvalue and implement it in the algorithm as a table of value.

Another problem of the current distance estimation via the pinhole model is the fact that is is assumed that pedestrians have a height of $1.70m$ resulting in a $2m$ high bounding box. Particularly small or tall pedestrians are thus less likely to be estimated at their actual range. A solution to that problem could be, when a pedestrian is first detected, instead of only taking into account radar values around the position estimation of the camera, considering every possible radar observations along the camera's radial axis in order to be certain to capture the pedestrian with the radar observations, whatever its size is. Then, whenever the certainty of having detected the same pedestrian with the radar and the camera is sufficiently high, the pedestrian's actual height can be saved for further distance estimations. This solution, however, does not seem suited for crowded environments.

8.1.3 Better handling of multiple objects

This section aims to improve the way multiple targets are handled by the tracking algorithm. The first two subsections present suggestions to improve the already implemented multiple object management of the tracking algorithm while the third subsection proposes a different way of handling multiple targets by making use of an independent camera-based tracking algorithm.

8.1.3.1 Target discrimination

The current algorithm, when updating with the camera detections, only uses the RGB mean colour as a feature in order to discriminate between detections. Also, the RGB mean colours of tracked pedestrians are only saved at the first detection and are not updated during the tracking operation. Due to the issues that arose during the multiple target tests (discussed in Section 7.6), it is believed that a more robust approach concerning the target discrimination would benefit the algorithm greatly, especially in challenging situations for the camera. Two major upgrades

have been imagined in order to improve the robustness of the discrimination: the use of more robust features as visual cues to compare for discrimination and constantly updating the target's characteristics with camera detections.

More robust features, such as the ones presented in Section 3.1.1.2, could be used. The best suited features for this approach would be the less variant ones for the same pedestrians as well as the more discriminative ones between different pedestrians.

Updating these features with upcoming camera detections should also be done in order to adapt the model as the pedestrian's appearance evolves.

8.1.3.2 New way of detecting new targets

In the final algorithm, a tracked pedestrian is added whenever a camera detection has a different appearance from all the other tracked targets. This approach is risky because, if a new pedestrian enters the tracking area and a target with a similar appearance is already being tracked, the new target will never get tracked. In order to solve this problem, the distance between the detections should be held into account in some way so the difference between two targets can be clearer. It is believed that this distance could be quantified by the influence of a camera detection on all the other particle filters tracking targets of similar appearance. If the influence is considered as small enough, a new particle filter can be created. Thus, if a new pedestrian arrives in the field of view, far enough from any other similar targets, its camera detection should barely influence the other particle filters with a similar appearance, and therefore, a new particle filter will be created.

8.1.3.3 Using a state-of-the-art pedestrian tracker

As can be seen in the sections above, most of the improvements to the algorithm are camera-related. The main issue of the current algorithm is differentiating between targets using the image data from the camera. This problem is in fact already tackled by a multitude of monocular 2D and 3D pedestrian tracking algorithms such as the ones tested in [67] and [77]. These algorithms not only accurately detect the pedestrians, but also keep track of which detections belong to which targets. This type of information is exactly what is needed in order for this algorithm to perform ideally. Two possible methods can be used in order to integrate these existing algorithms to this thesis'.

Firstly, 2D algorithms can be used to compute which bounding box corresponds to which target. This information is usually computed under the form of probabilities, which can directly be used as a replacement for $Col_{diff}^{a,b}$ in the current implementation of this algorithm. All other computations can stay identical to the current version. Only the parameters might have to be re-calibrated, but apart from that, it is believed that adding a 2D tracking algorithm should not change the core programming behind the current algorithm.

The other possible solution would be to integrate a 3D pedestrian tracking algorithm to this thesis' program. These algorithms would, similarly to the 2D tracking methods, determine the correct bounding boxes for each target, but in addition to that, provide an initial estimation for the targets' positions, effectively replacing the pinhole model. This would require more effort to integrate due to the fact that various side functions would have to be re-written. A new error model would also have to be developed for the camera data and it is presumed that, due to the nature of 3D pedestrian tracking algorithms, this error model would largely depend on the positioning of the camera.

8.1.4 Equipment

Because the aim of this project is to develop a prototype, the sensors that are used were chosen in large part for their ease of use rather than their overall performance and their responses to

the requirements. It would thus be interesting to select more adequate hardware for future developments. Possible improvements for the choice of equipment are thus described in this section.

8.1.4.1 Choice of camera

Our current camera is a USB camera module. This choice has been made for the ease of use of this type of camera. But other cheaper and more advanced cameras are available on the market, such as, for instance, the IP cameras that are widely used for surveillance purposes. These cameras are often equipped with infrared LEDs that offer the possibility of night vision. It is thus believed that using, for example, a cheaper IP camera would add the possibility of tracking pedestrian in a dark environment. A possible limitation of the night vision, however, would be the range of vision (depending on the power of the IR LEDs).

As the current camera's view angle is of 60° and the radar's is of 30°, it is also recommended to get a camera with a closer field of view to that of the radar, or inversely, in order not to waste any data from one of the two sensors.

Another performance improvement would be to integrate a camera with a controllable optical zoom. This new feature could allow the camera resolution to remain quite low while detecting pedestrians at long ranges, thus allowing a low computational time for the pedestrian detection. The controllable zoom, however, will have to be relatively accurate in order to adjust the focal length with great precision.

A better choice of camera would lead the pedestrian detection algorithm to be able to detect people much further away from the device than with the current setup. By selecting the proper hardware, it is believed that targets could be detected at distances upwards of 50m, and thus, hopefully fulfil the initial goal that was set: to accurately track pedestrians at distances between five and fifty meters away from the device.

8.1.4.2 Use of a cheaper radar

As stated earlier in Section 3.2, for the purpose of focusing on the tracking and not on the signal processing of the radar, it was decided to buy a radar provided with an in-board signal processing unit, which was expensive. A future improvement and a substantial cost reduction would be to perform the signal processing in a computer or controller and thus buying a more basic radar only outputting the signals.

8.1.4.3 Parallel computation requirements

For the moment, the current algorithm is coded in the MATLAB programming language. If the real-time version is run, three different threads working simultaneously are needed (in the case of running it in playback, only two threads are needed). However, MATLAB only allows one thread per core, thus a three threads tracking would not be possible on a dual-core processor, even if it has more than two virtual cores. On the other hand, other lower level programming languages such as C or C++ can run on virtual cores. This drawback can thus be avoided by translating the code in a lower level programming language and by running the real-time algorithm in a computing unit having at least three virtual cores.

8.1.4.4 Micro controller integration

A next step in the project is to integrate the data processing and fusion inside a micro controller. However, micro controllers are often programmed in C language (C++ for more sophisticated

ones). Thus, in order to integrate this thesis' algorithm on a micro controller, it would, once more, be recommended to translate the code in C or C++ programming language.

The code of this project is divided into multiple parts: running the camera and its detector, running the radar and its communication protocol (its signal processing in the case of a lower level radar), and the tracking. It is believed that translating the tracking part would be quite straightforward. The signal processing of the radar, although not being implemented yet, should be feasible as well. Concerning the pedestrian detector, it would be unadvised to translate the current MATLAB code as it is extremely long and uses many native functions that would have to be created manually. Thus, another solution would be to find another pedestrian detector coded in C or C++. For instance, [5], which is written in C++, would be a good candidate as it can reach a detection frequency of $50fps$ on high quality images on a typical computer and with a slightly lower miss rate than the "Fastest Pedestrian Detector in the West". However, this code is only available for research purposes, which does not correspond to the commercialisation requirements of the project.

8.1.4.5 Extension to a moving platform

A major drawback of the device that is currently used is the small azimuth angle range of the radar. Because of this, targets can only be tracked in a relatively narrow field of view. In order to solve this problem, it was thought that the device could be placed on a rotating platform that would let the device scan a larger area. In the current state of the implementation of the algorithm, it is believed that it can be adapted to correctly function on a moving platform as long as the movements of this platform are known. The only difficulty will reside in dealing with uneven update rates for the targets because they will only be inside the field of view for a few frames in a row before not being detected until the device turns towards them again.

8.2 Applications

The algorithm proposed in this work offers a way of combining a radar and a camera to track pedestrians from a low vantage point. Although the implementation described does not yet deal satisfyingly with multiple similar targets, it is strongly believed that, following the suggested improvements above, the algorithm will be able to have comparable performances to the state of the art in multiple target tracking. It is also thought that the resulting algorithm, after being further developed, will be able to rival current camera-only-based tracking in low vantage point situations. This means that the finished device and program could replace the monocular vision systems for applications requiring medium to long-range tracking with this kind of device placed at low height.

The first domain that can be interesting for such a product would be autonomous vehicles. Indeed, the height is always restrictive in these kind of applications, which force the sensors to be placed inside the dimensions of a vehicle. Some smart vehicles already integrate a radar and multiple cameras but usually use them for separate tasks, having the radar focus on distant vehicles, and the cameras analysing the environment as well as tracking pedestrians. It is believed that integrating the algorithm discussed in this text in such vehicles could largely benefit their safety features.

Another field that could benefit from this type of tracking would be the security in large industrial zones. Being able to detect and follow trespassers in restricted areas can be very useful for certain firms wishing to control access on their property. Using a rotating device with a small viewing angle camera could be used to track people accurately over long distances. This could improve the protection of the area while requiring less usual security cameras.

Chapter 9

Conclusion

As the demand for accurate real-time pedestrian tracking increases, a large amount of researches are constantly being conducted, all aspiring to develop a method that could outperform all others. While most works in this domain base themselves on monocular vision, some research teams choose to experiment with other sensors, or combinations thereof. The underlying idea behind these new developments is that, in certain scenarios, a single camera might not always be the most efficient sensor that can be selected. As a matter of fact, due to the intrinsic nature of these sensors, monocular vision-based pedestrian trackers generally struggle when dealing with large distance variations. Radars on the other hand, while having other drawbacks, are a lot more reliable in such scenarios, hence the idea of combining these two aforementioned sensors in the hope of obtaining better results than either would be able to reach on their own.

The objective of this master thesis is to identify and exploit the strengths of both types of sensors in order to perform accurate pedestrian tracking in a scenario known to be challenging for both camera, and radar-based methods. For this purpose, a prototype device combining a sensor of each type was thus conceived and a pedestrian tracking algorithm capable of tracking multiple targets in the azimuth plane was implemented. Situations where the device is placed at less than a meter above the ground, tracking one or more pedestrians walking up to more than 25 meters away from the sensors, and in different patterns were exclusively tested in an attempt to prove the importance of this multimodal sensor combination.

The steps that were taken to implement a multiple target tracking algorithm combining a radar and a camera are explained throughout this master thesis and the perks and drawbacks of this fusion are explored. After establishing the context of the project, this work is divided in three main parts.

The first portion of this study focuses on the data gathered by the sensors and how it is transformed in order to be used by the tracking algorithm in itself. Firstly, the initial choices of hardware and software platform are detailed and justified. The signal processing that is performed on the radar data, and the pedestrian detection algorithm applied on the frames captured by the camera are then explained. Various methods used to fuse these types of processed data as well as multiple object tracking approaches are then mentioned.

After explaining and justifying essential choices that had to be made before the programming and elaborating a series of tests that would be used to measure the performances of the tracking, the implementation in itself is explained. A first version, only capable of tracking a single target, is initially introduced following methods described in the state of the art. This is established as being a first basic iteration on which more complex elements can be built. By analysing the results obtained by the first version, a second algorithm is implemented based on suggested improve-

ments as to increase the tracking accuracy and add the functionality to track multiple pedestrians.

In the last part of this work, after multiple tests are performed on the final algorithm, its accuracy and precision are assessed and conclusions are drawn from the observations. Limitations of the method that was developed are then expressed, combined with suggested solutions for future works. This thesis ends on proposed situations in which using a camera and radar combination would be optimal for multiple target tracking.

The final developed algorithm consists of a method of multiple object tracking using multimodal sensors, namely a camera and a radar. The signals from the radar are processed using a method developed by RFbeam [90] and outputs a series of observation points. The frames from the camera are analysed by a pedestrian detector developed by Dollár, Belongie and Perona. These detections are then converted into usable position estimations by the pinhole model. The data fusion is then performed based on architectures described in [70] and [32], which, when possible, matches radar observations with camera detections using the targets' appearances. This processed information is then handled by a particle filtering method inspired by [55] that iteratively outputs probability density functions of targets' locations.

It is found in this work that combining a camera and a radar with a particle filter is an effective method for multiple object tracking. Indeed, the test results show that when targets can be discriminated from one another, the particle filter approximates the targets' positions with an accuracy close to half a meter. This algorithm resists well to obstructions and converges back on the targets' positions after they are detected again. Unfortunately, in its current version, when targets are too visually similar, the particle filter does not handle them properly and loses its accuracy. This leads to the conclusion that the inherent accuracy of the tracking comes, in most part, from very good radar observations, but the crucial task of matching these observations to the appropriate targets is primarily done by the camera.

Although the algorithm's inability to deal with similar targets seems like an important shortcoming, it is firmly believed that it can be solved by updating the vision-based target discrimination using state-of-the-art methods. Multiple other possible suggestions were listed such as (1) improving the visual detector performances and efficiency, (2) improving the distance estimation by the camera, (3) using cheaper and better performance sensors and computing units and (4) adapting the device to a moving platform. Additionally, these improvements are believed to lead to an accurate tracking for targets at distances of more than 50 meters, as was hoped for in the initial goals of this work.

This work will hopefully serve as a baseline for researchers trying to improve current multiple target tracking algorithms by combining a camera and a radar. It is believed that future developments based on this study could obtain results comparable to the current state of the art in multiple object tracking.

In conclusion, this master thesis proposes a compelling way of combining a radar and a camera for real-time multiple object tracking with a particle filter. Results have shown that the algorithm that was developed offers good accuracy with clearly distinct targets and promises to get even better results once suggested improvements are implemented.

Bibliography

[1] M Sanjeev Arulampalam et al. 'A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking'. In: *IEEE Transactions on signal processing* 50.2 (2002), pp. 174–188.

[2] Jeffrey A Barnett. 'Computational Methods for a Mathematical Theory of Evidence.' In: *IJCAI*. Vol. 81. 1981, pp. 868–875.

[3] Otman Basir, Fakhri Karray and Hongwei Zhu. 'Connectionist-based Dempster-Shafer evidential reasoning for data fusion'. In: *IEEE Transactions on Neural Networks* 16.6 (2005), pp. 1513–1530.

[4] Alessio Benavoli et al. 'An approach to threat assessment based on evidential networks'. In: *Information Fusion, 2007 10th International Conference on*. IEEE. 2007, pp. 1–8.

[5] Rodrigo Benenson et al. 'Pedestrian detection at 100 frames per second'. In: *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE. 2012, pp. 2903–2910.

[6] Rodrigo Benenson et al. 'Ten years of pedestrian detection, what have we learned?' In: *arXiv preprint arXiv:1411.4304* (2014).

[7] Jerome Berclaz et al. 'Multiple object tracking using k-shortest paths optimization'. In: *IEEE transactions on pattern analysis and machine intelligence* 33.9 (2011), pp. 1806–1819.

[8] Gary Bishop and Greg Welch. 'An introduction to the kalman filter'. In: *Proc of SIGGRAPH, Course* 8.27599-23175 (2001), p. 41.

[9] Hermann Borotschnig, Lucas Paletta and Axel Pinz. 'A comparison of probabilistic, possibilistic and evidence theoretic fusion schemes for active object recognition'. In: *Computing* 62.4 (1999), pp. 293–319.

[10] Bernhard E Boser, Isabelle M Guyon and Vladimir N Vapnik. 'A training algorithm for optimal margin classifiers'. In: *Proceedings of the fifth annual workshop on Computational learning theory*. ACM. 1992, pp. 144–152.

[11] Thierry Bouwmans et al. *Background Modeling and Foreground Detection for Video Surveillance*. CRC press, 2014.

[12] Denis Bouyssou et al. *Decision making process: Concepts and methods*. John Wiley & Sons, 2013.

[13] Boris R Bracio, Wolfgang Horn and Dietmar PF Moller. 'Sensor fusion in biomedical systems'. In: *Engineering in Medicine and Biology Society, 1997. Proceedings of the 19th Annual International Conference of the IEEE*. Vol. 3. IEEE. 1997, pp. 1387–1390.

[14] Derek Bradley and Gerhard Roth. 'Adaptive thresholding using the integral image'. In: *Journal of Graphics Tools* 12.2 (2007), pp. 13–21.

[15] Ming-Ming Cheng et al. 'BING: Binarized normed gradients for objectness estimation at 300fps'. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 3286–3293.

[16] Corinna Cortes and Vladimir Vapnik. 'Support-vector networks'. In: *Machine learning* 20.3 (1995), pp. 273–297.

[17] Dan Crisan and Arnaud Doucet. 'A survey of convergence results on particle filtering methods for practitioners'. In: *IEEE Transactions on signal processing* 50.3 (2002), pp. 736–746.

[18] Navneet Dalal. 'Finding people in images and videos'. PhD thesis. Institut National Polytechnique de Grenoble-INPG, 2006.

[19] Navneet Dalal and Bill Triggs. 'Histograms of oriented gradients for human detection'. In: *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. Vol. 1. IEEE. 2005, pp. 886–893.

[20] Navneet Dalal and Bill Triggs. 'Histograms of oriented gradients for human detection'. In: *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. Vol. 1. IEEE. 2005, pp. 886–893.

[21] Navneet Dalal, Bill Triggs and Cordelia Schmid. 'Human detection using oriented histograms of flow and appearance'. In: *European conference on computer vision*. Springer. 2006, pp. 428–441.

[22] Nando De Freitas et al. 'Sequential Monte Carlo methods for neural networks'. In: *Sequential Monte Carlo methods in practice*. Springer, 2001, pp. 359–379.

[23] Piotr Dollár, Serge J Belongie and Pietro Perona. 'The Fastest Pedestrian Detector in the West.' In: *BMVC*. Vol. 2. 3. 2010, p. 7.

[24] Piotr Dollar et al. 'Pedestrian detection: An evaluation of the state of the art'. In: *IEEE transactions on pattern analysis and machine intelligence* 34.4 (2012), pp. 743–761.

[25] Piotr Dollár et al. 'Fast feature pyramids for object detection'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36.8 (2014), pp. 1532–1545.

[26] Piotr Dollár et al. 'Feature mining for image classification'. In: *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*. IEEE. 2007, pp. 1–8.

[27] Piotr Dollár et al. 'Multiple component learning for object detection'. In: *Computer Vision–ECCV 2008* (2008), pp. 211–224.

[28] Piotr Dollár et al. 'Pedestrian detection: A benchmark'. In: *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE. 2009, pp. 304–311.

[29] P. Dollár et al. 'Integral Channel Features'. In: *BMVC*. 2009.

[30] P. Dollár et al. 'Pedestrian Detection: A Benchmark'. In: *CVPR*. June 2009.

[31] Peilei Dong and Wenmin Wang. 'Better region proposals for pedestrian detection with R-CNN'. In: *Visual Communications and Image Processing (VCIP), 2016*. IEEE. 2016, pp. 1–4.

[32] Alessio Dore, Matteo Pinasco and C Regazzoni. 'Multi-modal data fusion techniques and applications'. In: *Multi-Camera Networks: Principles and Applications* 9 (2009), pp. 213–237.

[33] Didier Dubois and Henri Prade. 'Possibility theory and data fusion in poorly informed environments'. In: *Control Engineering Practice* 2.5 (1994), pp. 811–823.

[34] Hugh Durrant-Whyte. 'Introduction to sensor data fusion'. In: *University of Sydney* (2002).

[35] Hugh Durrant-Whyte and Thomas C Henderson. 'Multisensor data fusion'. In: *Springer Handbook of Robotics*. Springer, 2008, pp. 585–610.

[36] Markus Enzweiler and Dariu M Gavrila. 'Monocular pedestrian detection: Survey and experiments'. In: *IEEE transactions on pattern analysis and machine intelligence* 31.12 (2009), pp. 2179–2195.

[37] P Jorge Escamilla-Ambrosio and Neil Mort. 'Hybrid Kalman filter-fuzzy logic adaptive multisensor data fusion architectures'. In: *Decision and Control, 2003. Proceedings. 42nd IEEE Conference on*. Vol. 5. IEEE. 2003, pp. 5215–5220.

[38] Martin Ester et al. 'A density-based algorithm for discovering clusters in large spatial databases with noise.' In: *Kdd*. Vol. 96. 34. 1996, pp. 226–231.

[39] Vladimir Estivill-Castro. 'Why so many clustering algorithms: a position paper'. In: *ACM SIGKDD explorations newsletter* 4.1 (2002), pp. 65–75.

[40] Pedro F Felzenszwalb et al. 'Object detection with discriminatively trained part-based models'. In: *IEEE transactions on pattern analysis and machine intelligence* 32.9 (2010), pp. 1627–1645.

[41] Pedro Felzenszwalb, David McAllester and Deva Ramanan. 'A discriminatively trained, multiscale, deformable part model'. In: *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE. 2008, pp. 1–8.

[42] Martin A Fischler. 'An inference technique for integrating knowledge from disparate sources'. In: *Multisensor integration and fusion for intelligent machines and systems* (1995), p. 309.

[43] Mihai C Florea, Anne-Laure Jousselme and Éloi Bossé. *Fusion of imperfect information in the unified framework of random sets theory: application to target identification*. Tech. rep. DEFENCE RESEARCH and DEVELOPMENT CANADA VALCARTIER (QUEBEC), 2007.

[44] Hassen Fourati. *Multisensor Data Fusion: From Algorithms and Architectural Design to Applications*. CRC Press, 2015.

[45] Jerome Friedman, Trevor Hastie, Robert Tibshirani et al. 'Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors)'. In: *The annals of statistics* 28.2 (2000), pp. 337–407.

[46] Jeff Frolik, Mohamed Abdelrahman and Param Kandasamy. 'A confidence-based approach to the self-validation, fusion and reconstruction of quasi-redundant sensor data'. In: *IEEE Transactions on Instrumentation and Measurement* 50.6 (2001), pp. 1761–1769.

[47] Kunihiko Fukushima and Sei Miyake. 'Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition'. In: *Competition and cooperation in neural nets*. Springer, 1982, pp. 267–285.

[48] David Geronimo et al. 'Survey of pedestrian detection for advanced driver assistance systems'. In: *IEEE transactions on pattern analysis and machine intelligence* 32.7 (2010), pp. 1239–1258.

[49] Ross Girshick et al. 'Region-based convolutional networks for accurate object detection and segmentation'. In: *IEEE transactions on pattern analysis and machine intelligence* 38.1 (2016), pp. 142–158.

[50] Irwin R Goodman, Ronald P Mahler and Hung T Nguyen. *Mathematics of data fusion*. Vol. 37. Springer Science & Business Media, 2013.

[51] Neil J Gordon, David J Salmond and Adrian FM Smith. 'Novel approach to nonlinear/non-Gaussian Bayesian state estimation'. In: *IEE Proceedings F (Radar and Signal Processing)*. Vol. 140. 2. IET. 1993, pp. 107–113.

[52] Wang Haijun and Chen Yimin. 'Sensor data fusion using rough set for mobile robots system'. In: *Mechatronic and Embedded Systems and Applications, Proceedings of the 2nd IEEE/ASME International Conference on*. IEEE. 2006, pp. 1–5.

[53] Robert M Haralick, Karthikeyan Shanmugam et al. 'Textural features for image classification'. In: *IEEE Transactions on systems, man, and cybernetics* 6 (1973), pp. 610–621.

[54] Jeroen D Hol, Thomas B Schon and Fredrik Gustafsson. 'On resampling algorithms for particle filters'. In: *Nonlinear Statistical Signal Processing Workshop, 2006 IEEE*. IEEE. 2006, pp. 79–82. URL: `http://ieeexplore.ieee.org/abstract/document/4378824/`.

[55] Michael Hoy et al. 'Bayesian tracking of multiple objects with vision and radar'. In: *Control, Automation, Robotics and Vision (ICARCV), 2016 14th International Conference on*. IEEE. 2016, pp. 1–6.

[56] Pablo H Ibarguengoytia, Luis Enrique Sucar and Sunil Vadera. 'Real time intelligent sensor validation'. In: *IEEE Transactions on Power Systems* 16.4 (2001), pp. 770–775.

[57] Rudolph Emil Kalman et al. 'A new approach to linear filtering and prediction problems'. In: *Journal of basic Engineering* 82.1 (1960), pp. 35–45.

[58] Tapas Kanungo et al. 'An efficient k-means clustering algorithm: Analysis and implementation'. In: *IEEE transactions on pattern analysis and machine intelligence* 24.7 (2002), pp. 881–892.

[59] Amit Kumar KC et al. 'Iterative hypothesis testing for multi-object tracking with noisy/missing appearance features'. In: *Asian Conference on Computer Vision*. Springer. 2012, pp. 412–426.

[60] Bahador Khaleghi et al. 'Multisensor data fusion: A review of the state-of-the-art'. In: *Information Fusion* 14.1 (2013), pp. 28–44.

[61] Du Yong Kim and Moongu Jeon. 'Data fusion of radar and image measurements for multi-object tracking via Kalman filtering'. In: *Information Sciences* 278 (2014), pp. 641–652.

[62] Genshiro Kitagawa. 'Monte Carlo filter and smoother for non-Gaussian nonlinear state space models'. In: *Journal of computational and graphical statistics* 5.1 (1996), pp. 1–25.

[63] Chris Kreucher, Keith Kastella and Alfred O Hero III. 'Tracking multiple targets using a particle filter representation of the joint multitarget probability density'. In: *Proc. of SPIE Vol.* Vol. 5204. 2004, pp. 258–269.

[64] Harold W Kuhn. 'The Hungarian method for the assignment problem'. In: *Naval Research Logistics (NRL)* 2.1-2 (1955), pp. 83–97.

[65] Manish Kumar, Devendra P Garg and Randy A Zachery. 'A generalized approach for inconsistency detection in data fusion from multiple sensors'. In: *American Control Conference, 2006*. IEEE. 2006, 6–pp.

[66] Manish Kumar, Devendra Garg and Randy Zachery. 'Stochastic adaptive sensor modeling and data fusion'. In: *Proceedings of SPIE Conference on Sensors and Smart Structures Technologies for Civil, Mechanical, and Aerospace Systems*. 2006, pp. 61740C–1.

[67] Laura Leal-Taixé et al. 'Motchallenge 2015: Towards a benchmark for multi-target tracking'. In: *arXiv preprint arXiv:1504.01942* (2015).

[68] Yann LeCun, Yoshua Bengio et al. 'Convolutional networks for images, speech, and time series'. In: *The handbook of brain theory and neural networks* 3361.10 (1995), p. 1995.

[69] Xin Li et al. 'A multiple object tracking method using Kalman filter'. In: *Information and Automation (ICIA), 2010 IEEE International Conference on*. IEEE. 2010, pp. 1862–1866.

[70] Martin Liggins II, David Hall and James Llinas. *Handbook of multisensor data fusion: theory and practice*. CRC press, 2017.

[71] Jun S Liu and Rong Chen. 'Sequential Monte Carlo methods for dynamic systems'. In: *Journal of the American statistical association* 93.443 (1998), pp. 1032–1044.

[72] Ronald Mahler. 'Random sets: Unification and computation for information fusion-a retrospective assessment'. In: *Proceedings of the Seventh International Conference on Information Fusion*. Vol. 1. 2004, pp. 1–20.

[73] Ronald PS Mahler. *Statistical multisource-multitarget information fusion*. Artech House, Inc., 2007.

[74] Ronald PS Mahler. '" Statistics 101" for multisensor, multitarget data fusion'. In: *IEEE Aerospace and Electronic Systems Magazine* 19.1 (2004), pp. 53–64.

[75] Subhransu Maji, Alexander C Berg and Jitendra Malik. 'Classification using intersection kernel support vector machines is efficient'. In: *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE. 2008, pp. 1–8.

[76] Jitendra Malik and Pietro Perona. 'Preattentive texture discrimination with early vision mechanisms'. In: *JOSA A* 7.5 (1990), pp. 923–932.

[77] Anton Milan et al. 'MOT16: A benchmark for multi-object tracking'. In: *arXiv preprint arXiv:1603.00831* (2016).

[78] XIA Mingge et al. 'Image fusion algorithm using rough sets theory and wavelet analysis'. In: *Signal Processing, 2004. Proceedings. ICSP'04. 2004 7th International Conference on*. Vol. 2. IEEE. 2004, pp. 1041–1044.

[79] Amihai Motro and Philippe Smets. *Uncertainty management in information systems: from needs to solutions*. Springer Science & Business Media, 2012.

[80] Woonhyun Nam, Piotr Dollár and Joon Hee Han. 'Local decorrelation for improved detection'. In: *Eprint Arxiv* (2014).

[81] Timo Ojala, Matti Pietikainen and Topi Maenpaa. 'Multiresolution gray-scale and rotation invariant texture classification with local binary patterns'. In: *IEEE Transactions on pattern analysis and machine intelligence* 24.7 (2002), pp. 971–987.

[82] Kenji Okuma et al. 'A boosted particle filter: Multitarget detection and tracking'. In: *European Conference on Computer Vision*. Springer. 2004, pp. 28–39.

[83] Wanli Ouyang and Xiaogang Wang. 'Single-pedestrian detection aided by multi-pedestrian detection'. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2013, pp. 3198–3205.

[84] Sakrapee Paisitkriangkrai, Chunhua Shen and Anton van den Hengel. 'Strengthening the effectiveness of pedestrian detection with spatially pooled features'. In: *European Conference on Computer Vision*. Springer. 2014, pp. 546–561.

[85] Constantine Papageorgiou and Tomaso Poggio. 'A trainable system for object detection'. In: *International Journal of Computer Vision* 38.1 (2000), pp. 15–33.

[86] Dennis Park et al. 'Exploring weak stabilization for motion feature extraction'. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2013, pp. 2882–2889.

[87] Zdzisław Pawlak. *Rough sets: Theoretical aspects of reasoning about data*. Vol. 9. Springer Science & Business Media, 2012.

[88] AG Amitha Perera et al. 'Multi-object tracking through simultaneous long occlusions and split-merge conditions'. In: *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*. Vol. 1. IEEE. 2006, pp. 666–673.

[89] Fatih Porikli. 'Integral histogram: A fast way to extract histograms in cartesian spaces'. In: *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. Vol. 1. IEEE. 2005, pp. 829–836.

[90] *RFBeam Microwave GmbH, Digital Radar Specification & Command-Description*. K-MD2. St.Gallen. RFBeam. Apr. 2016.

[91] JZ Sasiadek and P Hartana. 'Sensor data fusion using Kalman filter'. In: *Information Fusion, 2000. FUSION 2000. Proceedings of the Third International Conference on*. Vol. 2. IEEE. 2000, WED5–19.

[92] Friedhelm Schwenker and Günther Palm. 'Tree-structured support vector machines for multi-class pattern recognition'. In: *International Workshop on Multiple Classifier Systems*. Springer. 2001, pp. 409–417.

[93] Daniel J Sebald and James A Bucklew. 'Support vector machine techniques for nonlinear equalization'. In: *IEEE Transactions on Signal Processing* 48.11 (2000), pp. 3217–3226.

[94] John Shackleton, Brian VanVoorst and Joel Hesch. 'Tracking people with a 360-degree lidar'. In: *Advanced Video and Signal Based Surveillance (AVSS), 2010 Seventh IEEE International Conference on*. IEEE. 2010, pp. 420–426.

[95] Glenn Shafer et al. *A mathematical theory of evidence*. Vol. 1. Princeton university press Princeton, 1976.

[96] Eli Shechtman and Michal Irani. 'Matching local self-similarities across images and videos'. In: *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*. IEEE. 2007, pp. 1–8.

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

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

[99] Denis Tomè et al. 'Deep convolutional neural networks for pedestrian detection'. In: *Signal Processing: Image Communication* 47 (2016), pp. 482–489.

[100] Jasper RR Uijlings et al. 'Selective search for object recognition'. In: *International journal of computer vision* 104.2 (2013), pp. 154–171.

[101] Israel Veilleux et al. 'In vivo cell tracking with video rate multimodality laser scanning microscopy'. In: *IEEE Journal of selected topics in quantum electronics* 14.1 (2008), pp. 10–18.

[102] Paul Viola and Michael J Jones. 'Robust real-time face detection'. In: *International journal of computer vision* 57.2 (2004), pp. 137–154.

[103] Paul Viola and Michael J Jones. 'Robust real-time face detection'. In: *International journal of computer vision* 57.2 (2004), pp. 137–154.

[104] Paul Viola, Michael J Jones and Daniel Snow. 'Detecting pedestrians using patterns of motion and appearance'. In: *null*. IEEE. 2003, p. 734.

[105] SJ Wellington, JK Atkinson and RP Sion. 'Sensor validation and fusion using the Nadaraya-Watson statistical estimator'. In: *Information Fusion, 2002. Proceedings of the Fifth International Conference on*. Vol. 1. IEEE. 2002, pp. 321–326.

[106] Shiuh-Ku Weng, Chung-Ming Kuo and Shu-Kang Tu. 'Video object tracking using adaptive Kalman filter'. In: *Journal of Visual Communication and Image Representation* 17.6 (2006), pp. 1190–1208.

[107] Christian Wojek and Bernt Schiele. 'A performance evaluation of single and multi-feature people detection'. In: *Pattern Recognition* (2008), pp. 82–91.

[108] Zhenghao Xi et al. 'Multiple object tracking using A* association algorithm with dynamic weights'. In: *Journal of Intelligent & Fuzzy Systems* 29.5 (2015), pp. 2059–2072.

[109] Ronald R Yager. 'Generalized probabilities of fuzzy events from fuzzy belief structures'. In: *Information sciences* 28.1 (1982), pp. 45–62.

[110] Changjiang Yang, Ramani Duraiswami and Larry Davis. 'Fast multiple object tracking via a hierarchical particle filter'. In: *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*. Vol. 1. IEEE. 2005, pp. 212–219.

[111] John Yen. 'Generalizing the Dempster–Shafer theory to fuzzy sets'. In: *Classic Works of the Dempster-Shafer Theory of Belief Functions* (2008), pp. 529–554.

[112] Alper Yilmaz, Omar Javed and Mubarak Shah. 'Object tracking: A survey'. In: *Acm computing surveys (CSUR)* 38.4 (2006).

[113] Liu Yong, Xu Congfu and Pan Yunhe. 'A new approach for data fusion: implement rough set theory in dynamic objects distinguishing and tracing'. In: *Systems, Man and Cybernetics, 2004 IEEE International Conference on*. Vol. 4. IEEE. 2004, pp. 3318–3322.

[114] Shanshan Zhang, Christian Bauckhage and Armin B Cremers. 'Informed haar-like features improve pedestrian detection'. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 947–954.

[115] Shanshan Zhang, Rodrigo Benenson and Bernt Schiele. 'Filtered channel features for pedestrian detection'. In: *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*. IEEE. 2015, pp. 1751–1760.

[116] Shanshan Zhang et al. 'Exploring human vision driven features for pedestrian detection'. In: *IEEE Transactions on Circuits and Systems for Video Technology* 25.10 (2015), pp. 1709–1720.

[117] Shanshan Zhang et al. 'How Far are We from Solving Pedestrian Detection?' In: *CoRR* abs/1602.01237 (2016). URL: `http://arxiv.org/abs/1602.01237`.

[118] Hongwei Zhu and O Basir. 'A novel fuzzy evidential reasoning paradigm for data fusion with applications in image processing'. In: *Soft Computing-A Fusion of Foundations, Methodologies and Applications* 10.12 (2006), pp. 1169–1180.

[119] Qiang Zhu et al. 'Fast human detection using a cascade of histograms of oriented gradients'. In: *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*. Vol. 2. IEEE. 2006, pp. 1491–1498.

[120] C Lawrence Zitnick and Piotr Dollár. 'Edge boxes: Locating object proposals from edges'. In: *European Conference on Computer Vision*. Springer. 2014, pp. 391–405.

Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve **www.uclouvain.be/epl**