UCL Université catholique de Louvain

École polytechnique de Louvain (EPL)

People detection using time-of-flight camera

In collaboration with **AUTOMATIC SYSTEMS**

Dissertation presented by
Florian STÉVENART MEEÛS , Camille UYLENBROECK

for obtaining the Master's degree in
Computer Science and Engineering

Supervisor
Christophe DE VLEESCHOUWER

Readers
Kevin DE CUYPER, Christophe DE VLEESCHOUWER , Benoît MACQ

Academic year 2017-2018

Acknowledgements

The completion of this thesis was only possible thanks to the contribution of many individuals. First of all, we would like to thank our exemplary thesis advisor, Professor Christophe De Vleeschouwer, for giving us the best possible guidance throughout the year. We are grateful for the time you generously spent leading us onto the right path, with patience and understanding. We would also like to express special thanks to *Automatic Systems*, especially to Yves Therasse and Kevin de Cuyper, for having proposed this subject and for helpfully following its development. We have found the topic to be immensely engaging and satisfying.

Furthermore, we are very grateful to our official reviewers, Luc Uylenbroeck, Steve Sturm and Anne Harrington for your generosity and determination to make this document clear and understandable.

Finally, we would like to thank our families and friends for supporting us throughout the long process of preparing this thesis, during periods of both discouragement and of passionate enthusiasm. This note of personal thanks is directed in particular to Kathleen, Paule, Olivier, Jonathan, Noémie, Marie and Clémence, among many others.

Florian Stévenart Meeûs and Camille Uylenbroeck, 2018

Abstract

We study the problem of detecting people using time-of-flight camera images, i.e. depth images. These detections are used to measure the accurate flow of people through a security gate. This is a major problem in the case of access control, such as in a metro station or an airport. The detection of people from depth images is challenging because of the variety of possible scenarios and implied shapes. The scenarios can involve people in a wheelchair, pushing bikes, or carrying suitcases, bags, etc. We show that the detection can be processed by filtering and classifying regions of interest around the local maxima of the depth images. We then propose to consolidate the detections across a scenario with multi-object tracking. The missing and extra people are measured for the performance analysis. Our algorithm is developed within Matlab, and trained on a dataset of top-view depth images of people crossing a metro access gate. Extensive validation demonstrates its effectiveness. Detection rates on stand-alone images larger than 80% are typically achieved with fewer than 10% false positives. Measurement of the people flow is correct in 90.91% of the scenarios.

Contents

Chapter i

List of abbreviations

CSM	Concentric Square Mean
CSP	Cutting Stock Problem
CV	Cross-Validation
DR	Detection Rate
FDI	Foreground Depth Image
FHCR	False Head Candidate Rate
FN	False Negative
FP	False Positive
FPR	False Positive Rate
HOG	Histogram of Oriented Gradients
I2	Integral Image
MOT	Multi Object Tracking
MOTA	Multi Object Tracking Analysis
N	Negative
P	Positive
PRM	Person with Reduced Mobility
ROC	Receiver Operating Characteristic
SMSP	Steel Mill Slab Problem
SVM	Support Vector Machine
ToF	Time of Flight
TN	True Negative
TP	True Positive

Chapter 1

Introduction

The aim of the algorithm developed for this thesis is to count the number of people crossing a gate, using the depth images captured by a top-view time-of-flight camera (see Figures 1.1 and 1.2). Typical measures of these installations can be found in Appendices, Section 9.1. This system can be particularly useful in controlling flows of people in a restricted area. These people can be adults, children, disabled or not. They can hold suitcases, bags, a bike, or other objects.

Figure 1.1: Photo of typical gate and camera

The system has been developed in collaboration with *Automatic Systems*, a world leader in the automation of secure entrance control. The company has provided the datasets used to train our systems.

This work targets real-life situations, such as metro stations or airports, possible customers of *Automatic Systems*. A typical passage through the gate consists of six consecutive actions : the opening of the entrance door, the admission of people into the gate, the closing of the entrance doors, the opening of the exit doors, the release of the people and the closing of the exit doors. The aim is to know how many people there are in the gate when all the doors are closed, in order to open the exit doors only if the payment matches the number of people. This

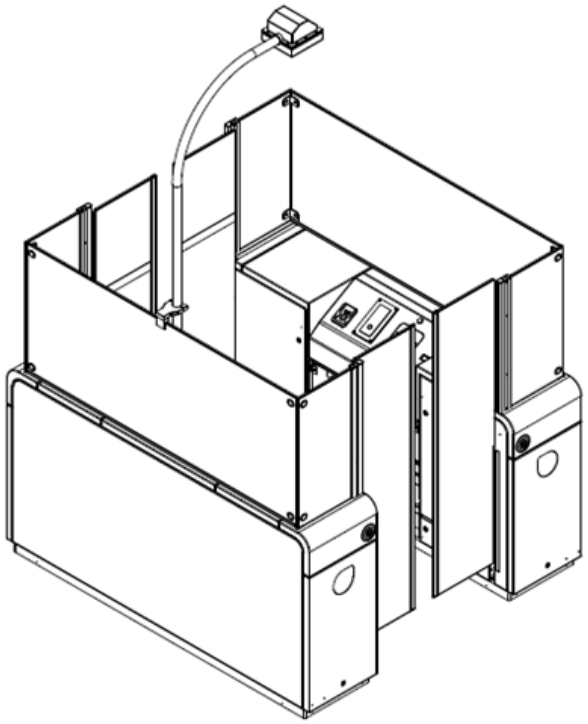

Figure 1.2: 3D picture of typical gate and camera

is where the system, developed within Matlab, comes into play.

The available data to develop and validate our algorithm corresponds to three datasets provided by *Automatic Systems*. Each dataset has been captured in a different site, and includes a variety of scenarios, each corresponding to a group of people (potentially carrying objects or adopting unusual poses) entering and leaving the gate. Our algorithm first counts the number of people in each image. It then estimates their trajectories by connecting detections across time, using a graph-based tracking algorithm. Finally, the number of consistent trajectories determines the number of people.

This thesis first presents, in Chapter 2, the datasets along with the preprocessing methods used to get a cleaner version of this material. The second part, in Chapter 3, describes two consecutive techniques to detect people on stand-alone depth images. Head candidates are selected among the local maxima and classified into two classes, *person* or *non-person*. Chapter 4 presents graph-based tracking to connect these detections across time to establish the trajectories of the people. The validation of these methods is then considered in Chapter 5, followed by their concrete results in Chapter 6. The thesis concludes, in Chapter 7, with recommendations for further work to better suit the objective.

The main contributions brought along with this thesis are some processing techniques to achieve detection over depth maps. First, we propose some heuristics to select points of interest among local maxima in Gaussian filtered depth images. Then, we propose the combination of some specific features to characterise a region of interest on a depth image and to feed a linear classifier: the Concentric Squares Mean to characterise the progressive height difference and the Histogram of Oriented Gradients to characterise the orientation of the slopes.

Chapter 2

Dataset and image preprocessing

2.1 Introduction

This chapter first presents the datasets made available by *Automatic Systems*, along with their composition, characteristics and differences. They are used to train and test the approaches presented in the following chapters. The second part addresses some preprocessing techniques used to obtain the foreground depth image and remove as much noise as possible from each original depth image.

2.2 Dataset

Three datasets were made available by *Automatic Systems*, each composed of multiple folders including the images corresponding to a specific passage of people into the gate, i.e. to a specific *scenario*. Each scenario is composed of a succession of images taken by the time-of-flight (ToF) camera. A ToF image is accompanied by a confidence image, in which each pixel defines the level of reliability associated to the corresponding ToF pixel. This confidence image is useful to detect noise and irrelevant measurements in the ToF image.

In practice, each image has the following characteristics:

- The file *radial.pgm* is the depth map, i.e. the main image taken by the camera. The value of each image pixel indicates its distance to the camera : the smaller the pixel value, the closer the corresponding point (visually, the darker the pixel). The dimension of this image is 132x176 pixels. Each pixel value is expressed as an unsigned integer in the range [0, 65535]. An example of such a file is presented in Figure 2.1.

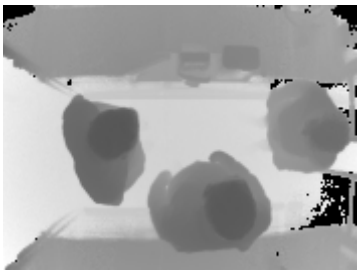

Figure 2.1: Example of a radial image, illustrating the distance to the camera

- The file *confidence.pgm* is an additional file containing the confidence image indicating the confidence of the camera, for each pixel. It then has the same dimensions as the radial image, but within an additional one, expressing the colours of the pixel : 132x176x3. The

value of each pixel is then expressed by a colour, formed by the combination of the three values, being into the interval [0, 255]. Visually, if the pixel is red, there is probably some noise and probably none if the pixel is green. An example of such a confidence image is illustrated in Figure 2.2.

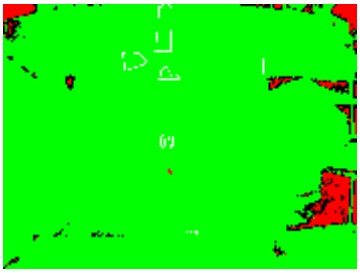

Figure 2.2: Example of a confidence image

The three sets considered in our study differ in some ways, summarised in Table 2.1:

- The first set was received in August 2017. It consists in 3103 depth maps derived from 72 scenarios, each one corresponding to a distinct flow of people through the gate, as observed experimentally in the buildings of *Automatic Systems*. Each scenario folder also contains a csv file that indicates the numbers of adults, children, PRMs and objects that are present in each image.
- The second set was received in October 2017. It consists in 32 020 depth maps, for 121 scenarios, each one corresponding to a distinct flow of people through the gate, as observed experimentally at a metro station of Lille. This place has a marble floor, that has the particularity to cause some reflections on the images of this set. A csv file is also joined to characterise the scenarios (person with bag or not, turnover or normal passages, number of people.).
- The third set was received in February 2018. It consists in more than 90 000 depth maps, for 408 scenarios, each one describing a distinct flow of people through the gate, as observed in real-life at the same metro station of Lille than the second set. The same type of csv file is also joined.

Sets	1	2	3
Scenarios	72	121	408
Depth maps	3103	32 020	+90 000
Situation	Experimental in labs	Experimental in metro station	Real life in metro station
Reception date	August 2017	October 2017	February 2018

Table 2.1: Summary of the datasets

2.3 Image preprocessing

Before counting the number of people, we need to apply some preprocessing steps on the images, in order to get a version more suitable for the application of the techniques presented in the following chapters. It implies removing the background and as much noise as possible from these images. A technique to transform the distance from the camera to an actual height is also presented. It is particularly useful to get more precise information and will be used later in the document.

2.3.1 Background removal

The first operation to apply on the radial images is the subtraction of the background, either by recording an image of the empty gate in the same condition than the scenario concerned, or by computing the background from the scenario itself with the deepest value observed for each pixel. We define the operation of subtracting the background from a given image as follows: the pixels having a value close to the value of these pixels in the background are fixed to the deepest value, i.e. 65535 (white on the image). This closeness is estimated at 5% of the deepest value.

$$\begin{aligned} image[X][Y] &= image[X][Y] && \text{if } |image[X][Y] - background[X][Y]| \geq 0.05 \times 65535 \\ image[X][Y] &= 65535 && \text{otherwise} \end{aligned}$$

This solution was preferred to a real mathematical subtraction to keep the foreground pixels untouched. As a definition of the 'close' notion, we selected the 5% maximum depth threshold. Furthermore, since we are only interested by the highest point of a person, the area cut by this operation is not significant and it succeeds to eliminate slight variations, due to some condition changes (light, vibration, etc.).

2.3.2 Noise removal

Then, to eliminate as much noise as possible, the confidence file is used to determine the low-confidence pixels and replace them by higher confidence neighbours thanks to the following operation :

$$image[X][Y] = image[X][Y] \text{ if } confidence[1][X][Y] <= 150$$
$$image[X][Y] = image[closeX][closeY] \text{ if } confidence[1][X][Y] > 150$$

($closeX$, $closeY$) is the closest point respecting $confidence[1][closeX][closeY] < 100$.

Indeed, the first element of the first dimension of the image represents the colour red. It implies that the higher the value, the less reliable the pixel is, with a maximum of 255. The new version of the images are called foreground depth images. An example is illustrated in Figure 2.3.

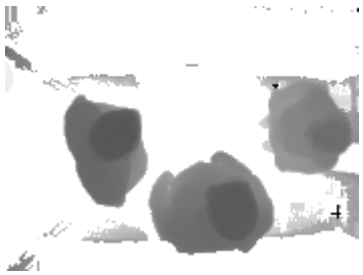

Figure 2.3: Example of a foreground depth image

2.3.3 Transforming the distance between the camera and a point to an actual height

Finally, there is a last preprocessing technique that can be used. Indeed, each pixel of the radial file represents the distance to the camera. It implies that the points that have the same height (from the ground) can have different values on the radial image, depending of their positions. This particularity 'bends' the image and can create problems. Indeed, when the camera is relatively low, the depth profile surrounding a head might significantly depends on the position of the head in the image due to self-occlusions.

A way to reduce the variability of head profiles appearance with the location of the head in the scene consists in rectifying the radial image, by replacing the distance to the camera by the distance to the horizontal place passing through the camera. Due to self-occlusions, this process however results in an irregular sampling of the rectified image, requiring interpolation or padding to reconstruct a rectified image on a regular grid. Therefore, we have decided to ignore this opportunity to rectify the image and focus on cases for which the camera is reasonably high compared to the head (typically above 3 metres) to limit the head profile appearance variability.

However, rectifying the actual height of a particular point can be useful. Indeed, knowing the precise height of a head is a requirement in multiple situations, to differentiate a child (according to *Automatic Systems* specifications, a person smaller than 1m30) from an adult for example. To compute this height, we use the Pythagorean theorem thanks to the depth expressed on the image and the horizontal distance between the head of a person and the camera (L), as illustrated in Figure 2.4. This computation is made trickier by the units of the image, that we have to transform to millimetres, as in the following equations.

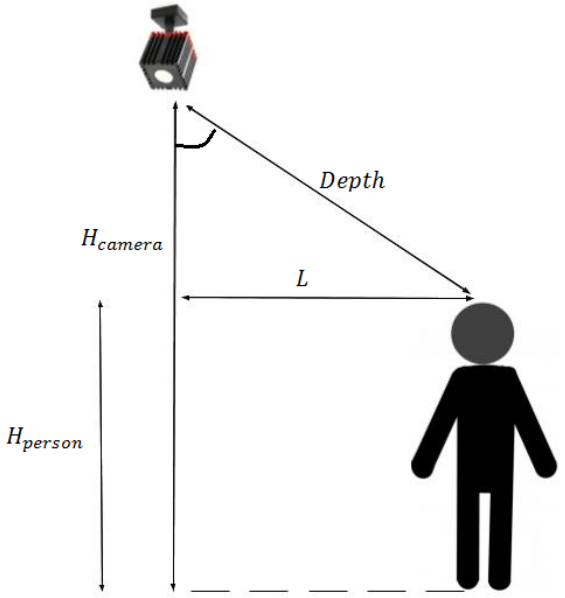

Figure 2.4: Schema of a typical camera-to-point situation

$$
\begin{aligned}
H_{person} &= H_{camera} - \sqrt{Depth_{mm}^2 - L_{mm}^2} \\
Depth_{mm} &= Depth_{pixel} * C_{depthToMm} \\
C_{depthToMm} &= \frac{Max_{mm}}{65\ 535} \\
Max_{mm} &= \sqrt{H_{camera}^2 + C_{pixelToMm}^2 * (\frac{W_{image\_pixel}}{2}^2 + \frac{L_{image\_pixel}}{2}^2)} \\
C_{pixelToMm} &= \frac{L_{image\_mm}}{L_{image\_pixel}} \\
L_{mm} &= \sqrt{d1_{mm}^2 + d2_{mm}^2} \\
d1,2_{mm} &= d1,2_{pixel} * C_{pixelToMm}
\end{aligned}
$$

For that, we need two coefficients : $C_{depthToMm}$ and $C_{pixelToMm}$. $C_{depthToMm}$ does the transition from the number of pixels to millimetres, by being the ratio between the maximal depth on the image (65 535) and the maximal (and furthest) depth in millimetres (Max_{mm}). This last value can be computed by another Pythagorean theorem, by taking into account the height of the camera (the maximum depth) and the distance to the furthest possible point (one of the four squares of the image). The horizontal distance between the camera and this point can be computed with again another Pythagorean theorem, from the width and the length of the image (respectively W_{image} and L_{image}), translated from a number of pixels to millimetres. Exactly the same technique is used to compute the horizontal distance between the head of the person and the camera (with the horizontal and vertical distances being d1 and d2). The conversion is done thanks to the second coefficient, $C_{pixelToMm}$ which is the ratio between the width of the image in millimetres and the width of the image in pixel. The used values that stay unchangeable for all people are expressed in Table 2.2.

Variable	H_{camera}	$C_{depthToMm}$	$C_{pixelToMm}$	Max_{mm}	L_{image_mm}	W_{image_pixel}	L_{image_pixel}
Value	3000	0.049	10.227	3204	1350	176	132

Table 2.2: Values of the variable used to get the actual height of a fixed point

2.4 Conclusion

The datasets are very different, whether in terms of the number of scenarios, the average number of images by scenarios or their production situation. Two preprocessing techniques remove the background and the noise from the depth images, using the radial and confidence files. They generate the foreground depth images, which are the primal inputs of the techniques presented in the following chapters.
The third presented technique allows to transform the distance to the camera of a particular point to an actual height in millimetres. This is an important measure, independent of the position on the image and, above all, understandable by humans.

Chapter 3

Formulating detection as a classification problem

3.1 Introduction

The purpose of this chapter is to describe ways of properly detecting and locating the people on every image. To achieve this goal, we aim at distinguishing the pixels that describe a head from those that do not. Specifically, we reduce this problem to the classification of a number of head candidates, defined as the local maxima of the foreground depth images. By limiting head candidates to local maxima, we are proceeding on the assumption that the head is the highest point of the body of a person (disabled or not) or at least one of the local maxima of the body (if the person raises a hand for example).

The algorithm presented in this Chapter works in two steps. The first one, presented in Section 3.2, consists of extracting the local maxima from the Gaussian filtered depth images. Then the head candidates are selected from among these maxima, based on a set of heuristics.

The second step, presented in Section 3.3, consists of classifying the selected head candidates with a Support Vector Machine, based on features characterising the candidate neighbourhood in the depth image.

3.2 Head candidates selection

3.2.1 Convolution with a Gaussian filter

The first step to select head candidates is to apply a preprocessing filter to smooth the foreground depth images. To do so, we convolve each image with an isotropic Gaussian filter [1] [2] having an impulse response given by $g(x, y)$ and a truncated support, as presented at Figure 3.1. We chose an isotropic filter because we do not want to exploit the head profile anisotropy since the orientation of the head is unknown (because the people can turn in the gate, by example to scan their card on the side). The filter function is evaluated on a grid of size $N \times N$ and depends of the standard deviation σ. To conserve a scope relatively similar through different sizes, we make the standard deviation proportional to the size of the filter, following the formula 3.1. The sigma value given by this formula is a good indication of typical standard deviation of a Gaussian filter.

The filter size impacts the level of smoothing applied on the image. The larger the filter, the smoother the resulting convolution. Typical values of the size N depend of the purpose of the filtering. To study the characteristics relative to the head of a person, a size of 15 can be appropriate since it matches the size of such a head on the depth image. On the other hand, if

$$g(x,y) = \frac{1}{2\pi\sigma^2}\ e^{-\frac{x^2+y^2}{2\sigma^2}}$$

$$\sigma = \sqrt{\frac{N^2-1}{12}} \qquad (3.1)$$

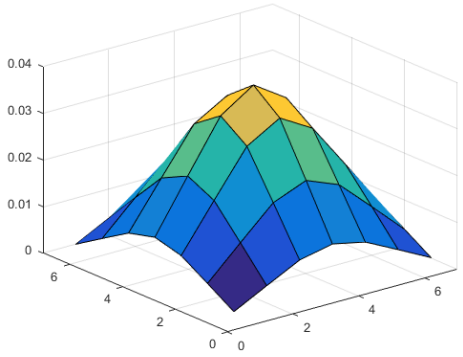

Figure 3.1: Gaussian filter of size 7×7

the purpose is to filter the noise on the depth image while keeping all the singularities, then a size of 3 or 5 is more appropriate. This is highlighted on the following illustrations: Figure 3.2 shows the top-view foreground depth image of one person, Figure 3.3 shows the same image on a 3D view, Figure 3.4 shows the image filtered by a Gaussian of size 3 and finally Figure 3.5 shows the image filtered by a Gaussian of size 7.

3.2.2 Maxima selection

We propose four heuristics to select head candidates among the local maxima of the Gaussian filtered image. Each heuristic is a function that considers either one or two maxima and returns zero if neither of the two should be rejected, one (resp. two) if the first (resp. second) maxima should be rejected. Furthermore, the heuristics depend on some parameters, for which a summary is available in Table 6.2.

Highland heuristic

We propose as first heuristic the elimination of the peak-shaped local maxima. Indeed, a typical maximum derived from a person head is surrounded by a lower large surface, resulting from the rest of the head and the shoulders. Furthermore, the maxima arising from objects or from the noise are usually more peak-shaped.

The principle of this heuristic is to measure the fraction of the Gaussian filtered image lying above a threshold in a square window centered at the local maximum. If this fraction is not sufficient, the maximum is rejected. The behaviour of this heuristic is illustrated in Figures 3.6 and 3.7.

$$Reject_{Highland}(x_1,y_1) = \begin{cases} 1 & \text{iff } \dfrac{\sum_{x',y' \mid (x',y') \in window(x_1,y_1)} (image(x',y') > C.image(x_1,y_1))}{WS^2} < RS \\ 0 & \text{Otherwise} \end{cases}$$

$$window(x_1,y_1) = \{x',y' \mid x' \in [x_1 - \frac{WS-1}{2}\ ;\ x_1 + \frac{WS-1}{2}]\ ,\ y' \in [y_1 - \frac{WS-1}{2}\ ;\ y_1 + \frac{WS-1}{2}]\}$$

Figure 3.2: Foreground depth image

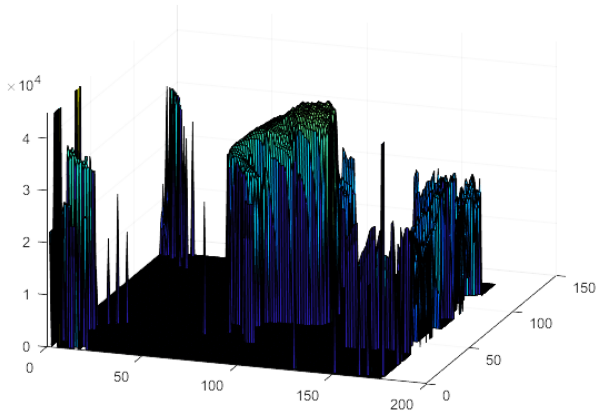

Figure 3.3: Primitive image - equivalent to the foreground depth image in 3D

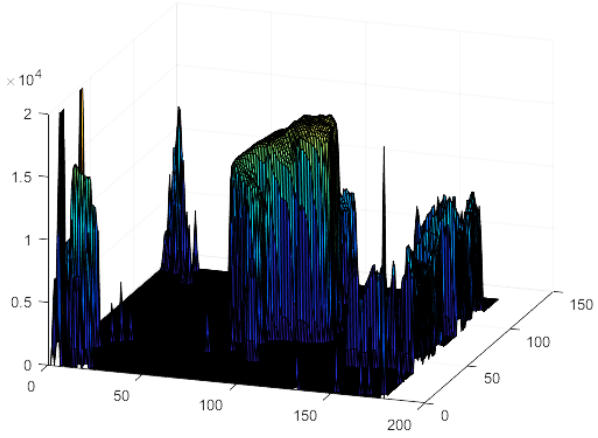

Figure 3.4: Convolved foreground depth image with a Gaussian filter of size 3

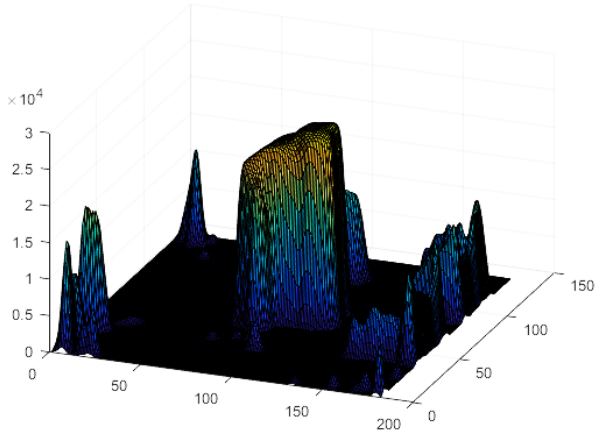

Figure 3.5: Convolved foreground depth image with a Gaussian filter of size 7

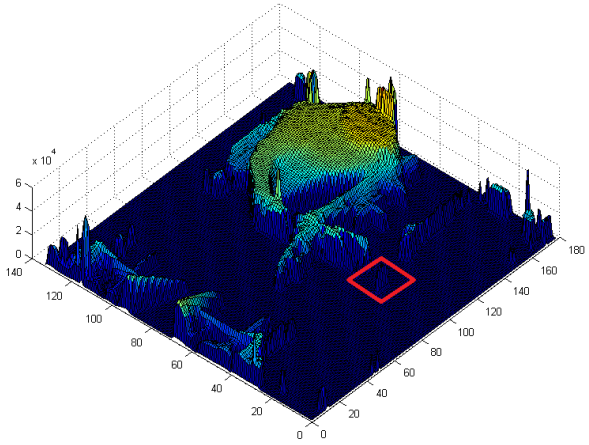

Figure 3.6: Example of the *Highland* heuristic window

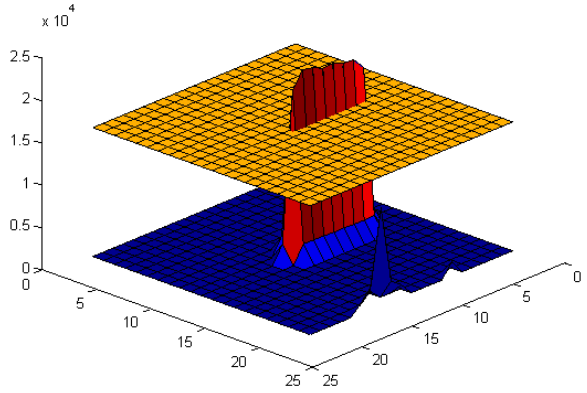

Figure 3.7: Example of a fixed percentage cut of the *Highland* heuristic

This method then depends of three parameters :

- The side of the square window, in pixels. Typical values are between 10 and 20 pixels, that roughly matches the average size of a head to the average size of the shoulders. We call it the *Window Size* (WS in the equation above).
- The cut height, in percentage. This value is multiplied by the height of the maximum, to produce the threshold of the method. Typically, values are between 60% and 80%, so that the remaining surface matches the upper part of the body. We call it the *Cut* (C in the equation above).
- The required surface, in percentage. Typical values are between 40% and 60%, according to the values of the two other parameters. We call it the *Required Surface* (RS in the equation above).

It is worth mentioning that another way exists to put into practice the principle of this heuristic. We could compute the surface (at the level of the cut), not on a pre-defined window, but on the connected component around the maximum. It is the size of this component, in squared millimetres, that would then need to be higher than a fixed required surface. This practice presents the advantage of having less parameters and being more precise. Indeed, we do not need the *Window Size* parameter. Furthermore, some maxima, that pass the present heuristic because there are several peaks non-connected in the window, would be eliminated. The main drawback, and the reason why we do not have applied this version of the heuristic, is the additional processing time. Indeed, computing a connected component is a more time-consuming method than simply summing the points on a window. It is however an interesting technique, if the computational resources are sufficient.

Height heuristic

This heuristic rejects the local maxima that are not sufficiently high to be a person. It then needs a threshold parameter named *Height* (He in the equation below). The heuristic simply eliminates all the maxima that have a height lower than this parameter, in millimetres, calculable from their value on the image by to the method described in the section 2.3.3.

$$Reject_{Height}(x_1, y_1) = \begin{cases} 1 & \text{iff } height_{mm}(x_1, y_1) < He \\ 0 & \text{Otherwise} \end{cases}$$

It is worth mentioning that the height parameter should be chosen, in this particular situation, to keep the maxima matching children and/or people in wheelchairs. Typical values for this parameter are between 900 and 1100. This interval can however vary according to the system utilisation and the kind of people that must be recognised.

Distance heuristic

This heuristic aims to prevent having two or more local maxima on the same head. To do so, it eliminates all the maxima that are too close to each others. In practice, this is done by rejecting local maxima that are close (i.e. distance smaller than a threshold) to another local maximum with a higher intensity. The impact of the heuristic is illustrated in Figure 3.8.

$$Reject_{Distance}(x_1, y_1, x_2, y_2) = \begin{cases} 2 & \text{iff } \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} < D \text{ and } image(x_1, y_1) \geq image(x_2, y_2) \\ 1 & \text{iff } \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} < D \text{ and } image(x_1, y_1) < image(x_2, y_2) \\ 0 & \text{Otherwise} \end{cases}$$

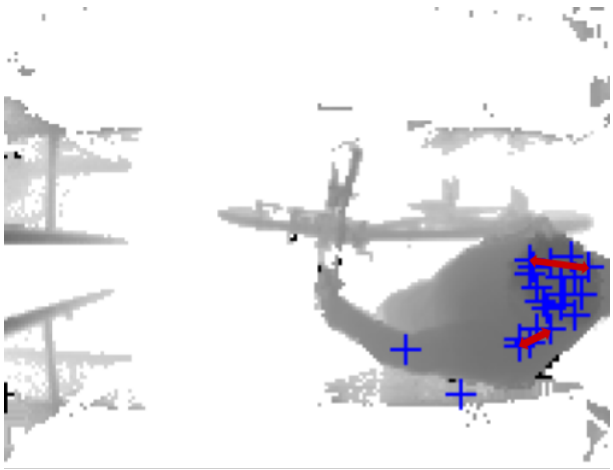

Figure 3.8: Example of the *Distance* heuristic

This heuristic depends of the *Distance* parameter (D in the equation above). Typical values are between 10 and 20 pixels, which respectively correspond to the typical largest distance between two points on the same head and to the largest distance between two points on the same body.

Hollow heuristic

This last heuristic ensures that any pair of distinct maxima is separated by a deep enough hollow. Given two maxima, if the minimum on the vertical cut going over both of them is not lower than threshold, than the lower maximum is eliminated. The impact of the heuristic is illustrated in Figures 3.9 and 3.10.

$$Reject_{Hollow}(x_1, y_1, x_2, y_2) = \begin{cases} 2 & \text{iff } hollow(x_1, y_1, x_2, y_2) > Ho \,.\, image(x_2, y_2) \\ & \quad \text{and } image(x_1, y_1) \geq image(x_2, y_2) \\ 1 & \text{iff } hollow(x_1, y_1, x_2, y_2) > Ho \,.\, image(x_1, y_1) \\ & \quad \text{and } image(x_1, y_1) < image(x_2, y_2) \\ 0 & \text{Otherwise} \end{cases}$$

$$hollow(x_1, y_1, x_2, y_2) = \min_{x',y' \mid (x',x') \in diag(x_1,y_1,x_2,y_2)} (image(x', y')$$

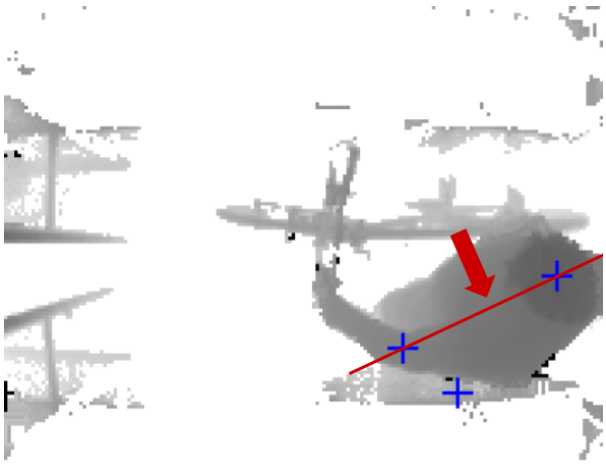

Figure 3.9: Example of a situation where the hollow heuristic acts

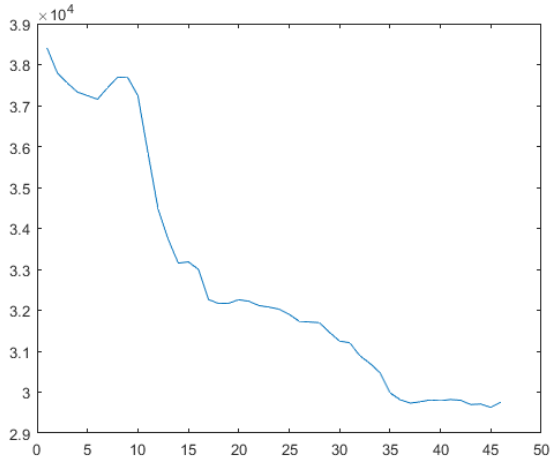

Figure 3.10: Example of the vertical cut between two maxima

The hollow heuristic depends of the *Hollow* parameter, the threshold under which there must be a minimum (Ho in the equation above). In practice, this threshold is defined as a fraction of the lower of the two maxima concerned. Typical values of this fraction are defined between 90% and 100%. For the last of these values, the heuristic is deactivated, because it is impossible to have the minimum of a line (here, the vertical cut) strictly higher than one of its point (here, the smallest of the considered maxima).

3.3 Head candidates classification

Given a list of head candidates, we define here the process of classifying those points into two classes, *person* or *non-person*, by the use of a linear classifier.
To discriminate these two classes, we select for each head candidate some features on a surrounding square window. For each feature, a vector of characteristics is extracted and appended to the others to form the whole feature vector, used as input space for the classification. As linear classifier, we propose the use of a Support Vector Machine with a Gaussian kernel.

3.3.1 Feature extraction

Given a square window surrounding a head candidate point on a depth image, we aim at defining a feature vector to discriminate people from other objects or noise.
Firstly a person is taller than an object or than a bike, so we chose to record the height of the point. Secondly the shape of a person is highly discriminant, even with additional accessories, and is typically composed of a circular head surrounded by two shoulders and then a deep space. To characterise such a shape, we chose to compute the normalised norms of the gradients on the window for a discrete set of directions [3]. Thirdly, to capture the progressive height as well as the progressive slope around a point, we chose to record the concentric squares mean.

Height

The height feature consists of measuring the height of the head candidate. Since the values of the pixels correspond to a depth in a scale from 0 to 65535, they need to be rectified to be converted in millimetres, as defined in Section 2.3.3.

Angle

The angle feature represents the angle (in radian) between the radial segment [observed point - camera], and the vertical axis passing through the camera. This is illustrated in Figure 2.4 as the angle between the segments H_{camera} and $Depth$.

Histogram of Oriented Gradients

This feature is designed to capture the relative orientation of the slope on a square window. We use a larger Gaussian filter before computing the gradients, more precisely 7×7 pixels, such that the resulting surface is smoother and no disturbance deforms the gradients. This is illustrated in Figure 3.11 for the whole convolved image and in Figure 3.3.1 for the map of gradients on a 51×51 window centered at the maximum on the head of the person. (Contours have also been added to help visualisation of the progressive slope).

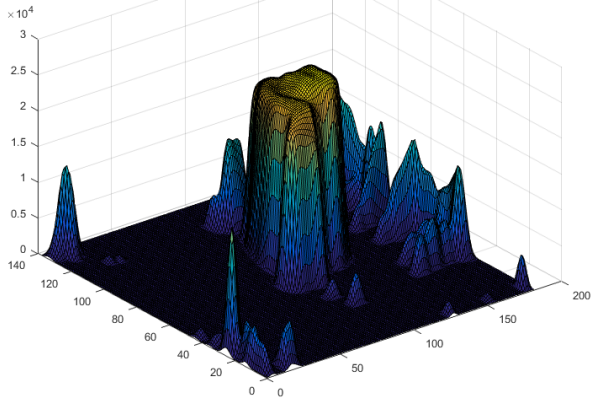

Figure 3.11: An image filtered by a 7x7 Gaussian filter

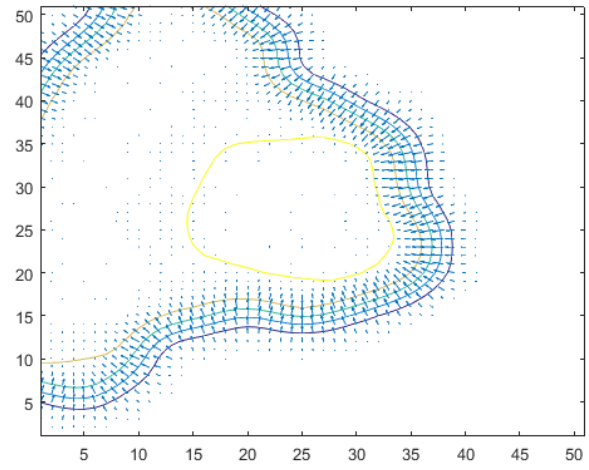

Figure 3.12: A plot of gradients as arrows on a 51×51 window centered on a maximum

For one histogram of oriented gradients (HOG) [3] of size N, the spectrum of every orientation is divided into N bins of $\frac{2\pi}{N}$ radians in which the norms of the corresponding oriented gradients are added. We designed a threshold named $limit$[1] to bound very high gradients value, to avoid that the large gradients induced by a sharp edge dominate the gradients caused by a human depth profile.

Formally, let $f(x, y)$ denote the surface on the filtered depth image, $\vec{\nabla} f(x, y)$ the gradients of the surface and $dir(\vec{v})$ the orientation of vector $\vec{v}$ (in radian), then we define the HOG as follow:

$$HOG[i] = \sum_{\forall x,y \mid dir(\vec{\nabla} f(x,y)) \in [i\frac{2\pi}{N} \; ; \; (i+1)\frac{2\pi}{N}[} min(\|\vec{\nabla} f(x, y)\|, \; limit)$$

[1]This parameter could be subject to a cross-validation procedure to be optimised with respect to the available training data. We have however set it to 100, for simplicity, and because tests have revealed that it has no significant impact on the classification outcome.

The length of a HOG vector being usually a power of two, we propose as length $N = 8$. Eight bins allow a sufficiently rich histogram in term of directional diversity, while keeping the size small enough for computational issue.

Even if one single HOG per window is already able to capture the orientation of the slopes on the window, it does not make any usage of the gradients location. Thereby, to take into account the spatial location of these oriented gradients, we advise to divide the window into a grid of sub-windows and to compute a HOG for each sub-window. The choice of the dimensions of this grid depends of several constraints:

- Since the result must be isotropic, the grid must contain as many columns as rows.
- To have comparable HOGs, the sub-windows must be of the same size.
- Too many sub-windows is not computationally efficient.
- The number of rows and columns must be odd such that the central sub-window is centered on the given head candidate, to optimise the shape study.

Among the choices left by these constraints, we advise a 3×3 grid, resulting so in nine histograms of eight bins, i.e. a vector of 72 numbers. This is the smallest grid fulfilling the constraints, while keeping the resulting vector small enough. Indeed a 5×5 grid would have produced a vector of 200 numbers.

Concentric Squares Mean

The previous feature proved to be quite good at catching the typical orientation of edges on a window depicting a true or false head among the candidates. However, HOGs are not able to measure the absolute height around a point.
To capture the shape as a progression of height, we propose as feature to compute the concentric squares mean (CSM) on a window surrounding a maximum. We define the CSM as a vector of size N where the i_{th} element is the sum of the height of every pixel on the square ring of side $2i + 1$, divided by the number of pixels on this square ring ($8i$).
To compute the sum over these rings efficiently, we suggest to make use of integral imaging [4, 5]. A new representation of the filtered depth image is computed, in which each pixel corresponds to the integral of the image from the origin $(0, 0)$ to this pixel. Let $f(x, y)$ denote the surface on the filtered depth image. The integral image (I2) is define as

$$I2(x, y) = \sum_{\forall x', y' \mid x' \leq x \,\wedge\, y' \leq y} f(x', y')$$

As example, integrating the image over the square $ABCD$ displayed in Figure 3.13 is simply $I2(x_C, y_C) - I2(x_D - 1, y_D) - I2(x_B, y_B - 1) + I2(x_A - 1, y_A - 1)$.

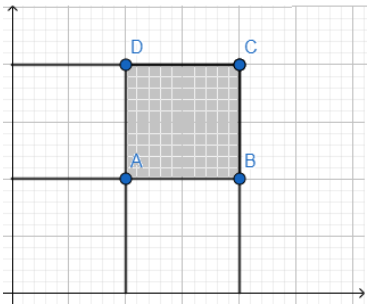

Figure 3.13: Example of a square image integration

Now to compute the CSM vector, let $Square(x_M, y_M, n)$ denote the integral of the filtered depth image over the square of side n centered in a maximum M, and $CSM[i]$ the i_{th} element of the vector:

$$\begin{aligned} CSM[i] = &\quad Square(x_M, y_M, 2i+1) \;-\; CSM[i-1] \\ CSM[0] = &\quad Square(x_M, y_M, 1) \end{aligned}$$

Note that the first element of the CSM vector, $CSM[0]$, actually corresponds to the unrectified height of the head candidate, so we can ignore it since the rectified height is already in the feature vector.

Combination of different features

Since the height, the angle with the camera, the histograms of oriented gradients and the concentric squares mean are designed to represent different features, they can be used all together. Moreover, the CSM and the HOGs represent complementary information for the surface (average height and derivative), so they have better to be used on the same square window of size $n \times n$. The CSM implies that n is odd since the window is centered on the head candidate and the squares are concentric. The HOGs imply that n is divisible by 3 since we divide the window into a 3×3 grid of sub-windows. Therefore, among the different possible window sizes, we choose 51×51 because it is the most suited one to the dimensions of a person body on the depth images.

The whole feature vector is produced by appending the vector of each feature. This is illustrated in Table 3.1 for a 51×51 window.

Height	Angle	9 HOGs of 8 bins	1 CSM of 25 rings
1	1	$9 * 8$	25

Table 3.1: Feature vector, totaling a length of 99

3.3.2 Support Vector Machine Classifier

Support Vector Machine as a hyperplane

This subsection aims to present the linear classifier we chose to predict if a head candidate belongs to the class of person or non-person, based on the feature vector.

Let x denote a vector of real values ($x \in \mathbb{R}^d$) in a space of dimension d, called input space. A linear discriminant function $g : \mathbb{R}^d \to \mathbb{R}$ is a linear function of x defining a decision rule for a binary classification problem:

$$f(x) = sign(g(x))$$

$$g(x) = W^t x + W_0 = \langle W, X \rangle + W_0$$

where $\langle W, x \rangle$ denotes the dot product between W and x. This linear discriminant function can be seen as a separating hyperplane on the input space. The Support Vector Machine algorithm [6] aims to maximise the margin between such plane and the closest examples, called support vectors. Figure 3.14 illustrates a 2-dimensional input space with a plane separating the circles and the crosses. The red examples support the plane because they are on its parallels at a maximal margin distance. The learning phase consists then of finding the most appropriate weights W regarding the Soft Margin Optimisation [7]. In practice we use the Sequential Minimal Optimisation solver which is quite simple but still efficient.

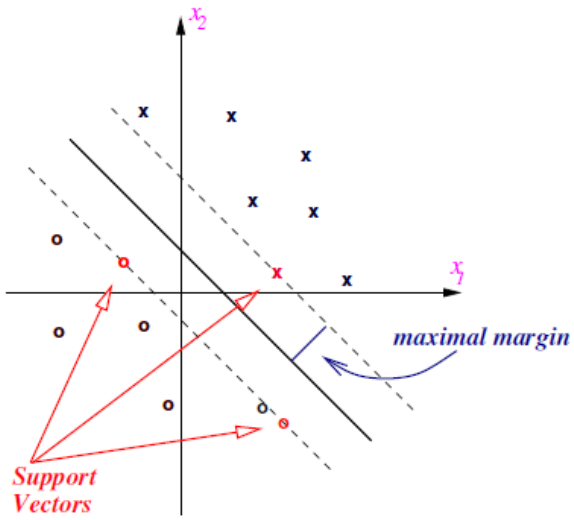

Figure 3.14: Hyperplane separating a 2D input space

Kernel trick

It might very probably occur that the examples are not linearly separable in the input space. To settle this problem down, the SVM algorithm enables to project the examples in a higher-dimensional space where they could be linearly separable, thanks to a (possibly) non-linear mapping ϕ. This is illustrated in Figure 3.15, where the non-linear mapping is $\phi(x) = [z_1\ z_2\ z_3]^t = [x_1^2\ x_2^2\ \sqrt{2}x_1x_2]^t$.
In practice, this mapping does not need to be computed thanks to the kernel trick. A kernel [8] denoted here k is a symmetric function:

$$k(x, x') = \langle \phi(x), \phi(x') \rangle = k(x', x)$$

Thenceforth, any learning algorithm that uses the vectors only via dot products can rely on this mapping to a new feature space by replacing $\langle x, x' \rangle$ by $k(x, x')$. Different types of kernel are possible, each one enabling some hyper-parameters tuning. We chose to use only the Gaussian kernel (also called the Radial Basis Function) because it works relatively well for two-class learning and is widely used in the literature of visual recognition. Its expression is presented here at Equation 3.2.

$$k(x, x') = e^{\frac{-\|x-x'\|^2}{\sigma^2}} \tag{3.2}$$

Another reason that kept us from comparing the Gaussian kernel to some other kernels was the consequent learning time. A single learning phase is already quite slow regarding the number of examples to process. Studying other kernel functions would have required a validation set and so an extra cross-validation. However we recommend to try some linear kernels and the Sigmoid kernel to boost the general performance.

3.4 Conclusion

To perform detection of persons on depth images, we have assumed that the part of the body that is most recognisable on these images is the head, and that a pixel resulting from a point on a head is always among the local maxima. From these assumptions, we propose a two-step detection model.

The first step consists of selecting head candidates from the local maxima on Gaussian filtered depth images. These candidates are then filtered by four heuristics: *Highland*, which ensures that the maxima are surrounded with enough high surface; *Height*, which checks if the height is

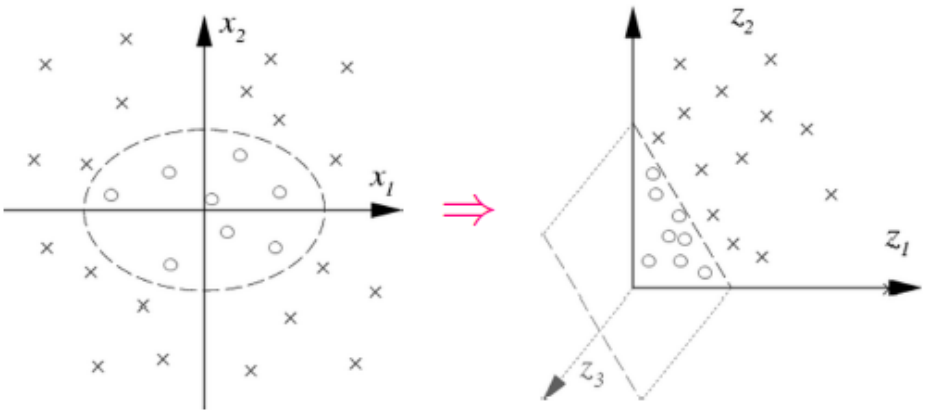

Figure 3.15: Mapping from a 2D feature space to a higher-dimensional space

sufficient; *Distance*, which ensures for every pair of maxima that they are not too close to each other; and *Hollow*, which checks for every pair of maxima if a sufficiently deep hollow separates both maxima.

The second step consists of classifying the remaining head candidates in two classes: *person* and *non-person*. A feature vector is evaluated for each candidate on a surrounding square window to characterise the point. We propose three different features: the height of the point in millimetres, a grid of histograms of oriented gradients and a vector of concentric squares mean. As classifier, we use a Support Vector Machine model with a Gaussian kernel.

Chapter 4

Detection tracking over time

4.1 Introduction

This chapter is concerned with post-processing the detections of people on stand-alone images, using Multi-Object Tracking (MOT) [9, 10, 11] to link together the detections of a person, thereby drawing its trajectory through a given scenario. Tracking approaches are commonly used in traffic monitoring [12] or in video supervision [13]. In the scope of this thesis, tracking detections is a way to isolate the consistent trajectories and so to count the number of people involved through a given sequence of images.

4.2 Graph-based formulation

The MOT problem is defined as building disjoint sets of detections, each set being associated to a physical person evolving through the gate during the scenario. Like Delannay et al. [9], we use graph formalism to specify this data association. Each detection is represented by a node u in a graph and is characterised by a time t_u, a spatial location l_u and an appearance, defined here as the height of the detection in the depth image h_u. Each pair of nodes is connected by a weighted edge representing the dissimilarity between the two nodes. The dissimilarity is defined as the difference in time, space and appearance. The MOT problem is then formulated as follow::

$$\begin{array}{rl} \text{minimise} & \sum_{i=1}^{K} cost(T_i) \\ \text{subject to} & T_i \cap T_j = \emptyset, \forall i \neq j, \\ & \bigcup_{i=1}^{K} T_i = V, \\ & \forall\ i > 0 \text{ and } \forall u, v\ \in\ T_i\ ,\ \ t_u \neq t_v, \end{array}$$

where $cost(T_i)$ denotes the dissimilarity cost within the i-th set and V the set of all nodes in the graph. The cost evaluation within a set is based on the weight of its edges. Let w_{uv} denote the weight of the edge between nodes u and v.

$$cost(T_i) = \sum_{u,v \mid u \neq v} w_{uv}$$

As pointed by Delannay et al., taking into account every edge in each complete sub-graph T_i leads to a computationally over-elaborated problem. Therefore, we impose directed edges between nodes of strictly increasing time-stamp, as illustrated in Figure 4.1. In practice, this is done by assigning an infinite weight to edges that are directed to a head candidate detected in the past or in the same image. More formally, we define the cost function evaluating the weight of the edge between nodes u and v as follow:

$$w_{uv} = \begin{cases} \alpha.\Delta_T^2 + \beta.\Delta_T.\Delta_D + \gamma.\Delta_H + \delta.f(V) & \text{iff } t_u < t_v \\ \infty & \text{Otherwise} \end{cases} \tag{4.1}$$

Where

$\Delta_T = t_v - t_u$ is the difference of time between nodes u and v.

$\Delta_D = \sqrt{(x_v - x_u)^2 + (y_v - y_u)^2}$ is the spatial distance between both detections.

$\Delta_H = |H_v - H_u|$ is the difference of height between both detections.

$V = \dfrac{\Delta_D}{\Delta_T}$ is the average speed between nodes u and v.

$f(V) = a\ e^{bV}$ is an exponential of the speed designed to highly increase when the speed is unlikely for a human, with regard to the image scale and the shooting cadence of the camera. Here, a and b are respectively fixed to 0.4 and 0.3, to achieve this goal.

α, β, γ and δ are parameters that will be discussed later on, in Chapter 5.

Thanks to this simplification, the partitioning problem can be solved by finding the shortest paths, either with the k-shortest paths algorithm [14] or with a greedy iterative using the shortest path algorithm of Dijkstra. The latter one is a simple greedy approach, sufficient in the scope of this thesis, since we are primarily interested in analysing the short-term temporal consistency of detections (to support reliable counting of people at a given time instant), without caring about long-term tracking performance. Using a different method that computes the paths simultaneously (as the k-shortest path method) could be interesting in cases for which we want to limit the assignment of distinct people to a same track (identity switches along the track).

Since the first and last node of a trajectory are not necessarily known, we define two virtual nodes, v_{source} and v_{sink}, the first one being fixed in time at the beginning of the scenario and the second one fixed in time at the end of the scenario. The weight of the edges connected to these two nodes is then defined in term of Δ_T only, i.e. $w_{uv} = \alpha.\Delta_T^2$.

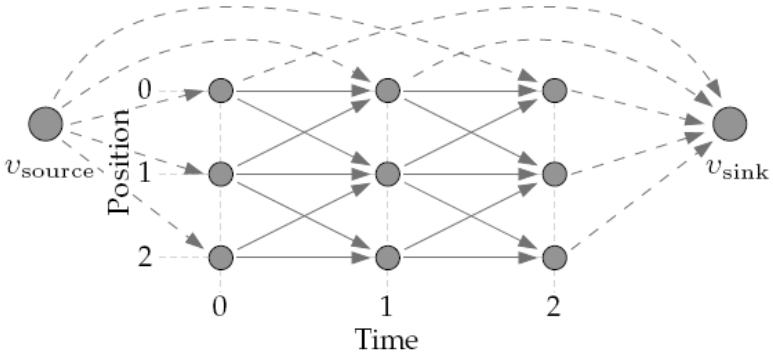

Figure 4.1: Example of a tracking graph, from [15]

The iterative Dijkstra algorithm may return too many paths compared to the real number of people, i.e. creates some partitions of the graph containing false positives. To avoid numbering these wrong paths as people, we define the function $Reject : path \rightarrow \{true, false\}$ rejecting a path if it is probably not characterising the trajectory of a person. The symbols $thres1$, $thres2$ and $thres3$ denote additional parameters that will be discussed later on.

$$Reject(path) = \begin{array}{l} path_{\#nodes} \leq thres1 \vee \\ \frac{path_{\#nodes}}{length_{scenario}} \leq thres2 \vee \\ \frac{path_{cost} - w_{v_{source}n_{start}} - w_{n_{end}v_{sink}}}{path_{\#nodes}} > thres3 \end{array} \tag{4.2}$$

This function rejects a path if one of these three conditions is verified. Firstly if the path is too short in term of nodes, to reject the paths that are too short to represent a person crossing the gate. Secondly if the number of nodes divided by the number of images is too low, to reject low density paths, with multiple time jumps. Thirdly if the cost of the path, minus the cost of the first edge and the cost of the last edge, divided by the number of nodes is too high, i.e. if the average cost by edge is too high, simply because these paths are less likely. In the latest condition, we deduct the cost of the first and last edge from the total cost of the path because it corresponds to the time penalty if the trajectory begins late or ends early in the scenario.

4.3 Conclusion

To track detections through a scenario we use graph formalism to reduce the Multi-Object Tracking problem to the k-Shortest Path problem. A graph is built by associating a node to each detection, using as attributes the time, the spatial location and the appearance of the detection. The beginning and the end of the scenario are respectively associated to a *source* and a *sink*. Nodes are then connected by weighted edges representing their dissimilarity, defined as their difference in time, space and appearance.

The final objective of the tracking approach is to count as accurately as possible the number of people in the gate at a given time. We are therefore more interested in the number of paths than their accuracy. As a result, we propose a greedy iterative shortest path algorithm to select the k shortest paths between the source and the sink.

It may occur that a path returned by the shortest paths algorithm is inconsistent, because it is built on some false detections. To avoid counting these as persons, we rule out the trajectories that are inconsistent, according to their number of nodes, their density of nodes and their average cost by image.

Chapter 5

Validation methodology

5.1 Introduction

The purpose of this chapter is to present the validation of the proposed detector. This is essential since it proves the validity of our results, the consistency of our methods and the re-usability of our work.
First, we specify how we use the datasets received from *Automatic Systems* to train and test our system.
Then, we discuss the parameters used for the selection of the head candidates. We introduce some appropriate metrics to evaluate the sensitivity of the parameters and the impact of the chosen configurations on the method usage.
Next, we introduce a way to process the outputs of the selection of the head candidates as inputs for the classification. A cross-validation is essential whenever a model is to be built and evaluated on the same dataset; hence, we propose a ten-fold cross-validation to fulfil this requirement and we detail how we solve the problem of grouping data into ten sets as an optimisation problem. We then introduce other metrics to evaluate the classification performance, and how to compare models with different class weights.
Finally, we discuss the parameters of the detections tracking algorithm and propose appropriate metrics to measure the performance of the tracking - specifically, in our case, the detection of persons in a gate. This method also needs a cross-validation to prevent the tracking model being trained on the same data with which the classifier was trained.

5.2 Use of datasets and labelling

As presented in Chapter 2, the datasets we received are quite different regarding their size and content. Because the dataset one was received earlier and because its scenarios, recorded experimentally in the lab of *Automatic Systems*, are representative of practical use cases, we decided to use it extensively for the development of our algorithms. On the other hand, the dataset two was recorded in Lille where the light conditions and the floor material are different, so we use it as final test set to prove the validity of our algorithm and its re-usability in a new situation.

To develop our models, we built a ground truth consisting of a hand-made database providing the coordinates of the heads for every image, by recording a point at the centre of each head. Though this operation was achieved for the whole dataset one, it is time consuming. Hence only a part of the dataset two has been annotated.

Thanks to this labelling, it is possible to identify whether a head candidate is a true positive or as a false positive. However, the probability that a point selected as a head candidate is

at the exact same position than the centre of the head registered in the ground truth is very low. To handle this issue, we propose to adopt a validation method that identifies a candidate as true or false positive, based on the Euclidean distance between both points. We also introduce a hyper-parameter named *radius*. Then, a head candidate is validated as a true positive if there exists an entry in the ground-truth such that the distance between the entry's coordinates and the candidate is lower or equal to *radius* and if this point is not yet bound to another closer head candidate. Typical values for this parameter are between 10 and 25 pixels, to be close to the size of a head on the images.

Regarding the dataset one, we do not use every head sample nor every scenario to validate our algorithm. First, the head samples that are located too close to the borders of the images are ignored by our assessment. Indeed, they lack of relevance and are not suited to our algorithm since the features considered for the classification involve a window of 51×51 pixels to compute the HOG or the CSM, detailed in Section 3.3.1. Taking these points into account would then create a confusion and inappropriately reduce the precision of the classification. In practice, we chose to reject the points that are located less than 12 pixels away from the borders of the images, and to extend the images with ground-depth pixels to complete the window surrounding other points. As example, Figure 5.1 shows the ignored borders as a red frame.

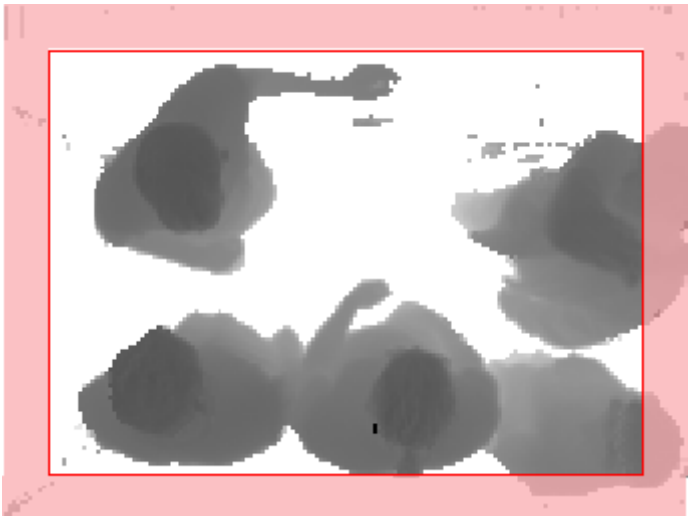

Figure 5.1: Example of an image with a red frame showing ignored borders

Furthermore, we also ignored a number of irrelevant (not representative) scenarios in our assessment. Two of those ignored scenarios are composed of people hiding themselves under a jacket and they do not ever show their head. Four additional scenarios have been ignored during tracking assessment because their trajectories through the gate are not in agreement with the specifications of the problem, i.e. detecting people entering a gate and numbered when the doors are closed. Three of these scenarios start with a person located in the middle of the gate and are too short. The person passes through the gate without waiting for the entrance doors to close and the exit doors remain opened. The fourth scenario is composed of five people who run through the gate, with both entrance and exit doors opened. These inconsistencies would have biased the validation of the tracking.

5.3 Validation of the head candidates selection

As introduced in Chapter 3, the head selection algorithm mainly depends on some parameters that influence the behaviour of the Gaussian filter and the heuristics used. Those parameters are summarised in Table 5.1.

Gaussian filter		Highland			Height	Hollow	Distance
Size N	Standandard deviation σ	Window Size	Cut	Required Surface	Height	Hollow	Distance

Table 5.1: Summary of the parameters involved in the head candidates selection

In total, there are eight parameters to set for head candidate selection. In our experimental validation, we analyse how the values of those parameters impact the head selection. Here, we discuss the metrics that are relevant to interpret the head candidate selection results, and point out the fact that the objectives (and associated performance metrics) are different if the method is destined to be used as a stand-alone one or as a preprocessing one.

The metric generally used in the literature to evaluate the number of false detections is the False Positive Rate (FPR), which is computed as the ratio between the False Positives (FP) and the Negatives[1] (N). This is however inappropriate here because N is very high here. This issue leads us to propose another metric to evaluate the false detections: the *False Head Candidates Rate*, as a ratio between the false head candidates (FHC) and the true head candidates (THC) that are selected by the algorithm. Formally:

$$FHCR = \frac{FHC}{THC}$$

On the other hand, we keep the usual metric used to measure the true detections, i.e. the Detection Rate (DR), as the ratio between the true head candidates and the number of the Positives[2] (P). Formally:

$$DR = \frac{THC}{P}$$

Sensitivity of the heuristics parameters

We use the previous metrics to represent the performance of the head candidates selection on a plot, in a quite similar way than a Receiver Operating Characteristic (ROC) Curve, except that we replaced the FPR by the FHCR.

The variation of the parameters value produces different configurations of the head candidates selection, for which more or less candidates are selected. Each configuration will be represented as a point on the plot. Our experimental validation then aims at finding the best configuration regarding the intended usage of the algorithm.

The Pareto curve and an example of the sensitivity of the heuristics parameters are illustrated in Figure 5.2. This graph is built by giving to each parameter a value among all its typical ones. The Pareto curve is composed of Pareto optimums. These are the best combinations, i.e. the ones that have the highest detection rate and the lowest false positive rate in the first dataset, without being dominated by any other combination. The coloured lines represents the variation of one and only one parameter, as indicated in the legend.

The *Distance* parameter and the *Hollow* parameter vary importantly both the detection rate and the number of False Positives, creating 'plateaus' separated by abrupt drops in the top of the graph. The other parameters vary the results more locally, except the *Window Size* parameter that needs to be used in coordination with the appropriate values of *Cut* and *Required Surface*.

[1]Negatives: In the scope of the head candidates selection, Negatives can be either described as the number of pixels not associated to a person in the ground-truth (all except a few ones), or by the number of local maxima rejected by the filtering heuristics.

[2]Positives: In the scope of the head candidates selection, Positives refers to the number of people identified in the ground-truth.

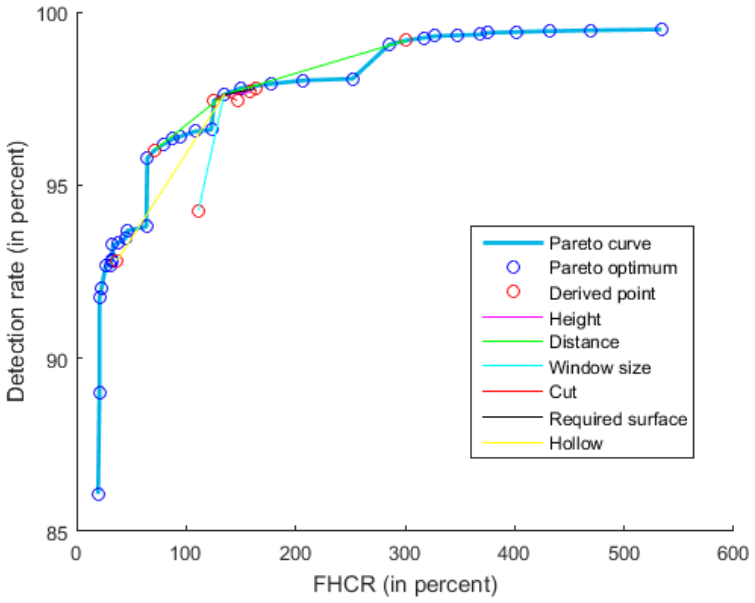

Figure 5.2: Pareto curve for the head candidate selection and heuristics parameters sensitivity

As a preprocessing method

Whenever this algorithm is intended to be used as a preprocessing technique, the selected head candidates are sent to a classifier. The objectives of the method is thus to get the more suitable configuration for the classification. We point out two decisive criteria:

- There should be as less *False Negatives* (people not detected) as possible. Since the number of people (number of *Positives*) on the images is fixed, it is the same as having the more *True Positives* as possible (people correctly detected). Indeed, if the heuristics rule them out, the people can not be detected by the classification.
- The number of *False Positives* should be as close as possible to the number of *True Positives*. Indeed, the difference between these numbers will have an impact on the classifier training. Having the same proportion of training examples between both classes allows the classifier to give the same importance to either of the classes.

As a stand-alone method

The head candidates selection algorithm can be used in a stand-alone way. Indeed it is able to detect people among the maxima of a depth image and to return their location. If the method is used in such a way, the appropriate configuration would probably be another one than for the preprocessing purpose. Indeed, the previously proposed criteria are not relevant anymore.

Here, we advise to use other metrics existing in the common literature, evaluating the performance for similar problems: the *F-score*, the *precision* and the *recall*. The *precision* refers to the number of correct detections among candidates. The *recall* is equivalent to the detection rate introduced before. The *F-score* is then usually used to measure the test accuracy of binary classifiers, while taking into account both the *precision* and the *recall*.

$$\begin{aligned} \text{Precision} &= \frac{True\ Positives}{True\ Positives + False\ Positives} \\ \text{Recall} &= \frac{True\ Positives}{True\ Positives + False\ Negatives} = DR \\ \text{F-score} &= (1 + \beta^2) * \frac{precision * recall}{\beta^2 precision + recall} \quad \beta \in \mathbb{R} \end{aligned}$$

We advise the maximisation of the F_1 score (with β=1) to get a maximum harmonic mean between the precision and the recall. It can however be useful, according to the situation, to promote one of these two values, by manipulating the β parameter of the maximised F-score : if it is greater than 1, it gives more weight to the recall and less to the precision and vice-versa in the opposite case.

5.4 Validation of the head candidates classification

Whenever the head candidates selection is used as a preprocessing technique, the returned points can be identified as true and false candidates, based on our ground-truth. Hence, we propose to label these examples among two classes: *person* and *non-person*, such that they can be used for the supervised learning of our classifier, the Support Vector Machine.
We present here after a possible cross-validation to train and test the model on the same dataset, and then we discuss a performance analysis.

5.4.1 Cross-validation over the dataset

The classifier receives examples from the head candidate selection that are labelled to the classes 'people' or 'non-people'. We propose a ten-fold Cross-Validation (CV), such that a model can be trained and tested over the same dataset. In the case of a ten-fold CV, the data are split into ten different sets and the classifier is built ten times, each time taking a different training set and test set. In that way, we can compute the error and the success rates on different training and test sets and thereby estimate the mean and the variance of the performance of the model. Importantly, in each fold, data that are used to train a model do not include any of the tested data.

Here, since all images in a scenario are reasonably similar, we must ensure while splitting the data that the examples used in the test set and the examples used in the training set do not belong to a same scenario. This last characteristic is essential since the test set is aiming at estimating how the model reacts if the examples are unseen.

This point implicates to group all the examples from a same scenario into the same set, but we still have to ensure that each set contains approximately the same number of examples. Therefore, we have to solve an optimisation problem that consists of grouping scenarios of examples to form ten sets whose sizes are as close as possible. It is worth mentioning here that we chose to form ten sets balanced in term of number of examples and not in term of number of scenarios, since the size of a scenario is very variable and would have biased the comparison. Furthermore, the sets are not formed to be equivalent in term of 'people' and 'non-people' examples. We however assume that they are uniformly distributed among the examples.
Strictly speaking, the problem of splitting the data in ten sets that are as similar as possible is not easy to solve, indeed it can be reduced to known NP-hard problems such as the Cutting Stock Problem (CSP) [16] and the Steel Mill Slab Problem (SMSP) [17, 18]. The reduction to the SMSP is described in Appendices, Section 9.2.

However, for our purpose, there is no reason to have the same number of samples in each set (this is especially true since the diversity/representativeness of frames in a set is more important than the number of frames regarding the quality of the trained model). Hence, we can tolerate differences in the sizes of the ten sets, and we solve this optimisation problem by a simple greedy solution. First, we form a list where the scenarios are sorted in terms of number of examples, in descending order. Then, as long as there are scenarios in this list, we take the first one and put all its contained examples in the emptiest set among the 10. This algorithm does not give a perfect solution, but a sufficiently good one, as presented in Table 5.2 for the first dataset.

Set	Number of scenarios	Number of examples
1	1	854
2	6	862
3	8	864
4	8	866
5	8	866
6	8	869
7	8	877
8	8	877
9	8	878
10	7	885

Table 5.2: Distribution of the scenarios of the first dataset in ten sets

5.4.2 Performance of the classifier regarding class weighting

As stated in the validation of the head candidates selection, one objective of this selection is to ensure that the amount of examples returned for each of the two classes, *person* and *non-person*, is rightly balanced. In that case, the classifier gives an equally weighted importance between the classification of the *person* and of the *non-person* examples. However it may occur that one of the classes is defined by the client to be more important than the other. By example if missing a person is better than detecting one more, then the class *non-person* is more important than the class *person*. To enable the adaptability of our algorithm to such a balance, we produced models with different weighting balance between the classes by replicating some of the examples of the heaviest class. Each model trades-off differently the detection and false positive rates. Performance is thus depicted by using a Receiver Operating Characteristic (ROC) curve.

Models are represented on a plot, in which the horizontal axis represents the False Positive Rate (FPR) and the vertical axis represents the Detection Rate (DR). In the scope of the head candidates classification, the FPR corresponds to the ratio between the number of False Positives (FP), i.e. examples classified as *person* while being labelled as *non-person*, and the number of Negatives (N), i.e. examples labelled as *non-person*.

$$FPR = \frac{FP}{N}$$

Similarly, the DR corresponds to the ratio between the number of True Positives (TP), i.e. examples classified as *person* and labelled as *person*, and the number of Positives (P), i.e. examples labelled as *person*.

$$DR = \frac{TP}{P}$$

Whenever these models must be ranked according to a single criterion, the Area Under the ROC Curve [19] can be used.

5.5 Validation of the detections tracking

Tracking detections is commonly used to evaluate trajectories of dynamic objects, based on detections, and is usually evaluated using the Multi-Object-Tracking-Analysis (MOTA) [20] metric. However, while this procedure is perfectly adapted to measure the precision of the trajectories, this particular feature is beyond the aim of the tracking used here. Indeed, as stated in Chapter 1 (Introduction), the objective is to correctly count the number of people in a gate and so the number of consistent trajectories, even if these paths are swapped or corrupted over a few nodes. We propose thus three distinct evaluation metrics: the percentage of scenarios

for which the number of people is exactly correctly estimated, the percentage of scenarios for which we over-count the people, and the percentage of scenarios for which we under-evaluate the number of people.

Thanks to this evaluation metric, we design a method to tune the parameters α, β, γ, δ, $thres1$, $thres2$ and $thres3$. Those are introduced in Equations 4.1 and 4.2 to specify the cost function for edge weighting and the rejection function for improbable paths. We determine the best values of the parameters under the following assumptions:

- α should be relatively high if there are few missed detections from the SVM. In this way the time jumps are more penalised. On the other hand, if a classifier model is used such as it misses many detections while it produces less false positive, then α should be much smaller.
- β should be relatively high since the people usually do not move a lot in the considered gates.
- γ should be relatively small since the noises can create a difference of height on a head, from an image to another.
- δ should be relatively small since either the speed is consistent with a human speed and does not have to increase the cost, or the speed is not consistent and the exponential highly increases the cost, regardless of its coefficient.

The thresholds parameters of the rejection function need to be determined without such information since their values highly depend of the comportment of the algorithm and, so, of the cost function. Given this information and based on some practical dependencies (detailed later in Chapter 6), we design an interval for each parameter, as presented in Table 6.4.

A grid search is combined with a cross-validation to determine the value of these parameters. Indeed, the tracking should be validated on the first dataset, which is completely labelled and where more information is available. However, the training of the SVM classifier for the previous step is also achieved on the dataset one, so the performance of the detections over known examples would have been biased and would not reflect a real-life situation for the tracking. Thereby, we propose a five-fold cross-validation to train a SVM model successively on four fifth of the data and then to apply the tracking on the last fifth of the data based on the detections of the model. The optimisation method presented in section 5.4.1 is necessary here to group the scenarios into five different folds, as balanced as possible in term of examples. In this way each scenario is tested only once by the tracking method.

5.6 Conclusion

To develop and evaluate the head candidates selection, the classification and the tracking algorithms, we propose appropriate metrics in each case allowing to compare different models or different configurations, regarding to the chose parameters values. Assessing these algorithms under a single criterion (by example the accuracy) is not suitable in the case of people detection because it doesn't allow to balance the two types of errors: undetected people versus additional false detections. The metrics we propose are fit to highlight such balance.

In this section, we also introduce a cross-validation method for the supervised learning of the classifier and of the tracking. The main problem is that (different) images from a given scenario are too similar to be used in training and testing. Hence, we propose to achieve a N-fold

cross-validation by grouping the scenarios into N sets as comparable as possible.

Last but not least, we detail how we use the data received from Automatic Systems. A hand-made ground-truth is built to provide the locations of person's head over the whole first dataset. This labelling is a key preprocessing step because it allows an accurate supervised learning of our algorithms.

Chapter 6

Results and discussion

6.1 Introduction

This chapter presents the results of our validation and discusses the effectiveness of our algorithms. We decided to distinguish six categories of scenarios to detail accurately the performance in each situation. This categorisation will be used for both the classification and the tracking performance analysis.

The selection of the head candidates is validated by exploring the possible configurations through the tuning of the parameters. As introduced in our Validation Methodology, we compare these configurations in relation to two variables, the DR and the FHC, and we illustrate this comparison by mean of a Pareto curve. We select the configuration that is the most suitable to a preprocessing step algorithm and detail the values chosen for the parameters.

The report of the results of the classification of the head candidates begins with a discussion of the validity if the discrimination features. These features are validated by proving that they are significantly different for a person and a non-person, in such a way that the classifier can use them to discriminate these two classes. To this end, we use boxplots to show the distribution of the possible values of these features, measured over all the input points, for the two classes of interest.
Then, we pursue the discussion of the results with the performance of the Support Vector Machine on stand-alone images, first for each category of scenario, then for each feature selection. The average performance and its variation are finally presented by the mean of a ROC curve.

Finally, the detections tracking algorithm is validated by counting for each category the number of scenarios for which the involved people are correctly counted, over-counted or under-counted. We also present the chosen parameter values for the cost and rejection functions.

6.2 Data characterisation

For this chapter, it is particularly interesting to express the performance for different types of situation. We therefore group the scenarios of dataset one into five categories, as presented in Table 6.1. The number of candidates express how many head candidates are selected by the first-step algorithm in the corresponding scenarios.

The '*PRM*' category contains all the scenarios during which a wheelchair appears, no matter if there is another person or not. The '*Objects*' category contains all the situations in which an object (bike, bag, hat, suitcase, etc.) appears and either deforms the shape of a real person or can be confused with a person because of its head-like shape. The scenarios of '*Unusual postures*' category present at least one person taking a very unusual posture (crawling, bending, jumping,

Categories	1 person	2-5 people	PRM	Objects	Unusual postures
Number of scenarios	11	14	13	17	15
Number of candidates	775	2912	1407	2028	1576

Table 6.1: Distribution of scenarios of dataset one into categories according to their type

carrying another person, etc.). The two last categories contain all the scenarios left, according to the number of people involved.

6.3 Head candidates selection

The head candidates selection among the local maxima of depth images is done by filtering these points by the way of four heuristics. Each one is adjustable by some parameters, enabling the choice between a high detection rate with many false detections and a lower detection rate with less false detections. As presented in Section 5.3, the requirements to use this algorithm as a preprocessing technique are first having the less *False Negatives* as possible, and secondly having a number of *False Positives* close to the number of people (for the first set, between 4000 and 6000).

All the possible configurations produced by the variation of the parameters value appear on Figure 6.1. From these points, we produce the Pareto curve, illustrated on Figure 6.2, composed of Pareto optimums, that are the points that are not dominated by any other point. As a remainder, the *False Head Candidates Rate* (FHCR) metric, defined in section 5.3, represents the fraction between the number of false head candidates and the number of true head candidates that are selected by the selection algorithm.

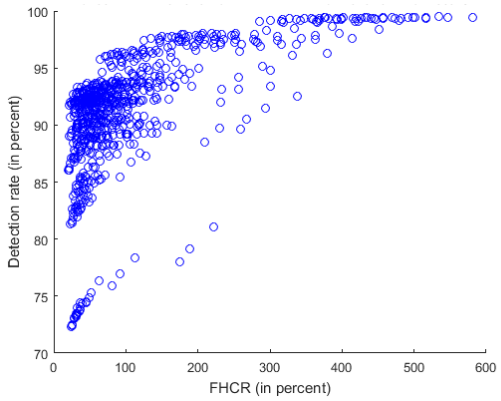

Figure 6.1: Scatter plot of all the results produced by different values of heuristics parameters

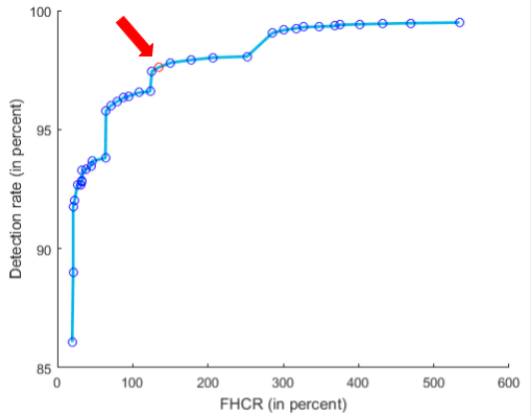

Figure 6.2: Pareto curve and selected point from all the results produced by different values of the heuristics parameters

The chosen configuration is marked in red on this curve. The parameters value for this point is presented in Table 6.2. This choice respects the requirements because the FHCR is not too high while the detection rate is still quite good. As summarised in Table 6.6, this configuration corresponds to a *Detection Rate* of 97.61 % (4046 *True Positives* versus 99 *False Negatives*) and a *False Head Candidate Rate* of 134.79 % (5587 *False Positives*).

As an example, we apply this set of parameters on an image of the first dataset, representing a person with a bike. The gradual filtering of the maxima is illustrated in Figure 6.3, to produce

Heuristics	**Highland**			**Height**	**Hollow**	**Distance**
Parameters	Window Size	Cut	Required Surface	Height	Hollow	Distance
Typical Intervals	10-20	0.6-0.8	0.4-0.6	900-1100	0.9-1	10-20
Chosen Values	10	0.7	0.6	1100	1	15

Table 6.2: Summary of the heuristics and their parameters for a preprocessing technique

at the end two head candidates, one true (on the head) and one false (on the arm).

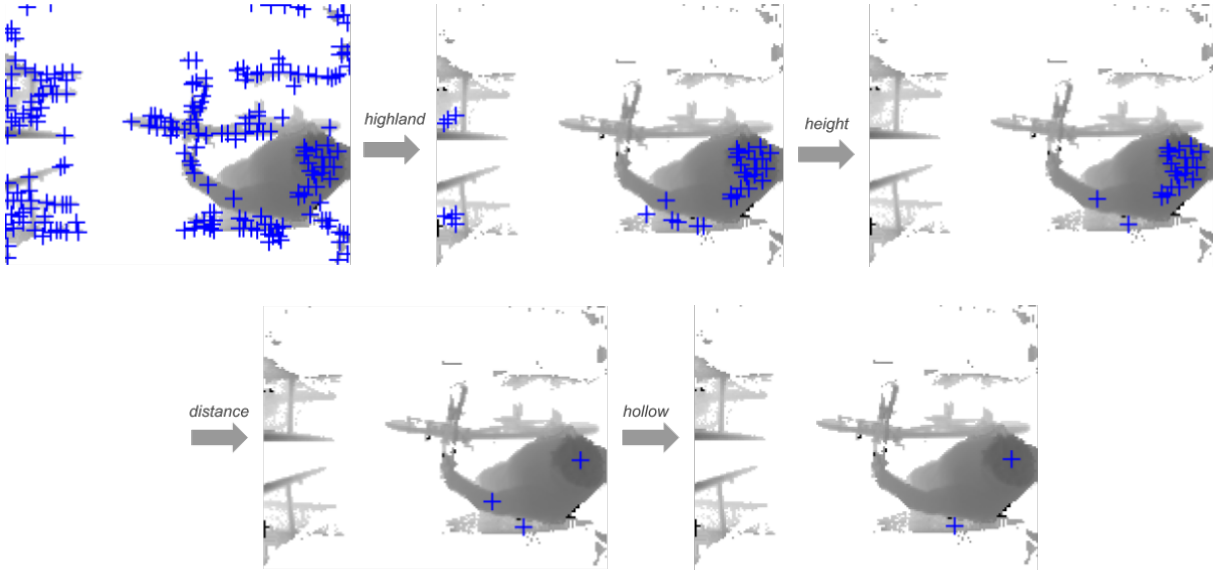

Figure 6.3: Example of the head candidates gradually selected by the heuristics for a Foreground Depth Image

The Pareto curve expressed in Figure 6.2 shows very well that this approach can be used as a preprocessing technique as well as a stand-alone technique. The chosen configuration, for the first alternative, gives good quality input points to the classifier, respecting a balance between true and false examples. On the other hand a chosen configuration with less False Head Candidates would have produced on good F_1 score if used in stand-alone. According to the utilisation of the system, the most suited configuration can be selected, such that the classifier is fed with more points in the class we want the system to perform the best: either to recognise the most people as possible, or to avoid detecting wrongly 'non-people'.

6.4 Head candidates classification

In this section, we detail the behaviour of the Support Vector Machine classifier, based on the chosen configuration of the head candidates selection algorithm presented in the last section. First, we analyse the significance of each feature selection to characterise a given head candidate, by the mean of statistical boxplot. Then, we present the performance of the classification for different categories of scenario and the generated variance. Finally, we present the performance of the classification for each feature selection.

6.4.1 Feature significance analysis

Height

The distribution of the height (in millimetres) for the training examples of both classes is illustrated in Figure 6.4 and 6.5. Note that although the height is used in millimetres, disturbance on the head of people significantly widen the possible interval, even beyond 2 metres high. This plot

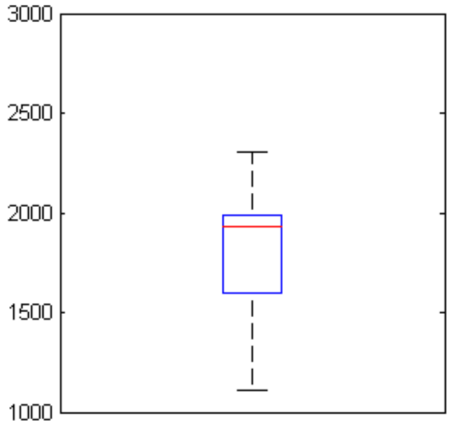

Figure 6.4: Distribution of the height for the examples of the *person* class

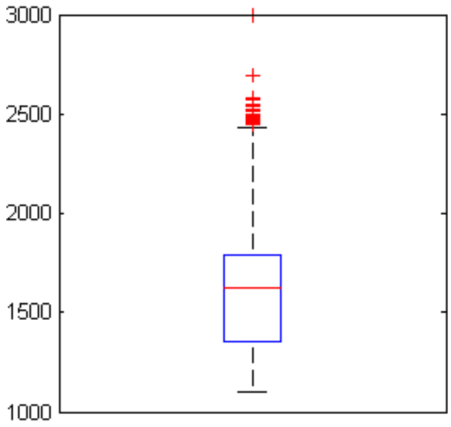

Figure 6.5: Distribution of the height for the examples of the *non-person* class

however shows that the distribution of the two classes is quite different. Indeed, for the class *person*, the median is higher and the interquartile range (IQR) is smaller and clearly above the one of the class *non-person*. Moreover, any candidate characterised by a height above 2300 mm can certainly by classified as non-person.

Histogram of oriented gradients

Figures 6.6 and 6.7 present the boxplots of the histograms for respectively the training examples of the class *person* and *non-person*. The values of the bins standing for a true head candidate point appear to significantly differ from the ones derived from a false head candidate: the central sub-window has an equal weak distribution of all the gradients and the other sub-windows have a high distribution in some bins (depending of their positions) versus a very low in others, indicating homogeneous slopes in the windows. On the contrary, the histograms of the false head candidate points are more equally distributed, indicating a rather random environment.

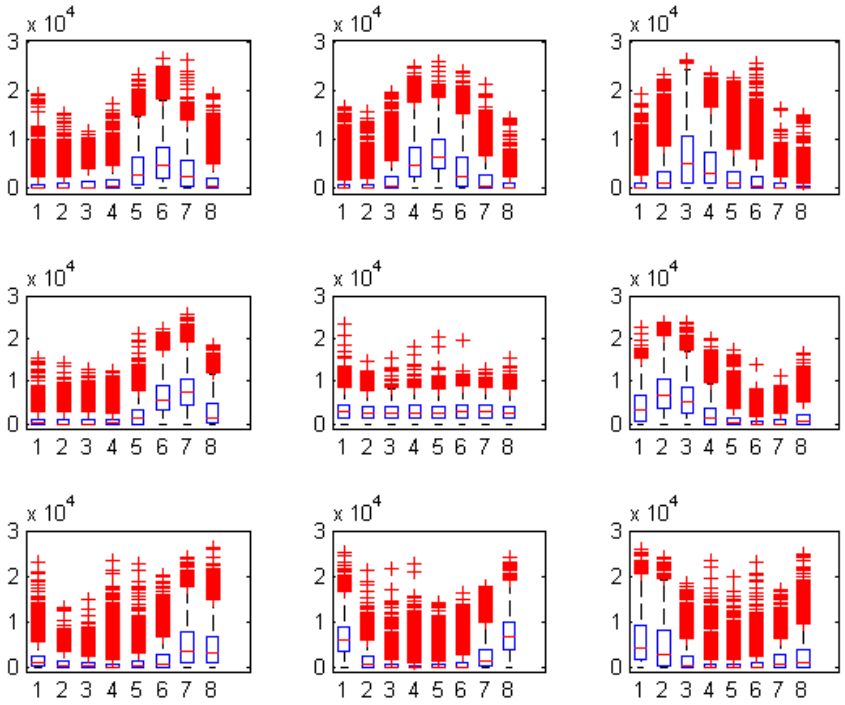

Figure 6.6: Boxplots of the Histograms of Oriented Gradients for examples of the class *person*

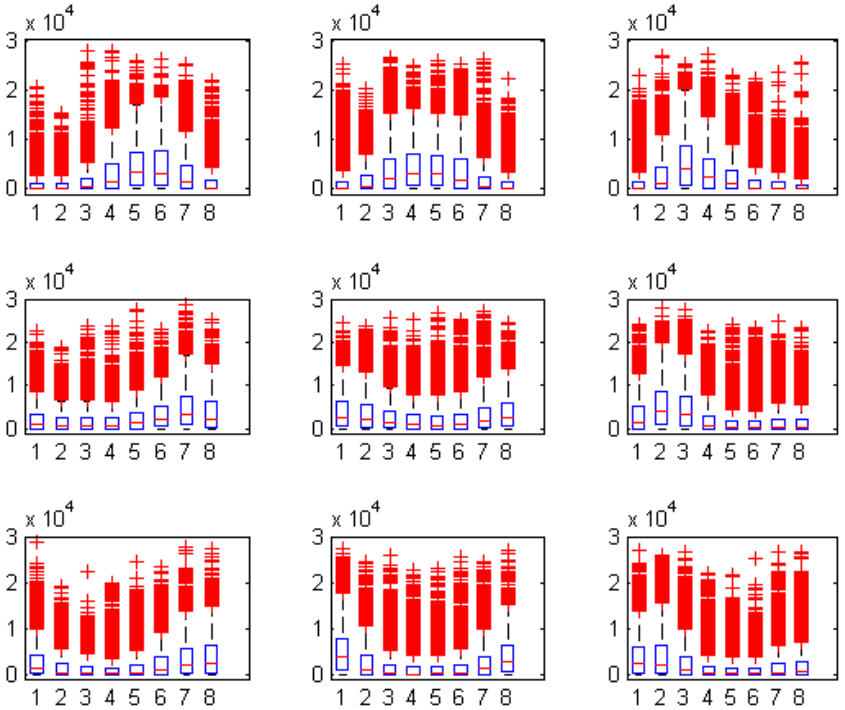

Figure 6.7: Boxplots of the Histograms of Oriented Gradients for examples of the class *non − person*

Concentric square mean

The distribution of the CSM variables, according to the examples of the classes *person* and *non-person*, is illustrated in Figure 6.8 and 6.9.
The twenty-five values of the abscissa represent the means of the twenty-five rings around each

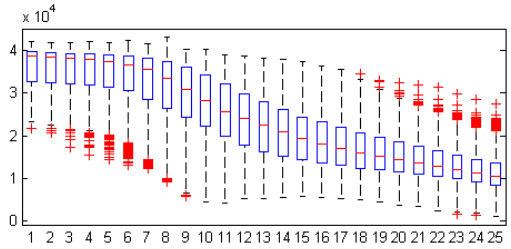

Figure 6.8: Boxplots of the CSM for examples of the class *person*

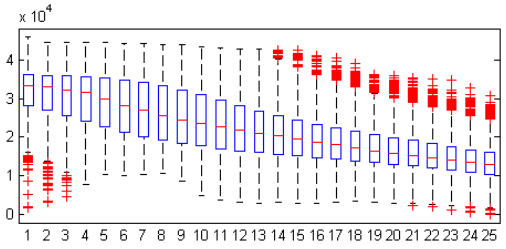

Figure 6.9: Boxplots of the CSM for examples of the class $non-person$

point on a window of 51x51 pixels. This feature can be supposed significant since the two boxplots are very different. Indeed, on one hand, the 'people' plot exhibits the shape of the main elements of the upper part of a body : it remains steady in the first rings for the head, then the values of the next rings drop suddenly for the passage from the head to the shoulders and finally it remains roughly steady for the shoulders. The 'non-people' plot, on the other hand, is more linear, nearly a straight line. Furthermore, it begins lower and ends higher than the 'people' graph.

6.4.2 Performance through categories and folds

The performance of a classifier is quite dependent on the kind of scenario over which it is tested. Thanks to the data characterisation introduced at the beginning of this chapter (Section 6.2), we study the results of the classifiers through each category, with equivalent class weight. Table 6.3 presents the results of the SVM classifier according to these categories of scenarios. It considers an operating point that gives the same penalty to both kinds of classification errors.

Category	1 person	2-5 people	PRM	Objects	Unusual postures
Detection Rate	95.62 %	90.48 %	86.49 %	90.92 %	75.17 %
False Positive Rate	14.63 %	9.71 %	6.08 %	7.10 %	5.47 %

Table 6.3: Results for each category of scenario

The rates expressed in the tab are computed according to these formula, and so weighted equivalently between each candidate no matter the size of its scenario. The Detection Rate represents the percentage of head candidates from the *person* class correctly classified. The False Positive Rate represents the percentage of head candidates from the *non-person* class wrongly classified.

$$DR = \frac{\sum_{scenario \in category} TP_{scenario}}{\sum_{scenario \in category} P_{scenario}}$$

$$FPR = \frac{\sum_{scenario \in category} FP_{scenario}}{\sum_{scenario \in category} N_{scenario}}$$

Where

$TP_{scenario}$ is the number of true positives in *scenario*, i.e. the number of candidates in *scenario* classified as *person* while being effectively labelled as *person*.

$FP_{scenario}$ is the number of false positives in *scenario*, i.e. candidates in *scenario* classified as *person* while being labelled as *non-person*.

$P_{scenario}$ is the number of positives, i.e. the number of candidates in *scenario* labelled as *person*.

$N_{scenario}$ is the number of negatives, i.e. the number of candidates in *scenario* labelled as *non-person*.

We highlight several points about the results presented in this tab. Generally, a high detection rate is carrying a higher false positive rate and contrariwise a low FPR is also carrying a lower DR, following the distribution on a typical ROC curve.

- The '1 person' category, supposedly the most common one where only one person is present in the gate, has a very good detection rate. That is expected since a person standing normally is very recognisable. However, its false positive rate is distinctly higher than the other categories. This can be explained by the fact that our two-step algorithm is trained uniformly on the five categories and weighted by example (not by scenario). Thereby, the heuristics parameters and the SVM variables are optimised to detect additional people even when they are less recognisable. A more restrictive classification would have decreased the FPR for the '1 person' category while decreasing also the DR of the '2-5 people' category.
- The '2-5 people' category presents a slightly lower detection rate. Some of the few missed detections are caused by two people walking very closely from each other, with their head side by side.
- The 'PRM' category has a quite low detection rate, compared to the other categories. A first reason to these missed detections are situations where a person is walking while pushing a wheelchair and is leaned over the wheelchair, hiding thus the person seated on it. A second reason to this low DR is a too small square window from which the HOGs and CSM features are extracted. Indeed the algorithm is trained over every category and so the optimal window side is fixed to 51×51 pixels. This size is fine for the walking people and for their carried objects, but not for a wheelchair since it is wider than 51 pixels. With a larger window, the shape of a wheelchair could be easily recognisable, but at the expense of the DR of other categories.
- The 'Objects' category has a good false positives rate, considering the fact that the scenarios of this category contain various objects of unexpected shapes. Some objects are nevertheless wrongly classified as people. By example, a large backpack is approximately shaped like a person and is moving at the same height. A soccer ball placed on the knees of a person in wheelchair also corresponds to a person (even us were hesitating how to label it). On the other hand, some people are missed because of some disturbing objects, like a jacket put in the air above head height.
- The 'Unusual postures' category has, by far, the lowest detection rate. That is an expected result, due to the principle of a classifier : indeed, it is trained with examples and it aims to link new situations with the ones that it has already seen. When the posture is unusual, the classifier has trouble to link the head and its direct environment to the known examples. On the contrary however, the rate of false positives is very low.

It is now clear that the performance of a classifier is quite different dependently of the situation. However, since the type of scenario is unpredictable for now, the same algorithm is used for any

situation. We present thus the grouped performance in Figure 6.10, by the way of a ROC curve, on which we added boxplots for different class weights to show the variation of the performance for the DR (on the left) and for the FPR (on the right). The variation is studied by the mean of a ten-fold cross-validation, as presented in Section 5.4.1 (Cross-validation). Since the scenarios are uniformly assigned to the folds, the categories are likewise so the variation corresponds to the dissimilarity between the categories.
We advise to use the classifier with both classes equally weighted, corresponding to the point that is in average the closest to the perfect classification point $(0, 100)$. This performance, summarised in Table 6.7, is suited to a direct usage for people detection or for further approaches such as tracking, to count people.

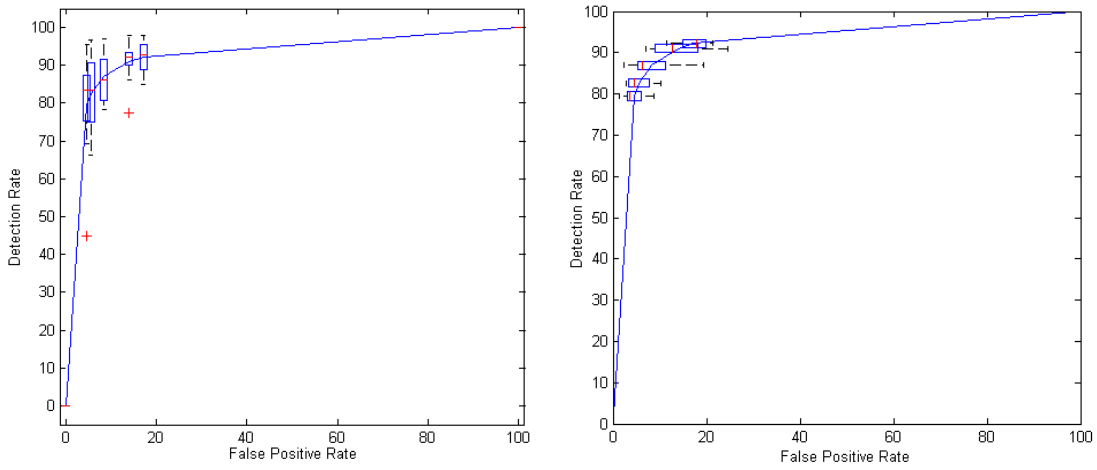

Figure 6.10: ROC curve of the candidates classification and boxplots showing the variance for the DR (on the left) and for the FPR (on the right)

6.4.3 Classification performance for each feature selection

The classification of the head candidates is entirely based on a vector of features, extracted from a square window around the point. To show the contribution of each feature selection, we study here the performance of different SVM models built with each selection, as presented in Figure 6.11. We vary the class weight to produce different ratios between DR and FPR and so to draw the ROC curve, as explained in the section 5.4.2. For each model, we report the average results of the ten-fold CV.

Firstly, the classifiers based only on the height of an input in millimetres, represented in green, are slightly better than the random classifier. The best balance is able the detect a person around 70% of the time. However, its false positive rate is almost of 40%.

Secondly, the light blue curve shows the performance of the classification based only on a single Histogram of Oriented Gradients of eight bins and the angle between the link camera-head candidate and the vertical. No matter the weight of the classes, these models are only satisfactory at classifying correctly true positives. On the opposite, the true negatives are often misclassified, producing so a high FPR. Likewise this curve, the purple one shows the performance of models classifying candidates based on nine HOGs of eight bins built on a 3×3 grid of sub-windows (and the angle again). While this is the same feature selection, the location of the gradients brings additional information, resulting in much better performance, both regarding the DR and the FPR.

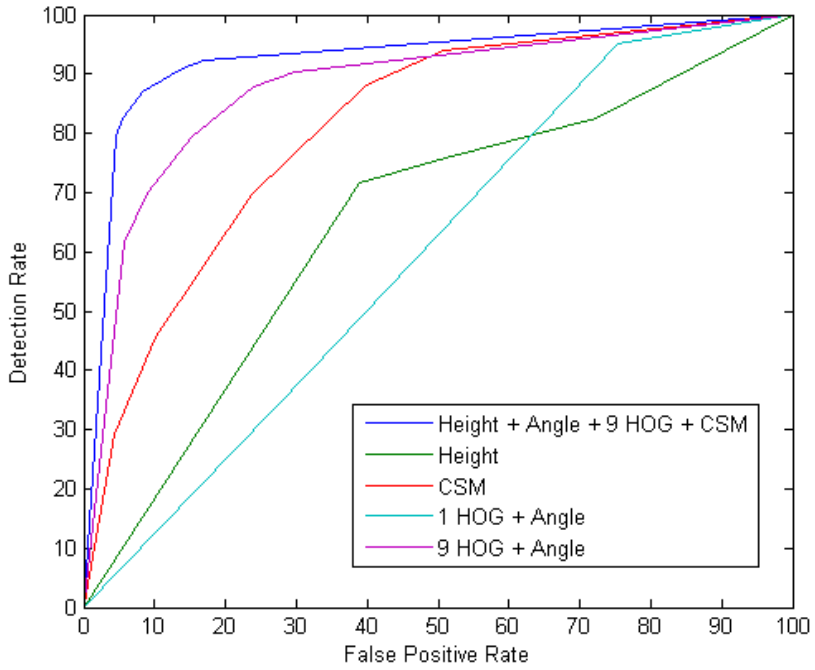

Figure 6.11: ROC curves for different SVM models with different features selection

Thirdly, the red curve shows the performance of the classification based only on the Concentric Square Mean feature. This selection brings good results especially considering the fact that each candidate is characterised by only 25 dependent variables.

Finally, the performance of classifiers using every feature (except the single HOG) are plotted as the blue curve. The satisfactory outcome is that these models are better than any other model thanks to the combination of the features. This comes from the fact that the features capture complementary information that the classifier exploit.

6.5 Detections Tracking

This section aims to present the results of the tracking algorithm, applied on detections through entire scenarios to count as accurately as possible the number of people evolving through the gate. First we discuss the possible values for the parameters of the algorithm, then we study the performance for each category of scenario.

As introduced in Chapter 4, the detections tracking algorithm uses seven parameters. The cost function attributing a weight to each edge, detailed in Equation 4.1, involves the parameters α, β, γ and δ. The function rejecting an inconsistent trajectory, detailed in Equation 4.2, involves the parameters $thres1$, $thres2$ and $thres3$. These parameters are dependent on the image scale, the camera shooting cadence, the quality of the detections (ratio between DR and FPR), and also on the situation staged in the scenario. The first dependencies being relatively fixed in our case, we designed the intervals presented in Table 6.4 according to the different situations available in our dataset. The chosen values are then selected using grid search, as proposed in Section 5.5 (Validation of detections tracking), such that the tracking algorithm is able to manage every situation. Although these values are found on dataset one, they are adapted to any other dataset for which detections are made with a similar SVM model, because they have a real physical meaning, suitable for person move.

Parameters	α	β	γ	δ	$thres1$	$thres2$	$thres3$
Intervals	3-7	0.5-1.5	0.5-3	0-1	4-9	0.3-0.6	70-100
Chosen values	5	1	1	0.2	5	0.4	100

Table 6.4: Appropriate interval and chosen value of the tracking parameters

Even though some other values would have been more suited for a particular category of scenarios, those chosen values are a good compromise between the different situations. Indeed, as detailed in Table 6.5, over the 66 scenarios used for the measurement of the tracking performance, only 4 were over-counted and 2 were under-counted.

Category	1 person	2-5 people	PRM	Objects	Unusual postures
Correctly-counted scenarios	11	10	10	16	13
Over-counted scenarios	0	0	2	0	0
Under-counted scenarios	0	2	1	0	1

Table 6.5: Results for each category of scenario

We note that the performance of the tracking algorithm is good for every category, no matter the DR and the FPR of the classifier. Indeed when one or several consecutive detections are missed by the two-step detection algorithm, the tracking is still able to recover the loss. Moreover, when additional (false) detections are returned, the tracking algorithm rejects the dummy trajectory passing through these detections or joins them into other trajectories. A few scenarios are however over or under counted. We are detailing here the situation producing these errors.

For the scenarios involving more than one person, someone can be missed if walking nearby another person with their head side by side, all along the scenario until the end. Moreover, a person walking all along the border of the gate can be missed because of the camera's lack of outlook. As detailed before and illustrated in Figure 5.1, we ignore detections from the borders of the images.

Concerning the scenarios involving PRM, additional trajectories can be wrongly counted as people whenever the person in wheelchair is carrying a head-like shaped object on his knees. This is the case for the two over-counted scenarios in this category.

For the scenarios involving unusual postures, a person can be missed whenever he is moving unexpectedly, generally with his head concealed. In the case of our scenarios for this category, the only missed person was crawling on all fours with someone else carried on his back.

6.6 Conclusion

This section summarises the results of the two-step people detection algorithm and of the detections tracking algorithm, which counts the consistent trajectories as people.

Table 6.6 presents the performance of the head candidates selection, based on local maxima, according to the selected parameters values. We observe a very high detection rate (based on the annotated ground-truth), although with many false detections.

Table 6.7 then presents the performance of the classification of the head candidates for equally weighted classes. The standard deviation is computed over a ten-fold CV through which the different categories of scenario are uniformly allocated. Here the average detection rate is quite a

	DR	FHCR
Head candidates	97.61%	134.79%

Table 6.6: Performance of head selection algorithm

bit lower, mostly because of some unusual situations not previously seen by the classifier during the training. On the other hand, the false positive rate is much lower than the one after the first-step algorithm.

		DR	FPR
Head classified	*Mean*	86.96 %	8.36 %
	Standard Deviation	6.53 %	5.17 %

Table 6.7: Performance of the head candidates classification algorithm

Table 6.8 presents the performance of the detections tracking algorithm, used to count the people moving through the gate. It is worth mentioning that the percentages reported in this table are evaluated over 66 scenarios in dataset one and are thus weighted equally for each scenario of every category, whose distribution is not representative of a real-life situation.

	Correct	Loss	Excess
Scenarios	90.91%	6.06%	3.03%

Table 6.8: Performance of the detections tracking algorithm

Thanks to the tracking algorithm, missed detections can be recovered such that the person is well detected at the end. On the other hand, false detections are grouped into inconsistent trajectories that are rejected, preventing thus the count of an additional person.

We can state that using in sequence the head candidate selection, the classification and the tracking algorithm, 90.91 % of the suitable scenarios of dataset one are correctly counted.

Chapter 7

Future Leads

7.1 Introduction

This chapter discusses possible improvements to the system described in this thesis and the generalisation of the methods used for the detection and the tracking. First, we state the problem of the adaptability of the system and propose a way to achieve it automatically. Then, we discuss a recommended improvement for more accurate detection by replacing the classification method by a regression method. Finally, we propose a generalisation of the two-class detection model studied in this thesis by a multi-class detection model.

7.2 System automatic adaptability

Our system can be adapted to a different usage, such as one with another type of gate or with a different camera height. Furthermore, the client could change the detection requirement about the importance balance between the two types of error (missed versus additional detections).

Rebuilding the system from scratch does not need too many changes. First, a new configuration needs to be determined for the head candidate selection, by tuning the parameters. Then, the SVM model needs to be trained on a small sample of images. Finally, the detections tracking can be tuned, although our proposed configuration is adapted for any situation involving people passages.
However, it is laborious and time consuming to proceed to such a rebuilding. Instead, we advise the reuse of the actual system. Whether the height of the camera was to be modified, we propose to simply add a preprocessing step adjusting the scale of the images pixels intensity to the present one. In the case of another type of gate, the present system is perfectly functional as long as the depth images are top-view. The most challenging change actually is the modification of the classes weight, i.e. if it becomes better or worse to detect a false person rather than to miss one. To solve this problem, we propose the implementation of the system automatic adaptability by allowing the input setting of the class weighting, so that the best configuration is automatically chosen for the head candidates selection, and the appropriate SVM model is selected among a number of different prerecorded models, with different balances between the weights of the two classes.

7.3 Regression approach

Among the possible further improvements, we propose to replace the linear classification by a regression method, such as Fully Convolutional Regression Networks [21]. Using neural networks is a way to achieve a very accurate measurement of the person density on depth images. The number of people can then be found by integrating the density map, by example using the image

integration method that we introduced in Section 3.3.

The drawback of using regression instead of classification is that the detections will not be represented by single points anymore, but by a group of pixels with high probability on depth maps. Therefore, the detections tracking algorithm we designed to recover missed detections (sometimes invisible on the image) or eliminate additional detections (caused by head-like shaped objects) will not be usable anymore.
Moreover, the real-time operation requirement could be a bottleneck for using such neural networks.

7.4 Generalisation

The present people detection is achieved by filtering and classifying regions of interest into two different classes, *person* and *non-person*. A possible generalisation of this system is to extend it to multi-class cases. By examples, classifying head candidates into *child*, *adult* and *PRM* classes to differentiate the children, adults and PRM. The detection could be even extended to large objects such as bikes, suitcases, bags, etc. The detections tracking algorithm would still be appropriate to group these detections into trajectories. It could even be improved by specifying in the appearance dissimilarity function a cost for the difference of class.

Also, as described in Section 7.2, the system can be adapted to other situations with different types of gates. By example, measuring the people flow at the entrance of a festival or any place where knowing the number of people is substantial. Moreover, the detections tracking algorithm outputs trajectories that can be analysed. By example, monitoring the people trajectories in unidirectional gates such as ones in airports.

7.5 Conclusion

Though the methods and algorithms developed in this thesis are effective, they can be improved or extended in different ways. First, we discuss the automatic adaptability of the system to a change of situation or requirement. Next, we introduce a lead to increase the detection by replacing the classification model by a regression model. Finally, we present a generalisation of the framework to a multi-class case in order to count the adults, the PRM, the children and eventually some objects. The outcome of this chapter is that, even after a year's worth of work, there is still room for improvement.

Chapter 8

Conclusion

This thesis proposes a framework to perform people detection on depth images taken by a time-of-flight camera. More specifically, the problem specified by the company *Automatic Systems* consists of measuring the flow of people passing through a gate during a given scenario.

The proposed solution operates first by filtering head candidates among the local maxima on each depth image. This technique is based on the hypothesis that a person's head generates always at least one local maximum and that its shape is the most recognisable part of the body on top-view depth images.
The second step of the algorithm consists of classifying the selected head candidates to the classes *person* or *non-person* by using a Support Vector Machine model. The classification input space is obtained by computing a vector of characteristics on a surrounding window, using very specific features such as the grid of Histograms of Oriented Gradients or the Concentric Squares Mean. Once detections are nominated in the depth images of a scenario, the framework uses multi-object tracking to group these detections into consistent trajectories. The measurement of the flow of people is then simplified to the counting of these consistent trajectories.

The hypothesis made for the head candidates selection and the features used for the classification proved to be effective. Indeed, the validation shows that performance is increased after each step. Multi-object tracking is, finally, very useful in recovering missed detections and discounting false detections on stand-alone images.

The framework proposed in this thesis can be used to perform people detection in real-time over the gate proposed by *Automatic Systems* and is generalisable to other types of gates. However, the performance can be further improved by the use of other Machine Learning algorithms, such as Neural Networks. The actual primary drawback of the framework is its incapacity to detect a person if the head is hidden throughout the scenario. The major improvement area resides therefore in the detection of the whole body of a person, not only of the head.

Bibliography

[1] Mark S Nixon and Alberto S Aguado. *Feature extraction & image processing for computer vision*. Academic Press, 2012.

[2] Richard A Haddad and Ali N Akansu. A class of fast gaussian binomial filters for speech and image processing. *IEEE Transactions on Signal Processing*, 39(3):723–727, 1991.

[3] 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*, volume 1, pages 886–893. IEEE, 2005.

[4] Franklin C Crow. Summed-area tables for texture mapping. In *ACM SIGGRAPH computer graphics*, volume 18, pages 207–212. ACM, 1984.

[5] Paul Viola and Michael J Jones. Robust real-time face detection. *International journal of computer vision*, 57(2):137–154, 2004.

[6] Dupont Pierre. Support vector machines. University Lecture - Machine Learning classification and evaluation. ICTEAM Institute, Université Catholique de Louvain – Belgium, 2018.

[7] Bernhard Schölkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[8] John Shawe-Taylor and Nello Cristianini. *Kernel methods for pattern analysis*. Cambridge university press, 2004.

[9] Damien Delannay, Christophe De Vleeschouwer, et al. Iterative hypothesis testing for multi-object tracking in presence of features with variable reliability. *arXiv preprint arXiv:1509.00313*, 2015.

[10] Amit Kumar KC, Damien Delannay, Laurent Jacques, and Christophe De Vleeschouwer. Iterative hypothesis testing for multi-object tracking with noisy/missing appearance features. In *Asian Conference on Computer Vision*, pages 412–426. Springer, 2012.

[11] Amit Kumar KC. *Detection-based multi-object tracking in presence of unreliable appearance features*. PhD thesis, UCL-Université Catholique de Louvain, 2015.

[12] Claudio Piciarelli, Christian Micheloni, and Gian Luca Foresti. Trajectory-based anomalous event detection. *IEEE Transactions on Circuits and Systems for video Technology*, 18(11):1544–1554, 2008.

[13] Mehmet Ocakli and Mubeccel Demirekler. Video tracker system for traffic monitoring and analysis. In *Signal Processing and Communications Applications, 2007. SIU 2007. IEEE 15th*, pages 1–4. IEEE, 2007.

[14] Jerome Berclaz, Francois Fleuret, Engin Turetken, and Pascal Fua. Multiple object tracking using k-shortest paths optimization. *IEEE transactions on pattern analysis and machine intelligence*, 33(9):1806–1819, 2011.

[15] De Vleeschouwer Christophe. Track detections. University Lecture - Image processing and computer vision. ICTEAM Institute, Université Catholique de Louvain – Belgium, 2018.

[16] Schauss Pierre. Column generation. University Lecture - Advanced Algorithms for Optimization. ICTEAM Institute, Université Catholique de Louvain – Belgium, 2018.

[17] Pierre Schaus, Pascal Van Hentenryck, Jean-Noël Monette, Carleton Coffrin, Laurent Michel, and Yves Deville. Solving steel mill slab problems with constraint-based techniques: Cp, lns, and cbls. *Constraints*, 16(2):125–147, 2011.

[18] Schauss Pierre. Modeling bin packing. University Lecture - Advanced Algorithms for Optimization. ICTEAM Institute, Université Catholique de Louvain – Belgium, 2018.

[19] James A Hanley and Barbara J McNeil. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 143(1):29–36, 1982.

[20] Keni Bernardin, Alexander Elbs, and Rainer Stiefelhagen. Multiple object tracking performance metrics and evaluation in a smart room environment. In *Sixth IEEE International Workshop on Visual Surveillance, in conjunction with ECCV*, volume 90, page 91. Citeseer, 2006.

[21] Weidi Xie, J Alison Noble, and Andrew Zisserman. Microscopy cell counting with fully convolutional regression networks. 2015.

[22] Sotiris B Kotsiantis, I Zaharakis, and P Pintelas. Supervised machine learning: A review of classification techniques, 2007.

[23] Carl Edward Rasmussen and Christopher KI Williams. *Gaussian processes for machine learning*, volume 1. MIT press Cambridge, 2006.

[24] Yuanqing Lin, Fengjun Lv, Shenghuo Zhu, Ming Yang, Timothee Cour, Kai Yu, Liangliang Cao, and Thomas Huang. Large-scale image classification: fast feature extraction and svm training. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1689–1696. IEEE, 2011.

[25] Benjamin Choi, Cetin Meriçli, Joydeep Biswas, and Manuela Veloso. Fast human detection for indoor mobile robots using depth images. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 1108–1113. IEEE, 2013.

Appendices

9.1 Measures of typical gate and camera set-up

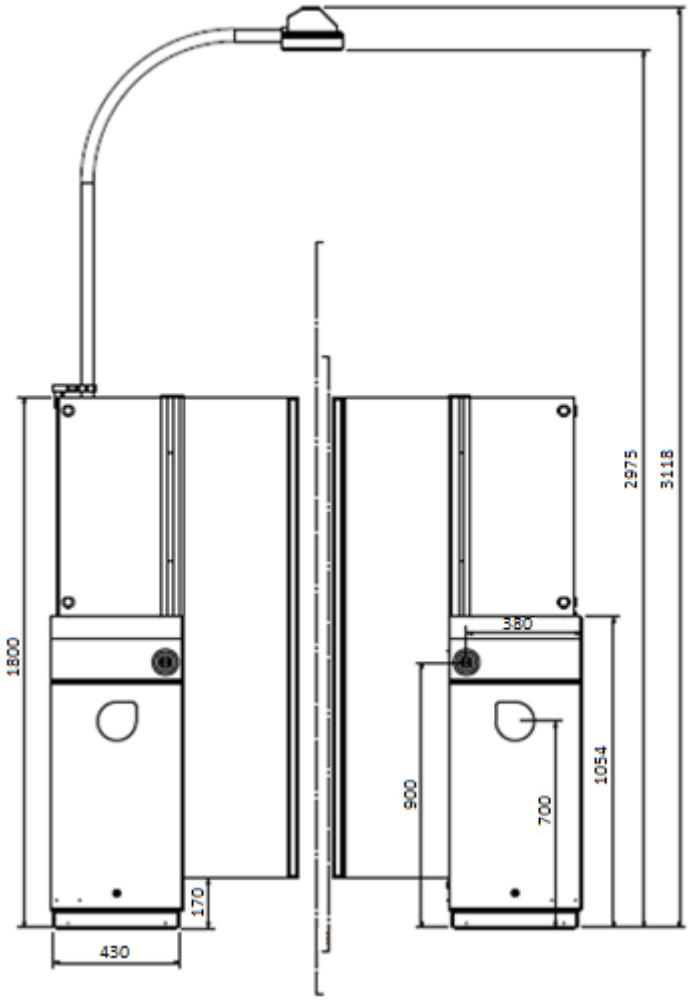

Figure 9.1: Typical gate and camera set-up with front measures

9.2 Reduction of the Steel Mill Slab Problem to the scenarios N-fold distribution problem

The Steel Mill Slab Problem (SMSP) [17] is reducible to the problem of dividing the scenarios into N sets in polynomial time. First we define the scenario partitioning problem, for which the goal is to minimise the difference of size between each scenario in term of examples, while the

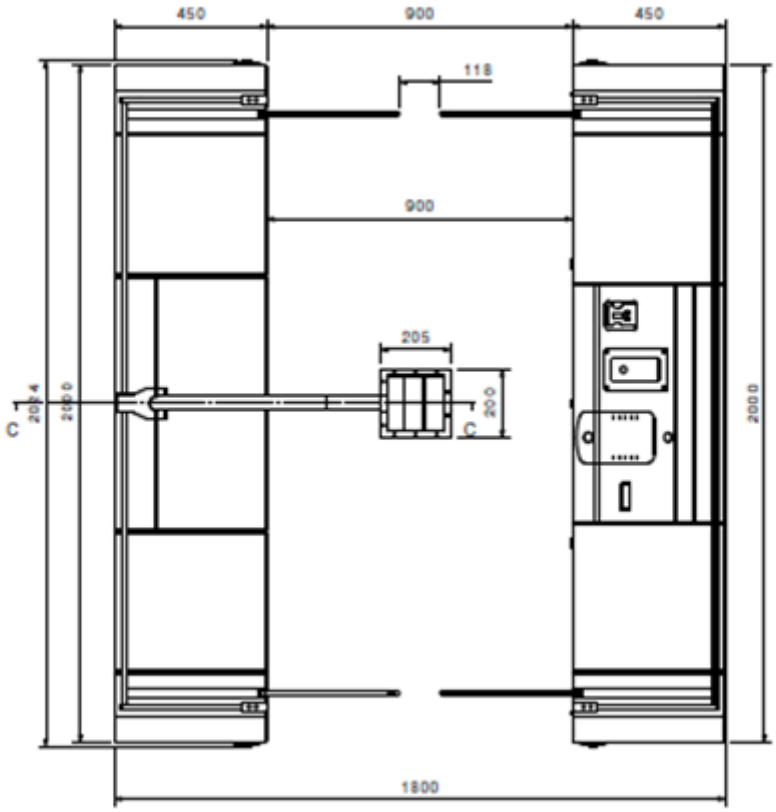

Figure 9.2: Typical gate and camera set-up with top measures

scenarios beholding examples cannot be split across different sets. Let $size(i)$ denote the size of the set number i and μ denote the average set size, which is constant and equal to the total number of examples divided by N. Then the formal definition of the problem is the following:

$$min \sum_{i=1}^{N} |\mu - size(i)| \tag{9.1}$$

The section aims then to show the reduction of the SMSP to this partitioning problem to prove its NP-hardness. The SMSP is an old problem describing a mill where steel can be produced by casting molten iron into some slabs of fixed size. Orders, that are generally characterised by both a colour and a weight, must be allocated into the slabs such as the total weight of each slab do not exceed its maximal capacity, while the sum of the losses is minimised. In the scope of this section, we will suppose a single unique colour for the orders in order to simplify the presentation while keeping the problem entire. The formal definition is then, following [18]:

$$min \sum_{j=1}^{m} loss_j$$

$$\text{subject to} \sum_{i \mid o_i = j} w_i < capacity_j \quad \forall\ j \in [1..m]$$

Where

$m \in \mathbb{N}$	is the number of available slabs.
$capacity_j \in \mathbb{R}$	is the maximal capacity of slab j.
$loss_j \in \mathbb{R}$	is the loss of slab j, i.e. the used space in this slab.
$o_i \in [1..m]$	is the assigned slab to order number i.
$w_i \in \mathbb{R}$	is the weight of order i, i.e. the consumed space in the allocated slab.

Solving the partitioning problem of the scenarios into N sets while minimising the difference of size between each set, as described in Equation 9.1, gives also a solution of one instance the SMSP with $m = N$, w_i being the size of the scenario i and for each slab j : $capacity_j = max_{\text{set } k}(size(k))$. Indeed it can be shown that if the sets are balanced optimally, then the loss of space between each size and the maximal size is also minimal.
Since every instance of the SMSP can be reduced in polynomial time to an instance of the partitioning problem, we state that the partitioning problem is NP-hard.

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