Video surveillance using a time-of-light camera

Davide Silvestre

Kongens Lyngby 2007
IMM-THESIS-2007-60

Technical University of Denmark
Informatics and Mathematical Modelling
Building 321, DK-2800 Kongens Lyngby, Denmark
Phone +45 45253351, Fax +45 45882673
reception@imm.dtu.dk
www.imm.dtu.dk

IMM-THESIS: ISSN 0909-3192

Summary

The research of these last years is more and more focusing on building systems for observing humans and understanding their appearance and activities.

The aim of this work is to implement a people detection and tracking algorithm using the time-of-flight camera SwissRanger 3000. This kind of camera is able to measure both the grayscale image of the scene and the depth for each pixel. With these two types of information the images coming from the camera can be processed to extract the blobs present on them, detect people moving and track them in the video. Besides the interest will be pointed not only on the results that can be obtained, but also on the analysis of the limitations and problems encountered using this camera prototype.

Preface

This thesis was carried out within the Image analysis and Computer Graphics group at the Informatics and Mathematic Modelling (IMM) department of the Technical University of Denmark (DTU) between the 22^{nd} of January and the 29^{th} of June.
The thesis serves as a requirement for the Master of Science degree in engineering, M.Sc.Eng. The extent of the project work is equivalent to 30 ECTS credits and was supervised by professor Rasmus Larsen and co-supervised by Dan Witzner Hansen.
I would like to thank my supervisors for the help and suggestions they gave me, the colleagues in the laboratory who acted in the sequences I took with the SwissRanger camera, Sigurjón Árni Gudmundsson who showed me how to use correctly the camera and Eina Boeck for the welcome given in the IMM department.

Furthermore I would like to thank my home university, the university of Florence, and my Italian supervisor, Carlo Colombo, that allowed me to work on my master thesis abroad.

Lyngby, July 2007

Davide Silvestre

Contents

Chapter 1

Introduction

1.1 Video Surveillance

In the last few years an important stream of research that has gained a lot of importance within computer vision is the understanding the human activities by analyzing video sequences. This kind of research has applications in many fields, the most important of which is *video surveillance*. Of course this kind of technology is used also in other fields like in the character animation for movies and games, biomedical applications, avatars for teleconferencing, etc. Regarding to the video surveillance in the literature there are a lot of proposals whose aim is to detect and track the people. In the next paragraph an overview of the principal methods for people detecting is presented.

1.2 Human detection and tracking

The relevant literature regarding human detection can be divided into methods that require background subtraction and other techniques that can detect humans directly from the input without a preprocessing. The methods using the background subtraction usually extract the foreground and classify it into cat-

egories like car, human, animal etc. This detection is usually performed using features like shape, color or motion.

The other kind of methods use features directly extracted from the image. These features include shape (i.e. contour), motion, color (i.e. the skin color) and also combinations of them.

Regarding people tracking it is possible to use high level knowledge and find people by means of the positions of the legs, the head, the arms and the body. Otherwise if no high level knowledge is used, people are tracked as simple objects like in the work of Tsukiyama and Shirai [15]. They detect moving people in a room by first detecting moving objects, then finding the objects corresponding to humans, and finally tracking the moving humans over time. The Walker system of Hogg [7] extracts the edges in the image and searches people bodies comparing the edges found with a human model. Regardless the method used, three common aspects can be identified:

1. Separate the background from the foreground which contains the blobs to track.
2. The segmented images are transformed into another representation to reduce the amount of information to store in the memory.
3. Definition of the method which the blobs are tracked with.

The segmentation between the foreground and the background can use either *temporal* or *spatial* information. In the case of a static camera the pixels belonging to a moving object can be detected by analyzing the differences among following frames. This method is weak especially if the background is not stable or if there is some noise. For that reason some improvements have been used, like using more frames instead than just two or using these differences to update a background model. Another possibility to use the temporal information is to calculate the direction for all the moving pixels by analyzing following frames and group them if they have the same *flow*.

Spatial methods can be divided into two categories:

1. Thresholding approaches.
2. Statistical approaches.

An example of thresholding approach can be the case in which the background is known and foreground is extracted by tresholding the differences between

the current value and the background value. In the statistical methods the background is analyzed and for each pixel some information are collected, like the mean value and the variance over the time. According to these values the pixels are classified between the ones belonging to the background and to the foreground. An advanced version for this method is to consider different statistical features for each blob composing the object or the human to track.

After having found the blobs in the different frames the next step is *to trak*, by finding corresponding objects in consecutive frames. The difficulty of this task depends on the complexity of the scene and of the blobs to track.

1.3 Common problems in people tracking

To perform the tracking of people, through a background subtraction algorithm, using a stationary camera able to measure the intensity values (color or grey-scale) there are some problems which it is necessary to face with:

1. **Intensity similarity:** if the moving object has an intensity similarity with the background it is more difficult to extract it and classify it as a foreground blob because it is less clear the difference between them.
2. **Presence of shadows:** the presence of shadows on the background caused by the moving people or the moving object in the scene could generate false detections, because the shadows can cause an enough variation of intensity to induce the system to label that area as foreground.
3. **Light changes:** also the light changes could cause false positives in the foreground, since this change modifies the whole background model.
4. **Overlap:** besides in case the aim is to extract the single moving blobs, it is harder if the blobs overlap with one another.

It is reasonable to think that using the depth information given by the Swiss-Ranger it is possible to solve some of these problems by combining the grey-scale values with the depth. As a matter of fact the depth is not sensitive to light changes and to the shadows. Besides during a partial overlapping the depth information can help in dividing the shapes of the people.

In the literature the depth information has been used to perform the human detection by F. Xu and K. Fujimura [1]. In their work the used a device similar to the SwissRanger, able to measure the brightness and the depth of the scene.

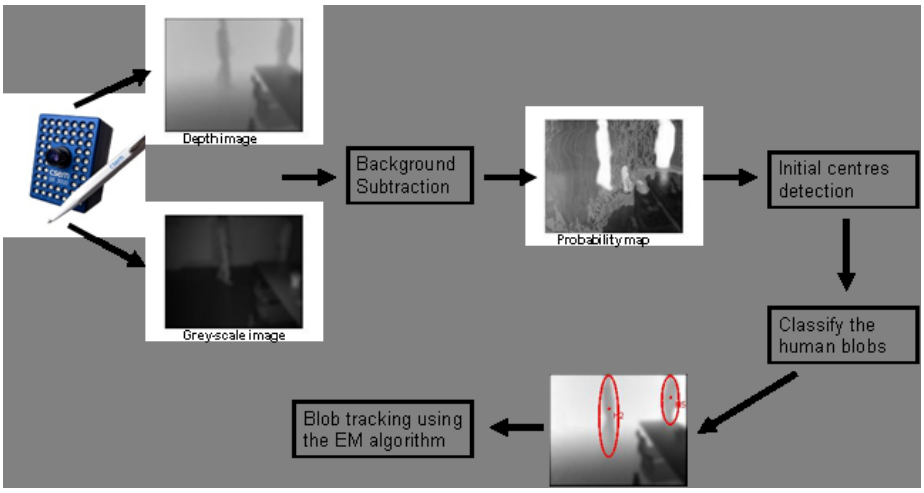

Figure 1.1: The scheme of the thesis.

To extract the human blobs they set the wave length of the camera so as to detect the foreground, which people move in, but not the background, too far for the current modulation frequency. In this way they can extract the foreground blobs, that correspond to the only part of the scene detected by the device. Of course this method depends much on the particular scene that has to be monitored and may have problems if the scene is not completely empty until the background wall.

In this work the depth information is used to implement a tracking algorithm much more independent from the environment and the illumination intensity and the first step to implement this has been the extraction of the foreground from the images, through a background subtraction algorithm.

1.4 Objectives of this work

The aim of this thesis is to employ the time-of-flight camera SR-3000 to perform the people detection and tracking using this prototype.

As shown in figure 1.1 both the grey-scale and the depth images are used to perform the background subtraction by associating to each pixel a probability to belong to the foreground. After that the foreground is segmented in homogenous regions, that are classified in two categories: human and non-human. For the

blobs considered humans the tracking is performed also considering the case of occlusions among people. As a matter of fact it is easier to split blobs belonging to different people, even if they overlap, if it is possible to know their depth.

In the chapter 2 the SwissRanger camera is described: the physical principle behind it and also the limitations of the current prototype. After that in the chapter 3 the first step in the algorithm, the background subtraction, will be shown as it is possible to see from the figure 1.1. In the chapter 4 the core of the algorithm will be presented; here it is described how the blobs are extracted using the probability map that gives for each pixel the likelihood to belong to the foreground. Using this information the foreground is divided in blobs whose shape is represented with ellipses. The parameters of the ellipses are estimated using the EM algorithm and kept updated in each frame starting from the values of the previous one. In the chapter 5 there is the description of the classification of the blobs into human and non-human. In the end, in the chapter 6, the experimental results will be presented and discussed.

Chapter 2

Swiss Ranger SR-3000

The SwissRanger is a range detection camera developed by the CSEM Zurich Center that uses the TOF technology.

The SwissRanger camera is based on a phase-measuring time of flight (TOF) principle. This tool emits a near infrared wave front that is intensity-modulated with a few tens of MHz. This light reflects on the scene and returns on the optical lens. The distance of the sample is calculated according to the phase delay that the wave has compared to the originally emitted light wave.

2.1 Time-of-flight principle

As we know very precisely the speed of the light in air for different environmental conditions it is possible to measure an absolute distance by measuring the time taken by a light pulse to travel from a target to a detector.

Rather than directly measuring a light's pulse total trip the SwissRanger measures the phase difference between the sent and the received signals. As the modulation frequency is known it is possible to obtain the distance by this measured phase difference.

Figure 2.1: The SwissRanger SR-3000 camera.

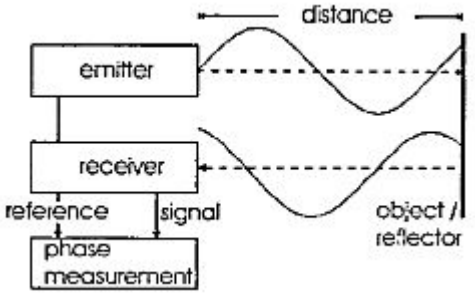

Figure 2.2: The time of flight principle.

We consider the following emitted signal,

$$e(t) = 1 + \cos(\omega t) \tag{2.1}$$

the received signal $s(t)$ and the correlation signal $g(t)$

$$s(t) = 1 + a \cdot \cos(\omega t - \varphi) \text{ and } g(t) = cos(\omega t), \tag{2.2}$$

The correlation function between them can be calculates as:

$$c(\tau) = s(t) \otimes g(t) = \frac{a}{2} \cdot cos(\varphi + \tau) \tag{2.3}$$

To calculate the parameters a and φ the function $c(\tau)$ is evaluated for four different phases: $\tau_0 = 0$, $\tau_1 = \pi$, $\tau_2 = 2\pi$, $\tau_3 = 3\pi$ and in these way it is possible to obtain the four following measured values:

$$C(\tau_O) = c(\tau_0) + K = +\frac{a}{2} \cdot cos(\varphi) + K \tag{2.4}$$

$$C(\tau_1) = c(\tau_1) + K = -\frac{a}{2} \cdot sin(\varphi) + K \tag{2.5}$$

$$C(\tau_2) = c(\tau_2) + K = -\frac{a}{2} \cdot cos(\varphi) + K \tag{2.6}$$

$$C(\tau_3) = c(\tau_3) + K = +\frac{a}{2} \cdot sin(\varphi) + K \tag{2.7}$$

Hence, we can determine the phase φ and the amplitude a of the signal $s(t)$

$$\varphi = \arctan\left(\frac{C(\tau_3) - C(\tau_1)}{C(\tau_O) - C(\tau_2)}\right) \tag{2.8}$$

$$a = \frac{\sqrt{(C(\tau_3) - C(\tau_1))^2 + (C(\tau_O) - C(\tau_2))^2}}{2}. \tag{2.9}$$

At this point it is possible to calculate the distance D using the following equation:

$$D = \frac{c}{2f_m} \cdot \frac{\varphi}{2\pi} \tag{2.10}$$

where f_m is the modulation frequency and c is the speed of the light.

2.2 TOF vs Stereo Vision

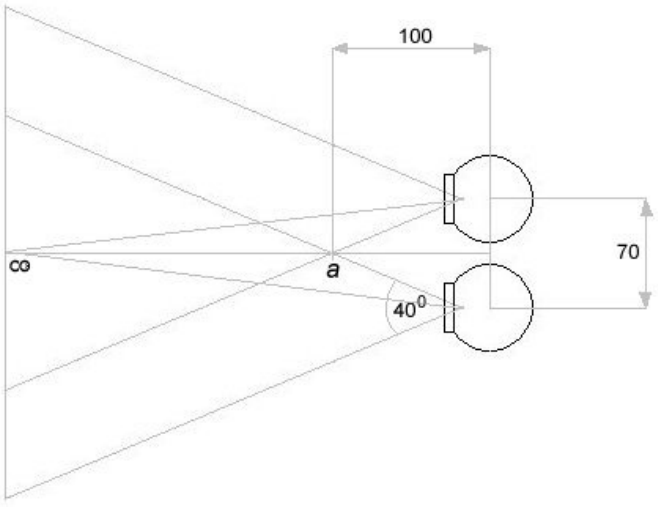

Figure 2.3: The stereo vision scheme. Using two traditional cameras it is possible to measure the depth of a point by intersecting the direction of the two different cameras.

To measure the depth among objects in computer vision there are a lot of methods that use two or more cameras. If the aim is to measure the distance from the scene it is possible to use just one tool, the SwissRanger, instead of more traditional cameras. In the following paragraph the main differences between the usage of the SwissRanger 3000 and the stereo vision are compared:

Table 2.1: Comparison between Stereo Vision and the SwissRanger

	Stereo Vision	*SwissRanger*
Portability	Two video cameras are needed and also an external light source	The size of the SwissRanger camera can be compared with a normal camera
Resolution	It is possible also to achieve a sub-millimeter resolution, but this depends on the distance between the two cameras and on the contrast in the scene	The resolution is sub-cm and there are no problems for uniform scenes, but may be affected by reflection angle
Computation	The search of correspondences may be hard to compute	Phase and intensity calculation are simple and can be performed directly in the hardware
Cost	It depends on the quality of the two high resolution cameras	The prototype costs 5.000,00 euro, but the price could be decreased when mass-produced

2.3 Limitations of the SwissRanger

The SwissRanger is very sensitive to the integration time and modulation frequency parameters and it is also affected by physical limitations. In this paragraph some of the main limitations encountered during the thesis work are described and they will be recalled and underlined in the experimental results sections.

2.3.1 Multiple reflections

All the SwissRanger LEDs are synchronized and all of them acquire the image simultaneously. In some cases, due to the light reflection, two different rays can be measured by the camera. As we can see in figure 2.4 the ray emitted can reflects on a surface and be deviated. This happens above all if there are corners in the scene. During the development of the tracking system this problem has been detected when people were close to the camera and the reflected rays in

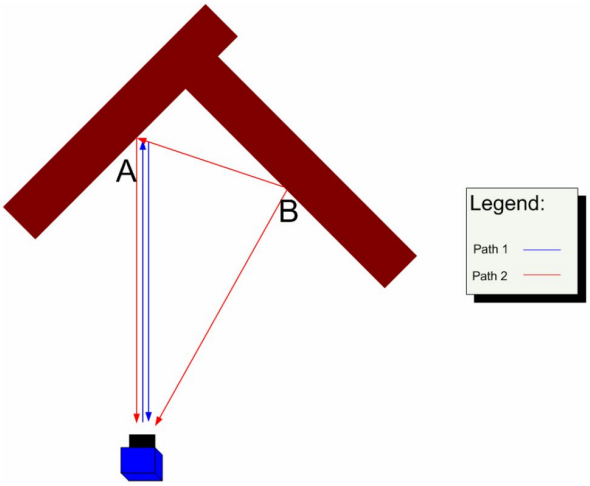

Figure 2.4: An example provided by the SR guide in which it is possible to see the deviation of the ray directed towards the point A.

this case have caused the presence of "fog" around the shape of the person. Examples of this case will be provided in the experimental results sections.

2.3.2 Not uniform reflection

The way in which an object reflects the light depends also on its texture and on its color. This can be seen by measuring the depth of a white sheet with some black vertical stripes. As we can see in figure 2.5 the depth measured on the black stripes is lower than the one for the white part of the sheet, even if the sheet is plane.

2.4 Image acquisition

To improve the quality of the sequences taken with the SwissRanger there are some aspects to take into account. First of all it is important to place the camera so that to reduce the reflection problem.

As seen in figure 2.6 it is better to place the camera on the border of the platform, in this way the ray are not reflected by the support. A practical example of this

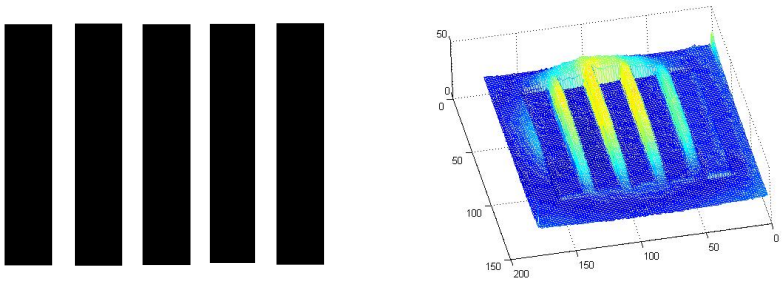

Figure 2.5: As it is possible to see from this example the depth measured on the sheet is not uniform.

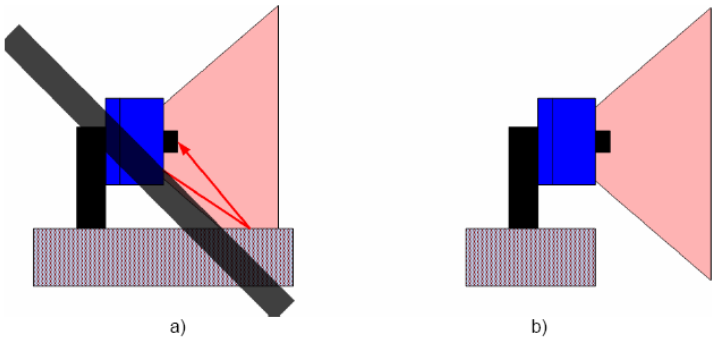

Figure 2.6: On the left there is an example of wrong mounting of the camera. In that case the rays reflect and cause noise. On the right the ideal mounting is shown

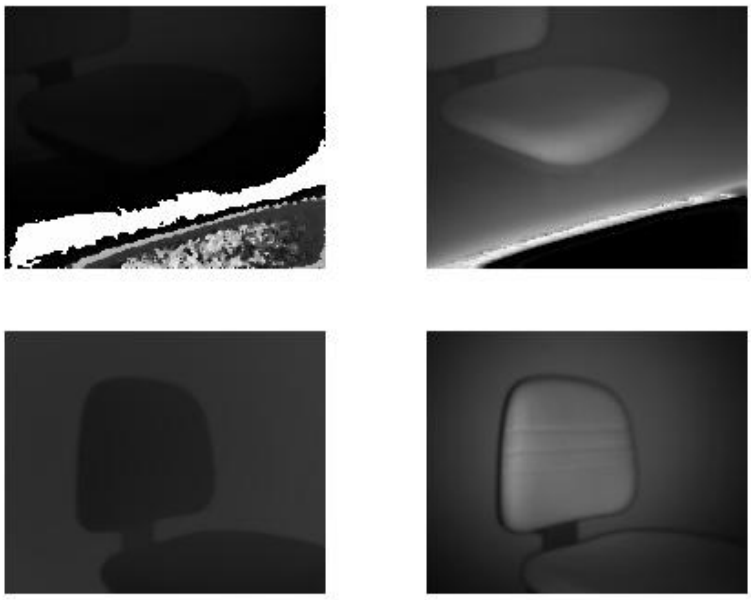

Figure 2.7: In the first row there are the depth and brightness images taken mounting the camera in a wrong way and below the same scene taken placing the camera on the border of the desk.

problem is shown in figure 2.7 where it is possible to evaluate the depth and the brightness of the same scene placing the camera in the right way and in the wrong way.

During the image acquisition one of the most sensitive parameters of the camera is the integration time, that regulates the exposure time and can be varied from 200 μs to 51.2 ms in steps of 200 μ, where 0 corresponds to 200 μs and 255 to 51.2 ms.

This parameter must be tuned according to the scene. As a matter of fact for a close object it is better to use small integration time, whilst for long-distance scenes a high integration time can reduce the noise.

In the figures 2.8 and 2.9 there is the same cup taken with two different integration times. As the cup is placed close to the camera the best results are obtained with a low integration time. As a matter of fact with a bigger integration time the surface of the cup reflects the rays and the measurement is wrong.

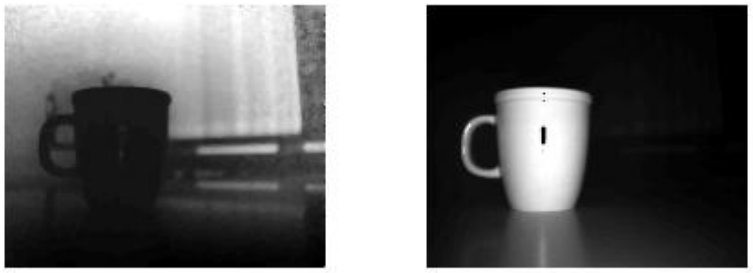

Figure 2.8: The depth (on the left) and the brightness (on the right) of a cup placed close to the camera and taken with an integration time equal to 10.

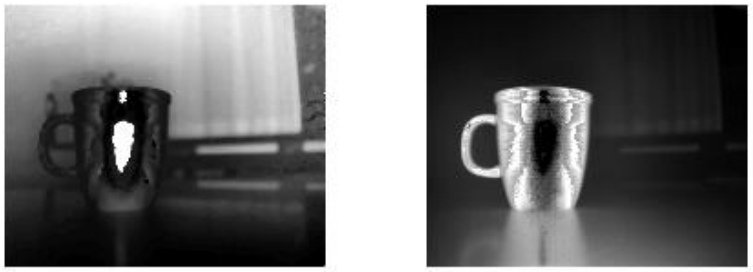

Figure 2.9: The depth (on the left) and the brightness (on the right) of a cup placed close to the camera and taken with an integration time equal to 100.

Figure 2.10: The depth (on the left) and the brightness (on the right) of a chair taken with an integration time equal to 10.

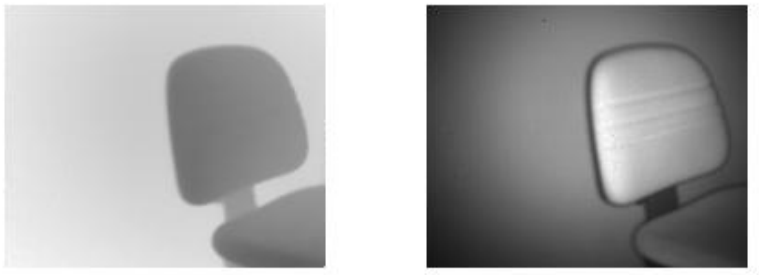

Figure 2.11: The depth (on the left) and the brightness (on the right) of a chair taken with an integration time equal to 100.

On the contrary if the scene is far from the camera the best results are obtained using a bigger integration time. In the figures 2.10 and 2.11 there is a chair taken with two different integration times: 10 and 100. As it is possible to see the results using 100 are better than the ones using 10.

Another important parameter to take into account is the modulation frequency. With this parameter it is possible to change the frequency of the camera and consequently the wave length. In this way the maximum depth that the camera can reach can be changed.

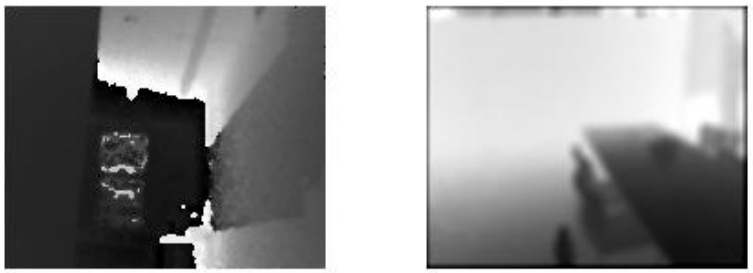

Figure 2.12: On the left there is an example of a scene in which the wall is too far for the modulation frequency and for that reason it is dark as it was in front of the camera. On the right there is an example in which the modulation frequency has been set correctly and in fact the background is the lighter part of the scene.

Chapter 3

Background subtraction

To track people in a video the first step that has been implemented is the background subtraction.

Background subtraction is a widely used approach for detecting moving objects using static cameras. The principle behind this kind of methods is that the pixels belonging to the background are stable and do not vary the brightness values as much as the pixels belonging to moving objects. This argumentation is true since a static camera is used; in case of a moving camera other methods should be implemented.

The background subtraction methods build a background model analyzing the frames and extract the foreground by performing a subtraction between the current frame and the model built. The background image is a representation of the scene without moving objects and it is regularly updated to adapt the scene to the changes of the people and objects positions and to the varying luminance conditions. Of course the speed of the background model updating depends on the application.

Figure 3.1: Using a traditional camera the background subtraction is performed analyzing following frames of the brightness.

3.1 Tracking people using background subtraction

To perform the people detection and tracking most of the techniques found in the literature employ the background subtraction as first step. Also in this work, as a stationary camera is used, the first stage of the algorithm consists of a background subtraction phase. In the literature the background subtraction has been used with different variations.

Wren et al. [5] build the background model using a gaussian distribution in the YUV space and model a person through the blobs extracted. Using the color and spatial information the blobs are associated to the different parts of the body starting from the head and the hands. Beleznai et al. [3] compute the difference image between the current frame and the reference image, extract the blobs using the Mean Shift algorithm and find the people using a simple model composed by a set of rectangles. Another way to detect people is considering the blobs movements after having extracted them from the image. Haga et al. [14] classify the blobs as human or non-human regarding to their motion features. Toth and Aach [16] detect the foreground blobs using the connected components and use the first ten fourier components as descriptor. After that the classification is performed using a neural network composed by four feedforward layers. Another possibility, used by Lee et al. [6], is to perform the detection using the shape. For each moving object the contour is reduced to 60 points, that are used to classify it as a human, a vehicle or an animal. Yoon and Kim [17] use both the skin color and the motion information. With this information the blob is resized and the classification is made by a SVM based classifier.

3.2 Background subtraction methods

In the literature there is a wide variety of techniques for performing background subtraction and all of them try to effectively estimate the background model from the temporal sequence of the frames. In this section the main methods to perform the background subtraction are presented and in the next ones the adaptation of some of them for the time-of-flight camera.

3.2.1 Running gaussian average

The idea proposed by Wren et al. [5] is to fit a Gaussian probability density function on the last n pixels' values updating independently each pixel (i, j) by running a cumulative average of the form:

$$\mu_t = \alpha I_t + (1 - \alpha)\mu_{t-1} \tag{3.1}$$

where I_t is the pixel's current value and μ_{t-1} the previous average. Besides α is a learning rate whose value must be chosen as a trade off between stability and quick update. The standard deviation σ_t can be calculated with the same principle and it is possible to classify each pixel as foreground if the following inequality is satisfied:

$$| I_t - \mu_t | > k\sigma_t \tag{3.2}$$

otherwise it is classified as a background pixel.

3.2.2 Temporal median filter

Some authors have argued that other kinds of temporal averaging perform better than the one exposed in 3.2.1. Instead of consider the average for each pixel Lo and Velastin [8] and Cucchiara et al. [12] propose to use the median value of the last n frames arguing that this method allow to extract an adequate background model even if the frames are subsampled. The main drawback of this method is that the last n frames must be stored in the memory and it does not provide a deviation measure like in the previous method.

3.2.3 Mixture of Gaussians

Sometimes the changes in the background are not permanent but one background pixel can present over time different intensity values like in the case of a tree with moving branches or the waves of the sea. For that reason each pixel should be associated to a set of values that might occur in the background at that position. Stauffer and Grimson [13] model the probability of observing a certain pixel value x, at time t by means of a mixture of Gaussians:

$$P(x_t) = \sum_{i=1}^{K} \omega_{i,t} \eta(x_t - \mu_{i,t}, \Sigma_{i,t}) \tag{3.3}$$

where η is a normal function with average μ and standard deviation Σ. In this model each of the K gaussian function describe only one observable background value. Usually K is set to be between 3 and 5.

At each frame the parameters of the model are updated using the expectation maximization algorithm and a pixel is considered to belong to the foreground if it does not belong to any of the gaussians modelling the background.

In the paragraph 3.4 a deeper description of this method will be presented and also its extension for the time-of-flight camera.

3.2.4 Kernel density estimation

Elgammel et al. [2] model the background distribution by a non parametric model based on kernel density estimation (KDE) using the buffer of the last n background values.

In this method the pdf is given as a sum of gaussian kernels centered in the most recent n background values, x_i:

$$P(x_t) = \frac{1}{n} \sum_{i=1}^{n} \eta(x_t - x_i, \Sigma_t) \tag{3.4}$$

based on this equation, the pixel x_t is classified as foreground if $P(x_t) < T$, where T is a threshold.

In the paragraph 3.5 the reader will find a deeper description of this method with the extensions for the time-of-flight camera.

3.2.5 Co-occurrence of image variations

This method, presented by Seki et al. [9], exploits the fact that pixels belonging to the background should experience similar variations over time. For that reason this algorithm, instead of analyzing the image pixel by pixel works on blocks of N by N pixels.

1. For each block the temporal average is computed using some samples of the block and also the differences between the samples and the average are considered.
2. After that the covariance matrix $N^2 * N^2$ is calculated and the dimension is reduced to K by means of an eigenvector transformation.
3. In this way the blocks are clusterized together according to their similarity in the eigenspace.

3.2.6 Eigenbackgrounds

Also the approach presented by Oliver et al. [10] is based on an eigenvalue decomposition, but this time this analysis is made all over the image, without dividing it in blocks.

1. Using n images the image average is computed and also all the differences between the images and the mean image
2. The covariance matrix is computed and the first M eigenvectors are stored in an eigenvector matrix Θ_{Mb} of size $M * p$
3. Each new image is then projected onto the eigenspace as $I' = \Theta_{Mb}(I - \mu_b)$
4. After that the image is back projected as $I'' = \Theta_{Mb}^T I' + \mu_b$. In this way I'' will not contain the moving objects because the eigenspace is a model for the static part.
5. The foreground points can now easily be found through a threshold T if $|I - I''| > T$.

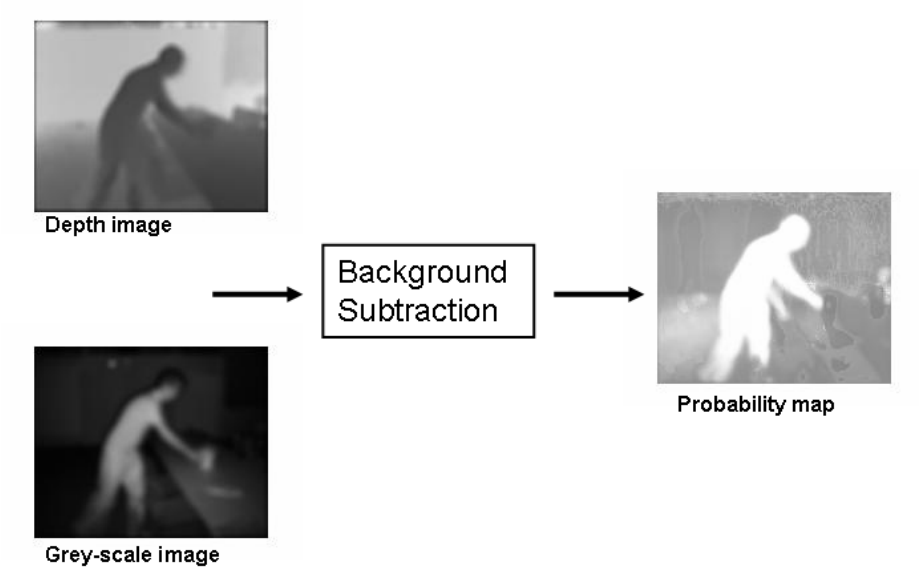

Figure 3.2: Using the SwissRanger it is possible to perform the background subtraction using both the depth and the brightness information.

3.3 Background subtraction for the TOF camera

The previous sections showed the reason why in this work the background subtraction has been implemented and also an overview of the methods to perform it that are present in literature.

Now the methods implemented for the TOF camera will be shown in detail remarking the extensions so that to use both the grey-scale and the depth information. Using the depth it is reasonable to think to solve some of the classic problem related to the background subtraction, such as the instability caused by the light changes or by the presence of shadows and also the extraction of people when the background brightness is very similar to the cloths. As a matter of fact the depth information is not sensitive to illumination or shadows and can detect more easily people moving.

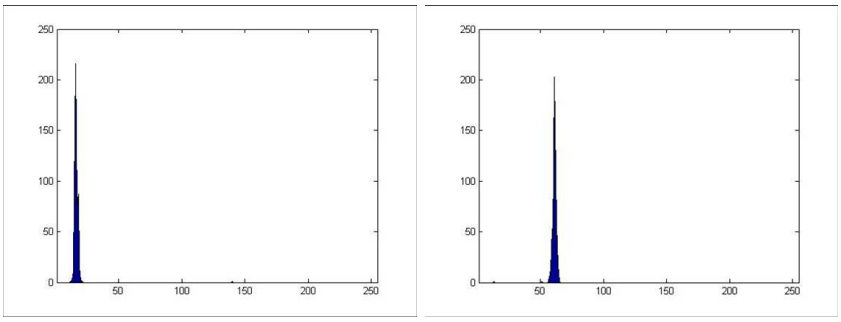

Figure 3.3: Histograms for a *stable* pixel showing the grayscale occurrences on the left and the depth on the right.

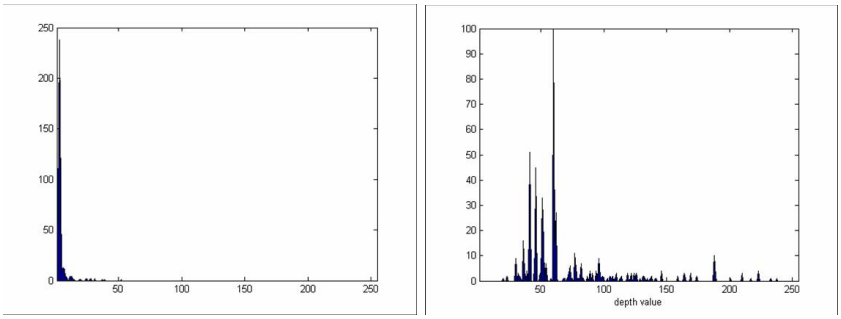

Figure 3.4: Histograms for a *moving* pixel showing the grayscale occurrences on the left and the depth on the right.

3.3.1 Improvements using the depth

Sometimes, especially for the grey-scale sequences for which the color information is not available, pixels belonging to a foreground blob are not detected because of the similarity between the blob brightness with the background. In this case the depth information can help to extract more easily the blob because of the gap between its depth and the background. In figure 3.3 it is possible to see the histograms for the depth values and the grayscale values of one stable pixel. If the pixel is not touched by moving objects its values are very concentrated around the background value.

Otherwise if people pass over it, its values are more instable as we can see from the histograms in the figure 3.4. Besides it must be noticed that the information coming from the depth is more clear and there it is more visible the instability of that pixel.

What we want from a background subtraction method is to associate to each pixel a probability measuring how likely it is that that pixel belongs to the background by using both the information coming from the depth and the intensity values. In the sections 3.4 and 3.5 two methods are presented, whose aim is to perform that.

3.4 MOG for the TOF camera

To perform the extraction of the foreground the method proposed by Stauffer and Grimson [4] has been analyzed and extended to exploit also the depth information.

In this method the probability that a pixel belongs to the foreground is modelled as a sum of normal function, rather than describing the behavior of all the pixels with one particular type of distribution. For each pixel some of these gaussians model the background and the others the foreground; in this way the single pixel is classified according to the fitting it has with these gaussians. The probability that the pixel has value X_t can be written as:

$$P(X_t) = \sum_{i=1}^{K} \omega_{i,t} * \eta(X_t, \mu_{i,t}, \Sigma_{i,t}) \tag{3.5}$$

where K is the number of distributions considered, usually between 3 and 5, $\omega_{i,t}$ is an estimate of the weight of the i^{th} distribution in the mixture at time t, $\mu_{i,t}$ is the mean value, $\Sigma_{i,t}$ is the covariance matrix and η is a gaussian probability density function as:

$$\eta(X_t, \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(X_t - \mu_t)^T \Sigma^{-1} (X_t - \mu_t)} \tag{3.6}$$

Under the assumption of the independence of the color channels the covariance matrix Σ could be written as:

$$\Sigma_{k,t} = \sigma_k^2 I \tag{3.7}$$

3.4.1 The updating algorithm

Rather than using the Expectation Maximization method to update the parameters, Stauffer and Grimson [4] propose an approximate method:

1. Each new pixel value X_t is checked against the current K distributions to verify if there is the following matching:

$$\frac{|X_t - \mu_{i,t}|}{\sigma_{i,t}} > 2.5 \tag{3.8}$$

2. If no distribution matches the current value the least probable gaussian is replaced with a normal function with mean equal to the current value, an initial high variance and low weight.

3. The weights are updated according to:

$$\omega_{k,t} = (1 - \alpha)\omega_{k,t-1} + \alpha(M_{k,t}) \tag{3.9}$$

where $M_{k,t}$ is 1 for the matched model and zero otherwise. After the updating the weights are normalized so that their sum is 1 for each pixel.

4. For the unmatched distributions the values for μ and σ remain the same and for the matched ones they change according to the following equations:

$$\mu_t = (1 - \rho)\mu_{t-1} + \rho W_t \tag{3.10}$$

$$\sigma_t^2 = (1 - \rho)\sigma_{t-1}^2 + \rho(X_t - \mu_t)^T(X_t - \mu_t) \tag{3.11}$$

where the learning rate ρ is:

$$\rho = \alpha\eta(X_t|\mu_k, \sigma_k) \tag{3.12}$$

In this way the more a pixel presents the same value (or a very close one to the average) and the more that distribution becomes relevant. Besides the main advantage of this method is that when a new value enters in the background image the past history is not destroyed, but it is kept in the model.

3.4.2 Adaptation for the TOF camera

For the time-of-flight camera the distance between the samples has been considered in a two-dimensional grayscale-depth space and can be written as:

$$d_{j,i} = \sqrt{(I_j - I_i)^2 + (Depth_j - Depth_i)^2} \tag{3.13}$$

where I are the intensity values and $Depth$ the depth values.

In this way the information coming form the grayscale values and the depth ones are used in the same way and the gaussian functions are built in this two

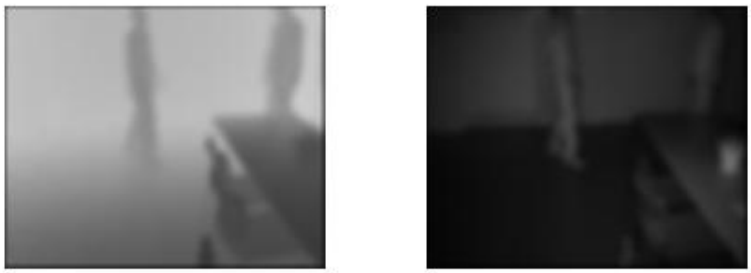

Figure 3.5: Test frame for the MOG algorithm (depth image on the left and grey-scale on the right).

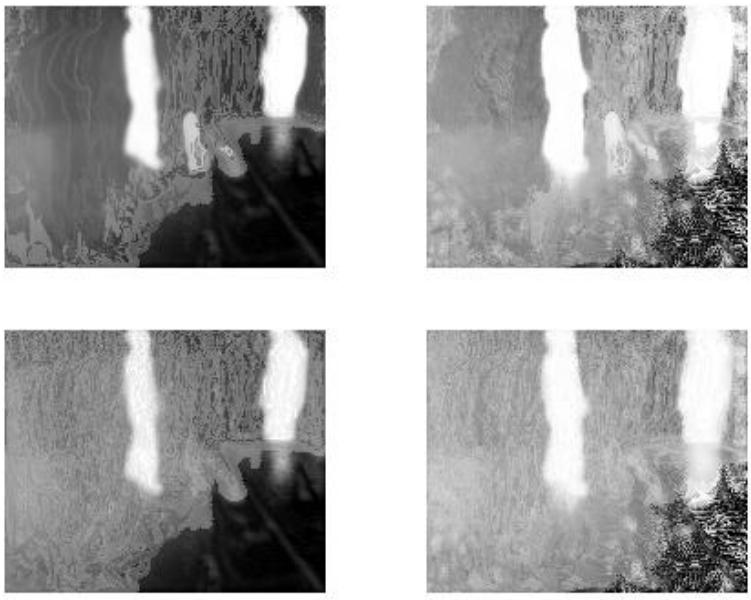

Figure 3.6: Here it is possible to compare some results regarding the MOG algorithm implemented. Above 3 gaussians have been used and 5 below. On the left the learning rate α is 0.01 and on the right 0.02.

dimensional space. In the paragraph 3.7 it is possible to compare the results coming from just using the depth or the grey-scale information and also both by considering the depth and the grey-scale levels dependent, as shown in this paragraph, and by considering them independent, i.e. just multiplying the two probability calculated independently.

As it is possible to see in the figure 3.6 the best results are obtained with a small learning rate α (0.01) both using a mixture of three gaussians and using five. The learning rate regulates the updating speed of the background model, i.e. the speed of the gaussians growth. If in the sequence the blobs move quickly it is better to use a bigger learning rate, for instance 0.02 or greater, otherwise a lower one.

The choice of the learning rate depends on the application. In case of people tracking an α of 0.01 is enough, otherwise if the scene to monitor had been a street on which cars pass a greater learning rate would have been necessary, because cars speed is much greater than people's one. The size of α influences also the amount of wake that a person leaves behind him. Of course if the learning rate is high the background will be updated more quickly and the person will impress much more the background model leaving more wake, otherwise the blobs do not modify significantly the background and it means that to change the background model it takes more time. It is also possible to argue that it is better to generate the background and not to allow to the foreground to modify it. This choice depends on the application to implement, but in the main the background has to adapt to the environment and to change according to the light changes or the objects moved in the scene.

Regarding to the number of gaussians for an indoor use three are enough as it is possible to understand by comparing the results using 3 normal functions and five. Using less than three gaussians all the advantages given by this method would be lost because an outlier value given by the noise could delete the most important gaussians for the current pixel.

Even if the results in term of probability are quite different, the background model is generated correctly by using those two different learning rates, as it is shown in figure 3.7.

3.4.3 Background Model Estimation

To build the background model some of the distributions, for each pixel, must be chosen. It is possible to argue that the values belonging to the background are the more persistent, and for that reason they might belong to those distributions

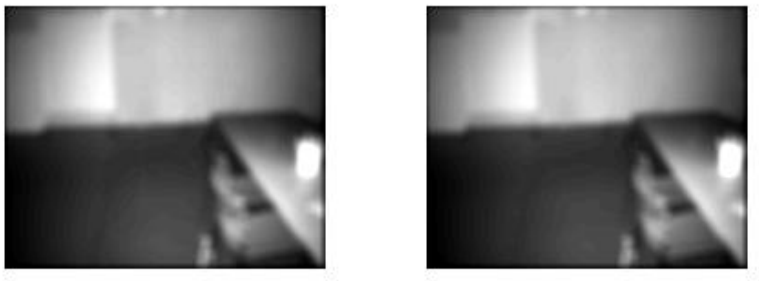

Figure 3.7: These are the background models generated by the Mog algorithm using 4 gaussians and a learning rate equal to 0.01 (left) and 0.02 (right).

that have a large weight and a small variance. In fact it could be argued that the values belonging to a moving object can introduce new gaussians, but as their effect is temporary, there is not time for those distributions to huge their weights and decrease their variances as it happens for the background ones.

At this point it must be decided the portion of the distributions that we can consider as background. To do that the distributions are kept ordered according to their weight and just the ones satisfying the following equation can enter in the background model:

$$B = argmin_b(\sum_{k=1}^{b} \omega_k > T) \tag{3.14}$$

It means that the first B distributions whose sum of normalized weights is greater than a threshold T are considered belonging to the background.

3.5 KDE for the TOF camera

A kernel density estimator belongs to a class of estimators called non-parametric density estimators. Unlike the parametric estimators where the estimator has a fixed functional structure and the parameters of this function are the only information we need to store, non-parametric estimators have no fixed structure and depend upon all the data points to reach an estimate.

Unlike the mixture of gaussians the one proposed by Elgammal et al. [2] is non-parametric and for each pixel the classification depends on the values that the

pixel has had in the last N positions. The idea behind it to build for each pixel a histogram and, according to it, give to the current pixel value a probability to belong to the foreground.

Let $x_1, x_2, ..., x_N$ be the last N values for a pixel. The probability that a pixel has value x_t at time t can be estimated with a kernel K:

$$Pr(x_t) = \frac{1}{N} \sum_{i=1}^{N} K(x_t - x_i) \tag{3.15}$$

Choosing the kernel function as a normal function it is possible to rewrite the equation as:

$$Pr(x_t) = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{(2\pi)^{\frac{d}{2}} |\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(x_t - x_i)^T \Sigma^{-1} (x_t - x_i)} \tag{3.16}$$

and if the independence between the color channels is assumed the matrix Σ becomes diagonal and the previous equation is reduced to:

$$Pr(x_t) = \frac{1}{N} \sum_{i=1}^{N} \prod_{j=1}^{d} \frac{1}{\sqrt{2\pi\sigma_j^2}} e^{-\frac{1}{2}\frac{(x_t^j - x_i^j)^2}{\sigma_j^2}} \tag{3.17}$$

where d is the number of the color channels and j is its index.

As in the mixture of gaussians model for the time-of-flight camera explained in the section 3.4, the values of the grey-scale and the depth are not considered independent and the distances between the samples are calculated in the same bi-dimensional space.

3.6 Maximum-value for the TOF camera

This method is much simpler than the other two described above. To build the background model the last N frames are considered and for each pixel the histogram is built, by dividing all the possible values into q bins. According to the histogram for each pixel the maximum value is taken as the background value and a probability to belong to the foreground is given to each pixel according to the difference between the current and background pixel depth and brightness.

$$Pr_i = 1 - \frac{\sqrt{(I_i - I_{back})^2 + (Depth_i - Depth_{back})^2}}{q} \tag{3.18}$$

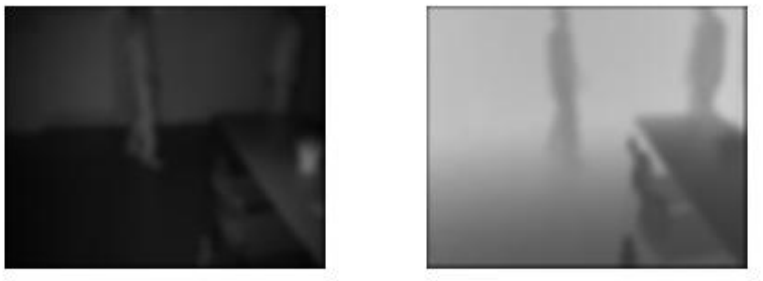

Figure 3.8: Example taken with the SwissRanger. On the left the grey-scale image and on the left the depth one.

where I_i and $Depth_i$ are the brightness and the depth value of the current pixel i, I_{back} and $Depth_{back}$ are the brightness and the depth of the background model for that pixel and q is the number of bins considered, in this way the value of the probability is between 0 and 1.

3.7 Experimental results

Before implementing the detection and the tracking of people the Background Subtraction has been tested to compare the different methods and decide which one could be used for the following work. In the figure 3.8 a frame of the sequence considered in this example is shown and it is possible to see the gray-scale and the depth images.

After having implemented the KDE, mixture of gaussians and maximum-value background subtraction methods they have been tested also on the sequence from which the frames in figure 3.8 are taken. In figure 3.9 it is possible to see the probabilities map generated by the methods. For the kernel density estimation method it is possible to vary the width of the window, i.e. the number of following frames used to generate the background model. Experiments have been performed varying this parameter from 10 up to 200. The more this parameter is great and the more the background model is accurate, but of course the algorithm becomes more slow. A window of 100 frames can generate good performance and it is a good compromise for the memory occupation. Regarding the mixture of gaussians the learning rate has been taken equal to 0.01 because of the results obtained in the section .

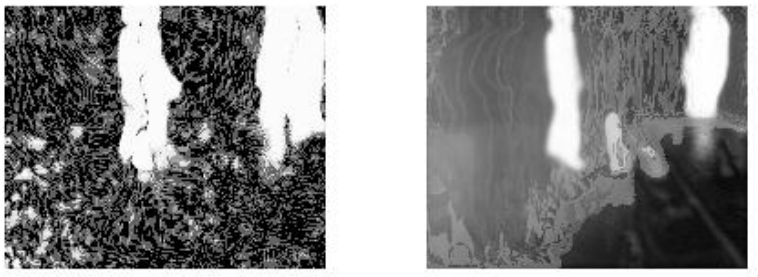

Figure 3.9: Probabilities map generated by the KDE algorithm (on the right) and by the Mixture of Gaussians (on the left).

As we can see the MOG (with $\alpha = 0.01$) is more accurate than the KDE (with a window of 100 frames) for the SwissRanger and generates a more accurate separation between the foreground and the background. As a matter of fact in the KDE there is much more noise than in the MOG. In the KDE method all the N values of the window are equally considered when the probability to belong to the foreground is calculated and the all outliers generated by the noise contribute to this calculation in the same way of the background values. Whereas in the MOG method when an outlier comes it can just generate the last gaussian and its effect will disappear in the following iterations when other values will take its place. As it is an outlier it cannot generate important gaussians, for which many more values are needed, otherwise it would not be an outlier. Therefore if the images are a bit noisy as the ones taken with the SwissRanger, the MOG method performs better because it is able to "hide" the outliers for the following frames.

In figure 3.10 there are the background models generated by the methods. As we can see the three background models are quite good and correspond to the real background. The third one is a bit worse because the values are sampled when the histograms are built for the calculation of the maximum value.

To appreciate the advantage of using the brightness and the depth information together also other experiments have been performed. In this experiment the Mixture of Gaussians algorithm has been used to extract the foreground just using the brightness, the depth and both of them together, by considering them independent or dependent. If the two types of information are considered independent the probability to belong to the foreground is just the product of the two probability calculated using the depth and the brightness separately. Otherwise if they are considered dependent the probability is calculated as shown in the paragraph 3.4, by measuring the distances of the samples in a two-dimensional

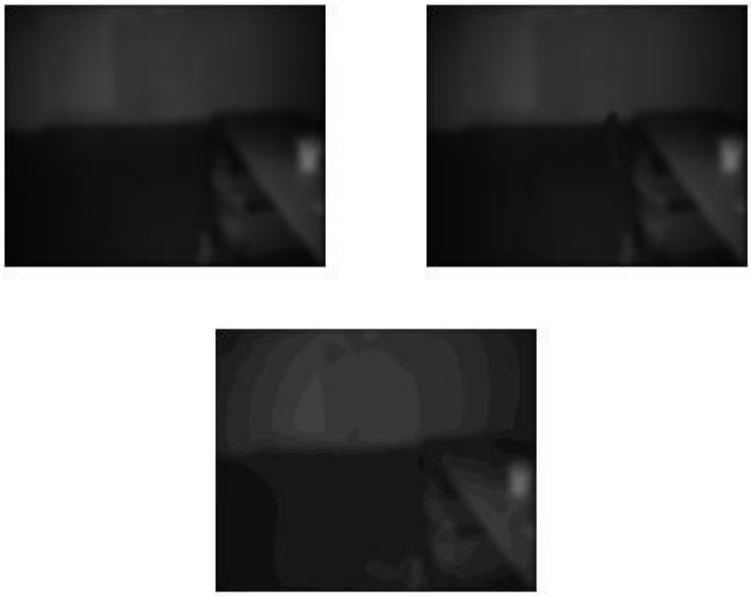

Figure 3.10: Background models generated by the KDE (left), MOG (right), MAX (below).

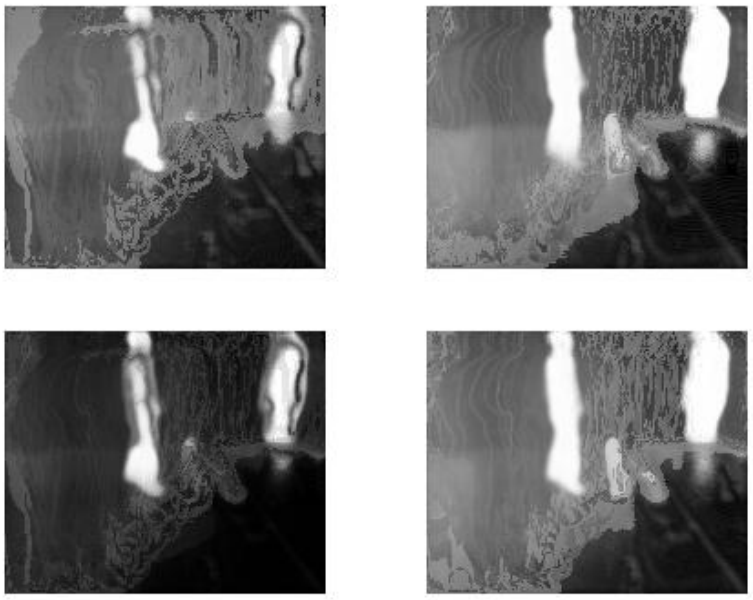

Figure 3.11: Above there are the probabilities generated just using the brightness (left) and the depth (right). Below on the left there is the probability map generated considering the brightness and the depth as dependent, whereas on the right the one calculated considering them as dependent.

space.

As we can see in figure 3.11 the use of the depth information improves the results in a sensible way and this can make easier the coming separation between the foreground and the background.

The results shown in this paragraph have been taken using an integration time equal to 100 and a modulation frequency of 20 MHz. The reason of these values comes from the considerations of the second chapter.

3.7.1 Reflection problems

As shown in the chapter 2 if the camera is not mounted in the right way or if the scene is too close to it according to the current integration time, it is possible to experience reflection problems. In figure 3.12 two frames are shown. On the left

there is a moving person enough far from the camera not to produce reflections and on the left the same case, but with the person too close for the current integration time and for that reason the images are very noisy and the resulting probability map of course is wrong. For both the examples the grey-scale, the depth and the resulting probability map are shown.

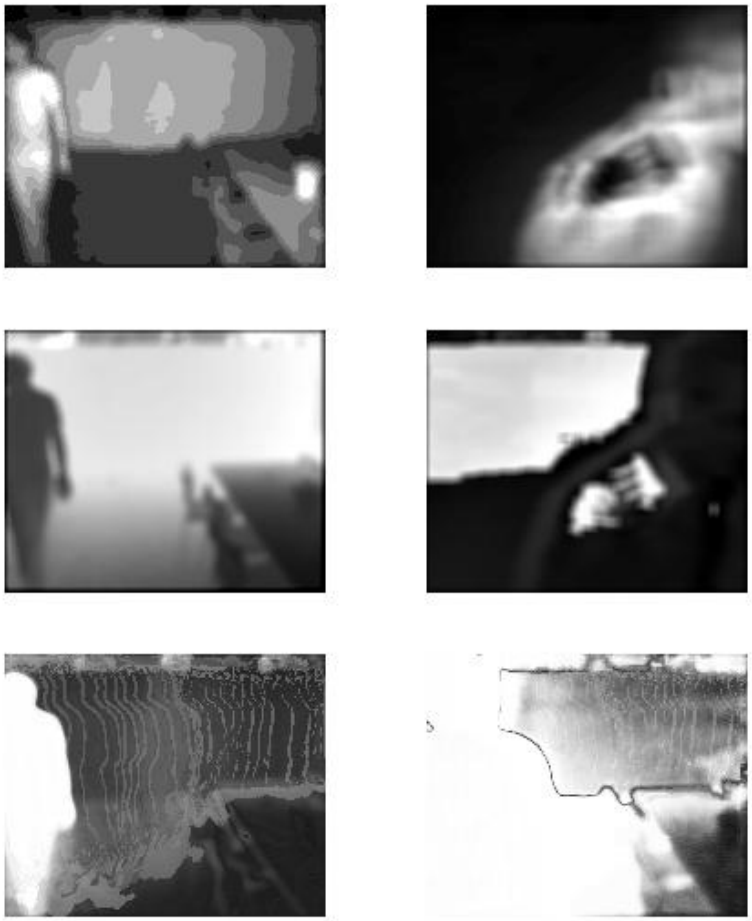

Figure 3.12: A comparison between a scene taken by the camera on a right way and, on the right, a case in which the background subtraction is not correctly performed because of reflection problems. For each sequence we can see the grey-scale, the depth an the probability map.

Chapter 4

Detection and tracking

In the third chapter we have seen how it is possible to associate to each pixel a measure of the probability that a pixel belongs to the foreground. The next step is to use this information to extract the blobs representing the humans and the non-humans. The easiest way to perform that is to threshold these probabilities and obtain the foreground blobs by searching the connected components. Besides to be hard to compute for a real time system, this method is also very sensitive to the threshold, that is difficult to choose because it might depend also on the particular conditions of the environment.

For these reasons the detection of the blobs has been performed by a method inspired by the generative-model-based tracking introduced by Pece [11], where the foreground is modeled with a two-dimensional gaussian distribution updated with the EM algorithm.

In the following paragraphs this method will be presented in more detail.

4.1 Statistical model

The model is a mixture of clusters: n clusters belonging to the foreground and one representing the background. In this way it is unnecessary to threshold

the probability image since the background is considered as a cluster. The background cluster has index 0 and the others $j > 0$. Besides each cluster has a set of parameters, whose updating is performed by the EM algorithm and indicated by λ_j. The set λ_j includes the prior probability w_j of generating a pixel, the average μ_j of the probability image for this cluster, the centroid c_j and the covariance matrix Σ_j. All these parameters will be more clear in the next sections.

The probability that the cluster j generates a pixel value at the location u can be split in two components:

$$f_j(u) = g(u|\lambda_j) \cdot h[\delta(u)|\lambda_j] \tag{4.1}$$

where g depends on the coordinates of the image and h on the gray-level difference observed at that location. Instead of using the differences between consecutive frames the probabilities to belong to the foreground have been used. In this way the extraction of the blobs is more accurate as those probabilities consider also the past history of the sequence and not just the previous frame.

4.1.1 Background cluster

For the background cluster the probability $f_0(u)$ depends only on the probability value, since the background is behind the whole scene, at every pixel location. For that reason the function g is constant:

$$g(u|\lambda_j) = \frac{1}{m} \tag{4.2}$$

where m is the number of pixel of the image.

The background depends on the values of the probabilities and the more they are high and the less it is likely that that pixel belongs to the background.

$$h[pr(u)|\lambda_0] = \frac{1}{2\mu_0} \exp(-\frac{|pr(u)|}{\mu_0}) \tag{4.3}$$

where μ_0 is the absolute value of the mean for the gray-scale values of the cluster.

4.1.2 Target clusters

For the target clusters the function h is considered a uniform distribution:

$$h[pr(u)|\lambda_0] = \frac{1}{q} \tag{4.4}$$

where q is the number of observable probability values. Besides the distribution g is considered normal and depends on the distance between the pixel and the cluster centroid:

$$g(u|\lambda_j) = \frac{1}{2\pi\sqrt{|\Sigma_j|}} \exp(-\frac{1}{2}\Delta u_j^T \cdot \Sigma_j^{-1} \cdot \Delta u_j) \tag{4.5}$$

where $\Delta u_j = u - c_j$ is the vector distance between the pixel and the centroid of the cluster j and Σ_j is the covariance matrix of the cluster. In this way each cluster has an ellipsoidal shape, regulated by the covariance matrix.

For each pixel the posterior probability that it belongs to the cluster j is given by:

$$p_j(u) = \frac{w_j f_j(u)}{f(u)} \tag{4.6}$$

where w_j is the weight (prior probability that a cluster generates a pixel) of the cluster, i.e. the portion of the image occupied by the cluster, and $f(u)$ the prior probability of the pixel:

$$f(u) = \sum_{j=0}^{n} w_j f_j(u) \tag{4.7}$$

During the updating steps, shown in the following paragraphs, the ellipses associated to the clusters are updated using the EM algorithm to fit the blobs in the foreground, according to the values of f and g.

4.2 Likelihood and parameters estimation

The probabilities shown in the previous paragraph can be computed from the parameters $\lambda_j = (w_j, \mu_j, c_j, \Sigma_j)$, that are estimated using the EM algorithm by maximizing their log-likelihood. This function is the logarithm of the probability of the probability map generated after the background subtraction:

$$L(\lambda_j|D) = \log \prod_u f(u) = \sum_u \log \sum_j w_j f_j(u) \tag{4.8}$$

where D is the probability map. Instead of maximizing this function the log-probabilities can be weighted with the clusters posterior probabilities:

$$\hat{L}(\lambda_j|D) = \sum_u \sum_j p_j(u) \log(w_j f_j(u)) \tag{4.9}$$

and it is possible to divide the function $\hat{L}(\lambda_j|D)$ according to the contribution $\hat{e}_j$ of each cluster j:

$$\hat{e}_j = \sum_u p_j(u) \log(w_j f_j(u)) \tag{4.10}$$

This partitioning of the objective function will be used to estimate the number of clusters. In fact with this quantity it is possible to estimate the variation of the objective function if all the pixels of the cluster j are assigned to the cluster k without changing its parameters:

$$M(j,k) = \sum_u p_j(u) \log \frac{w_k f_k(u)}{w_j f_j(u)} \tag{4.11}$$

and if it is greater than a threshold

$$M(j,k) > -\theta_M \tag{4.12}$$

the clusters can be merged.

The expectation-maximization algorithm alternates between performing an *expectation* step, which computes an expectation of the likelihood by including the latent variables as if they were observed, and a *maximization* step, which computes the maximum likelihood estimates of the parameters by maximizing the expected likelihood found on the expectation step. The parameters found on the maximization step are then used to begin another expectation step, and the process is repeated until when the parameters reach a convergence.

The updated estimate of the cluster average, for example, can be calculated as:

$$\mu_j^{k+1} = \frac{\sum_u |pr(u)| \cdot p_j^{(k)}(u)}{\sum_u \cdot p_j^{(k)}(u)} \tag{4.13}$$

where k indicates the k-th iteration. For the covariance matrices the ML estimate ($\hat{\Sigma}$) is weighted with a factor $1/\tau_\sigma$:

$$\Sigma^{(f,i)} = \Sigma^{(f-1,\infty)} + \frac{1}{\tau_\sigma}(\hat{\Sigma}^{(f,i)} - \Sigma^{(f-1,\infty)}) \tag{4.14}$$

where f is the index of the frame, i the number of the iteration and ∞ the index of the last iteration in the previous frame. Besides the ML estimate has been calculated as:

$$\hat{\Sigma}^{(f,i)} = \frac{\sum_u (u-c)(u-c)^T \cdot p^i(u)}{\sum_u p^i(u)} \tag{4.15}$$

where $p^i(u)$ is the posterior probability at the i-th iteration and c the centroid of the cluster.

At the beginning, when a new cluster is detected, Σ is initialized with an initial guess that corresponds to a small round blob. After this blob can grow according to the values of $p^i(u)$. In fact if a pixel u is close to the blob i and it has an high probability to belong to the foreground it means that $f_i(u)$ will be greater than $f_0(u)$ and thereby the posterior probability $p_i(u)$ will have an high value, so as to allow the blob i to grow and cover also the pixel u. Otherwise if this pixel had had an high probability to belong to the background the value of $f_0(u)$ would have been greater and the posterior probability $p_i(u)$ would not have been enough to allow the blob to enlarge.

Since the estimate of the background cluster parameters are hard to compute and they are not significantly affected by the changes of the other clusters, it is enough to update them only one time, after the convergence of the EM algorithm. Besides the initial estimates of the centroids and the covariance matrices at a given frame are the parameters of the previous frame after the convergence. If the clusters are well separated the convergence requires less than 10 iterations of the algorithm.

4.3 The algorithm

The following table lists the cluster parameters:

Table 4.1: Cluster parameters

symbol	*parameter*
w	prior probability of generating a pixel
μ	average of the probability image
c	centroid
Σ	covariance matrix

These parameters are kept updated using the EM algorithm shown in the previous paragraph and are used to divide the foreground in clusters.

The first step of the algorithm is to detect new clusters according to the information coming from the previous frame and their parameters are updated using the EM algorithm, with which the ellipses, whose shape is regulated by the covariance matrix, are built and tracked. After having found new ellipses and their parameters updated, the clusters are analyzed and according to their parameters they could be deleted, merged or split. If no change is performed the iterations stop, otherwise the EM algorithm is performed again.

4.3.1 Detecting of new clusters

New clusters are detected by locating the local maxima in the probability image, counting the clusters already present in the previous frame. To do this the probability image is weighted by the probability to belong to the background. In this way the local maxima are only searched on the background, without detecting again clusters already found:

$$pr_0(u) = pr(u)p_0^{(t,0)}(u) \tag{4.16}$$

where $p_0^{(t,0)}(u)$ is an estimate of $p_0(u)$, obtained in the last iteration of the previous frame. After that the image is smoothed and down-sampled to obtain $\hat{pr}_0(u)$, on which the local maxima are detected.

Not all the local maxima are taken as the centers of the new clusters, but just the ones greater than a threshold:

$$\hat{pr}_0(u) > \mu_0(1 + \log\frac{q}{2\mu_0}) \tag{4.17}$$

where μ_0 is the background average and q the number of possible values that the probability image can assume.

The equation 4.17 is motivated by the merging cost of one cluster into another as shown by the equation 4.11. If the cluster in which the other one is merged is the background it is possible to assume that its values are not much modified as it contains many more pixels than the other clusters and this cost can be written as:

$$M(j,0) = mw_j\left[\log\frac{\zeta_0}{\zeta_j} + \log\frac{q}{2\mu_0} - \frac{\mu_j}{\mu_0} + 1\right] \tag{4.18}$$

where

$$\zeta_j = \frac{mw_j}{2\pi\sqrt{|\Sigma_j|}} \tag{4.19}$$

is the estimated pixel density inside the blob, that for the background cluster can be taken as:

$$\zeta_0 = w_0 \tag{4.20}$$

Combining the equations 4.12 and 4.18 it is possible to write:

$$\frac{\mu_j}{\mu_0} \geq 1 + \log\frac{q}{2\mu_0} + \log\frac{\zeta_0}{\zeta_j} + \frac{\theta_M}{mw_j} \tag{4.21}$$

and neglecting the last two terms the 4.17 is obtained.

It is interesting to emphasize that to detect new clusters no threshold value is needed, because the minimum ratio μ_j/μ_0 to generate a cluster can be expressed as $1 + \log\frac{q}{2\mu_0}$.

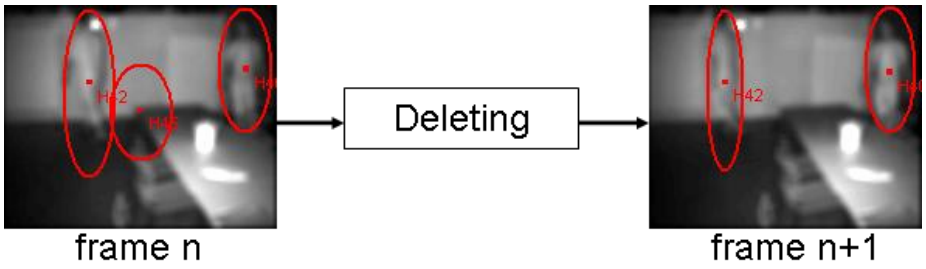

Figure 4.1: In this example it is possible to see a false detection probably due to the shadow of the walking person. However this blob is deleted since its pixels are similar to the background ones.

4.3.2 Eliminating clusters

The criteria to eliminate compare the average of the blob with the average of the background and the dimensions of the ellipse:

1. the average absolute value of the probability image for the cluster j is smaller of the background average multiplied by a constant κ_μ

$$\mu_j < \kappa_\mu \mu_0 \tag{4.22}$$

2. The weight (prior probability) of the cluster, w_j, is less than L^2/m, where L is the cell size used to down-sample the image during the detection of new clusters and m is the dimension of the image.

One cluster is eliminated if at least one of these conditions is satisfied. This test is performed at each iteration of the EM algorithm because the method has a complexity linear with the number of clusters and for that reason it is convenient to remove clusters as soon as possible.

4.3.3 Merging clusters

Two clusters are merged if they are enough close to each other and the resulting cluster has an ellipsoidal shape:

1. The centroinds of the two clusters i and j must be closer than a given

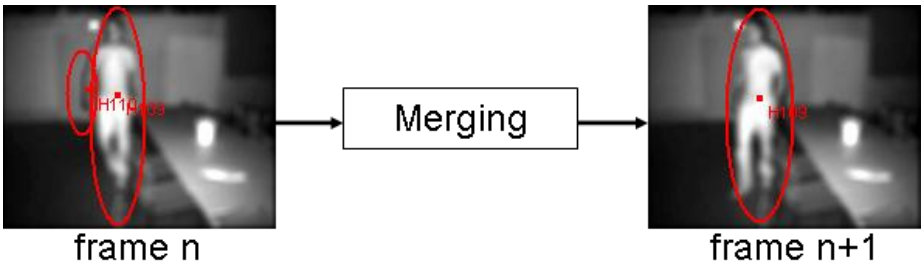

Figure 4.2: On the left picture even if there is one person there are two blobs, because the hand is detected as a separate cluster.

Mahalanobis distance:

$$\sqrt{\Delta c_{ij}^T \cdot \Sigma_i^{-1} \cdot \Delta c_{ij}} < \kappa_M \quad \vee \quad \sqrt{\Delta c_{ij}^T \cdot \Sigma_j^{-1} \cdot \Delta c_{ij}} < \kappa_M \tag{4.23}$$

where $\Delta c_{ij} = c_i - c_j$ and κ_M is a constant, that can be taken as 2.5.

2. The clusters i and j have approximately the same width in the direction orthogonal to the line connecting the centroids:

$$\kappa_T^{-1} < \frac{s_i}{s_j} < \kappa_T \tag{4.24}$$

with

$$s_i^2 = \frac{1}{\|\Delta c_{ij}\|^2} \Delta c_{ij}^T \cdot R^T \cdot \Sigma_i^{-1} \cdot R \cdot \Delta c_{ij} \tag{4.25}$$

where R is a 2×2 matrix whose aim is to rotate Δc_{ij} by 90 degrees. This condition ensures that the merging is performed between ellipses having the same direction avoiding to generate blob with a T-shape.

3. The depth averages of the two clusters are close to one another; in this way it is possible to avoid the merging of two people moving in two different depth levels, whose blobs become close in the camera scene:

$$\left| \frac{\sum_u Depth(u) \cdot p_i(u)}{\sum_u p_i(u)} - \frac{\sum_u Depth(u) \cdot p_j(u)}{\sum_u p_j(u)} \right| < \kappa_D \tag{4.26}$$

The merging between the clusters i and j is performed if all the three conditions are satisfied.

4.3.4 Splitting clusters

The expected density for a foreground blob is approximated by an ellipse. If the blob has not this shape it is likely that it is composed by two different objects and for that reason the blob must be split into two different ellipses so that the two resulting blobs are ellipsoids. This is performed by dividing the blob in 9 parts orthogonally to the main axes. For each of these 9 segments the squared deviation is calculated and normalized to obtain a χ^2 measure:

$$\frac{(observed - expected)^2}{expected} \tag{4.27}$$

where the observed and expected density for each section are used. The blob is split if the lower negative deviation is smaller than a threshold θ_χ and the splitting is performed at the position of this section.

If these conditions are satisfied the probability that this cluster is a human increases and now it would be suitable that the ellipse fits as good as possible the blob that for the moment is considered to be a person

4.4 Algorithm execution

In figure 4.3 there are the main steps of the algorithm. In the first step the initial centroids of the blobs are detected by finding the local maxima in matrix that is product between the probability map, coming from the background subtraction, and the background posterior probability to avoid to find blobs already detected. Not all the local maxima are taken as new centroids, but just the ones greater than $\mu_0(1 + \log \frac{q}{2\mu_0})$. In this way the centroids are chosen according to the background average and no thresholding is needed. If, like in this case, the blobs are new for the algorithm then their covariance matrix is initialized in this way:

$$\Sigma = \begin{pmatrix} 100 & 0 \\ 0 & 100 \end{pmatrix}$$

As it is possible to see the initialization of the covariance matrices does not use any prior assumptions, in fact it has just the shape of a circle.

Now all the parameters are updated with the EM algorithm and it stops when the convergence is reached, i.e. when the centroids do not move more than a small constant ϵ.

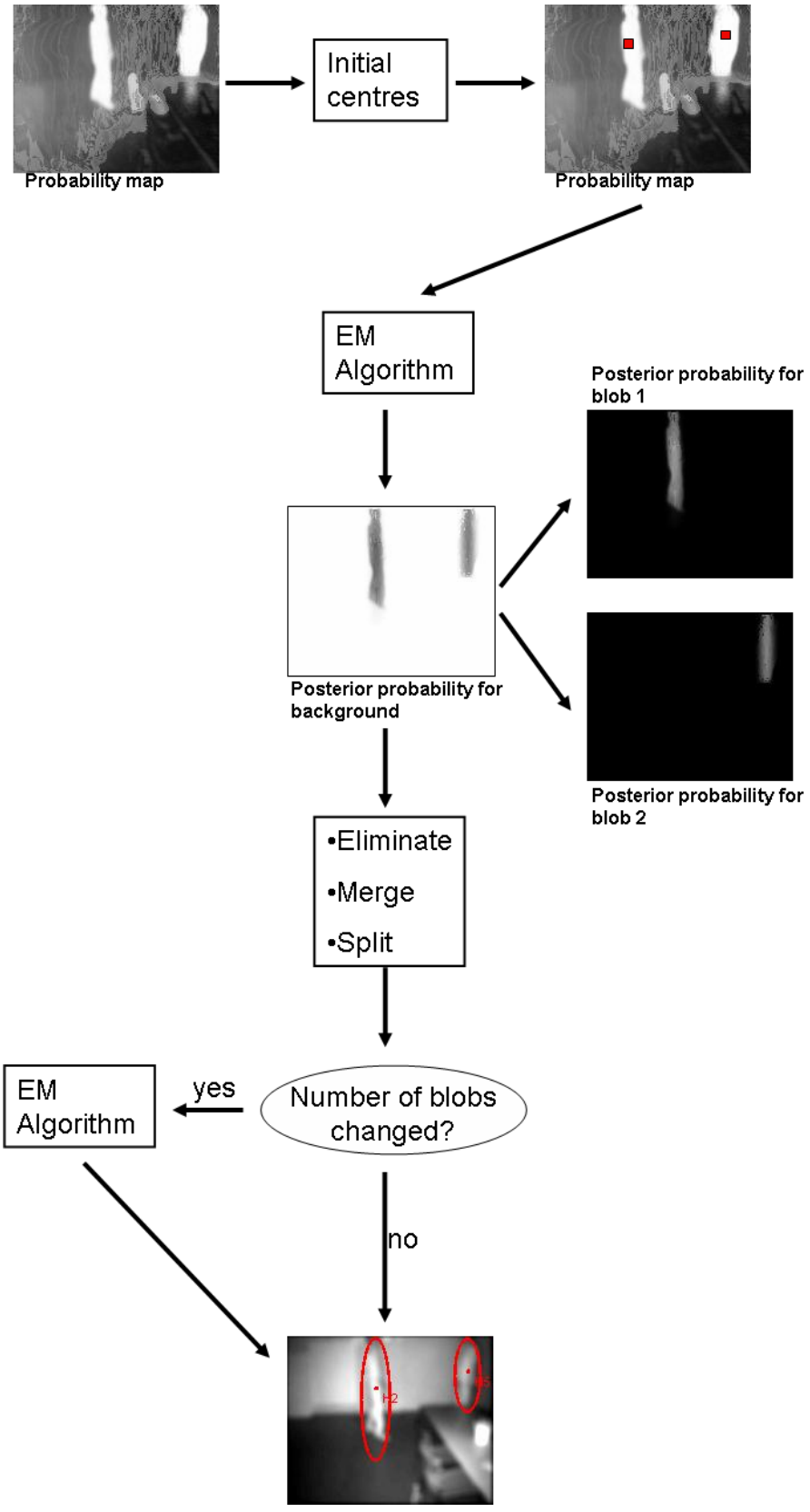

Figure 4.3: In this scheme the principal steps of the algorithm are shown.

At this point the algorithm checks the conditions to eliminate, merge and split blobs. If the blobs are merged together the resulting blob is considered be successor of the one with the greater weight (prior probability); besides the resulting centroid and covariance matrix come from a weighted sum of the two predecessors, where the weights are still the prior probabilities.

If after this last step the blobs remain the same, i.e. no blob has been merged, deleted or split, the algorithm can conclude for the current frame, otherwise the EM algorithm must be executed another time to update the blobs parameters.

Chapter 5

Blob classification

When new clusters are detected it is necessary to divide them between the ones representing humans and the non-human ones. To perform that the algorithm checks both the ellipse dimensions and orientation and the content inside it by comparing the shape of the object (or human) with a data set of positive and negative examples through a object-recognition algorithm inspired by the Viola-Jones method.

5.1 Cluster dimensions

A cluster containing a moving person has a very high likelihood to be inside an ellipse whose shape is long having a vertical orientation.

This can be easily checked by looking at the covariance matrix of the ellipse that directly influences the orientation and the dimensions of the ellipse. As a matter of fact the height of the ellipse depends on the value of a_{11} and the width depends on a_{22}.

$$\Sigma = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$$

These correspondences can be proved by looking at the ML estimate of the

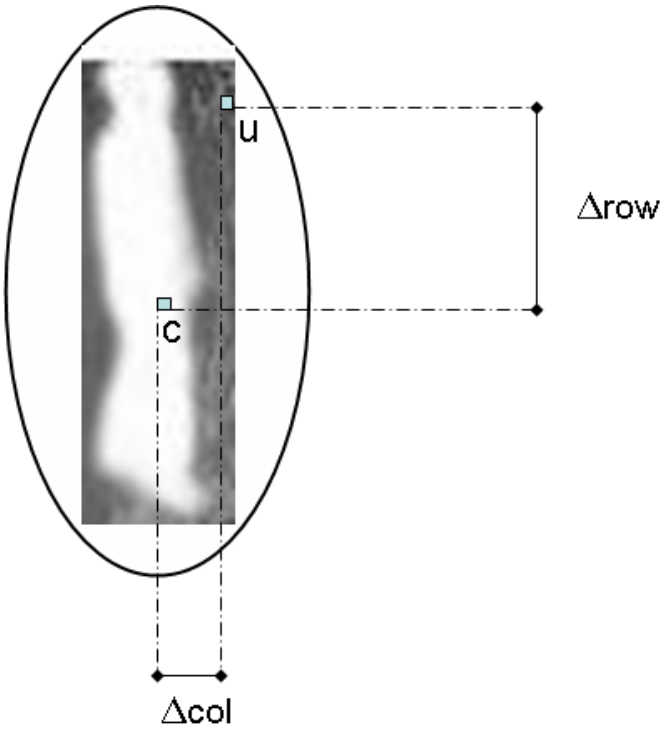

Figure 5.1: When Σ is calculated the distances between all the points u weighted by their probabilities are considered.

covariance matrix shown in the last chapter:

$$\hat{\Sigma}^{(f,i)} = \frac{\sum_u (u-c)(u-c)^T \cdot p^i(u)}{\sum_u p^i(u)} \tag{5.1}$$

From this equation it is easy to see that the matrix $(u-c)(u-c)^T$ can be written as:

$$(u-c)(u-c)^T = \begin{pmatrix} (row_u - row_c) & (col_u - col_c) \end{pmatrix} \begin{pmatrix} (row_u - row_c) \\ (col_u - col_c) \end{pmatrix} =$$

$$= \begin{pmatrix} (row_u - row_c)^2 & (row_u - row_c)(col_u - col_c) \\ (row_u - row_c)(col_u - col_c) & (col_u - col_c)^2 \end{pmatrix}$$

where c is the centroid of the cluster, u is the current pixel and row and col are the row and column values. As it is possible to see the value of a_{11} depends

on the differences of the row values (i.e. the height of the blob) and a_{22} on the column differences (i.e. the width of the blob). For that reason the values of the covariance matrix can be used to estimate the blob dimensions.
In particular the ratio between a_{11} and a_{22} is considered and if it is greater than 2 the blob has more probability to be human and the next step, human recognition, is performed.

5.2 Human recognition

If the blob has vertical orientation and the height is at least two times than the width, then probably the blob is a moving person. To divide the blobs into *humans* and *non-humans* at the beginning an object recognition system based on Viola-Jones method was developed. The algorithm was trained with a set of positive (human bodies) and negative (just background images) examples, to be able to recognize which ones are the blobs representing people.

Later this step has been modified to exploit this information to improve the performance of the tracking algorithm. As a matter of fact the comparison between the blob and a mask similar to the ones generated by the Viola-Jones training algorithm can be used not only to state if the blob is a human, but also to improve the tracking by resizing the ellipse around the blob according to the value obtained by multiplying the rectangular matrix. Of course for the algorithm modifying the ellipse size means modifying the related covariance matrix. In figure 5.2 it is possible to see the mask used to weight the rectangular matrix around the ellipse. The shape is divided in three regions to recognize a standing up person and for each region one matrix composed by ones and zeros has been defined to maximize the value on the head, the body and the legs.

If the product between the blob and this mask is enough high, then there is an high likelihood that the blob is a human.

The product is performed pixel by pixel. According to the model showed in figure 5.2 a mask of the same size of the blob is created and for each pixel in the cluster the probability to belong to the foreground is multiplied by the mask value and all these values are summed and normalized for the number of pixels that the blob has.
Experimental tests have shown that a good threshold to divide human from non-human blobs is 1.5.

Besides, as explained in the next section, this product is performed also resizing the blob, enlarging and reducing its size to find the best growth for the ellipse.

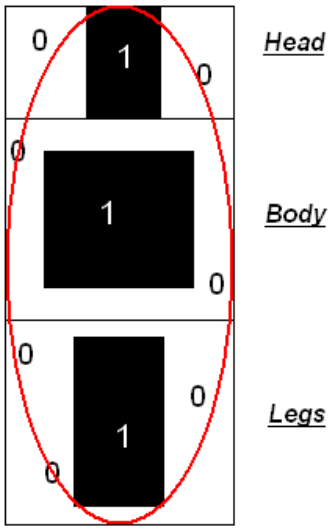

Figure 5.2: Mask used to state if the blob is a human or it is not.

5.3 Improving the performance

The mask shown in figure 5.2 is not only used to have a measure of *human similarity* for the current blob, but it is also used to resize the ellipse to improve as much as possible the capacity of the ellipse to surround the foreground blob. This is made by enlarging and reducing the ellipse and calculating the product between these resized blobs and a mask of the same size.

This optimization step ends when the dimension that maximize this mask is found and for the same considerations made in the paragraph 5.1 the covariance matrix is resized according to the output of this step; i.e. if the blob height is enlarged or reduced then the value a_{11} is enlarged or reduced proportionally and if the width is enlarged or reduced then a_{22} is enlarged or reduced.

5.4 Final classification

To divide the blobs into *humans* and *non-humans* each cluster is given a probability to be human. At the beginning, when a new cluster is detected, this probability is set to 0.5 and at each frame the blobs dimensions are measured and its shape is optimized as explained in the previous paragraphs. If the blob dimensions, orientation and the product with the mask satisfy the criteria shown previously then the probability to be human is increased by 0.1, otherwise decreased. In this way, instead of a simple classification frame by frame, also the past history is taken into account.

The main advantage of this approach is the stability in the tracking process. As a matter of fact if a person is classified as human for q following frames, but in the frame $q+1$ the algorithm fails, for example because of the person position, the cluster can still be considered human because of its probability accumulated during the previous frames.

In figure 5.3 the steps of the algorithm described in this chapter are shown graphically. At the beginning the shape of the ellipse is resized according to the current mask, the new dimensions and the multiplication value are checked and according to this the probability to be a human blob is increased or decreased.

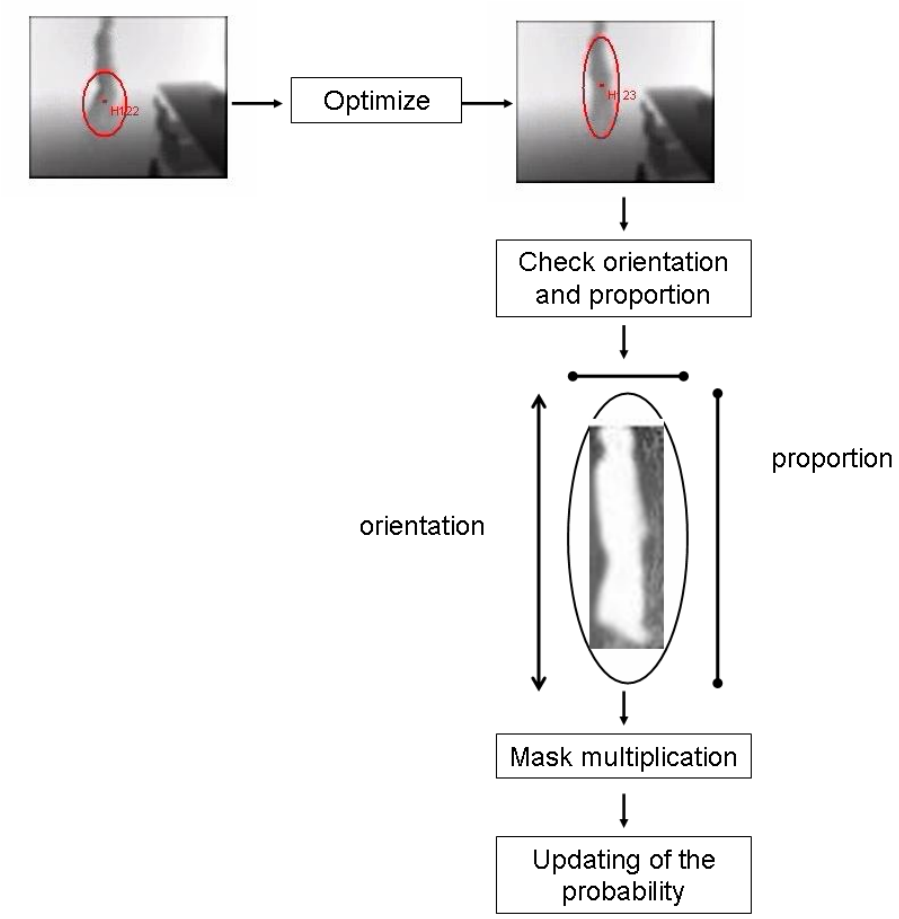

Figure 5.3: In this scheme there are the steps needed to perform the human recognition.

Chapter 6

Experimental results

6.1 Results

All the code implementing the background subtraction, the detection and tracking of moving people has been developed in Matlab, as the drivers to acquire the images from the camera and store them as Matlab matrix are available. Due to the memory occupation it has not been possible to take very long sequences, but this is not a big limitation if this algorithm will be used as a real time application and it will need to store into the RAM just a piece of the past history.

The sequences considered to test the algorithm have been taken at different times of the day and with different camera parameters, also considering the extreme situations, for example when the modulation frequency does not allow to detect in a correct way the background of the scene, as shown in the second chapter.

For each test sequence the number of detected people will be compared with the correct one and some of the most relevant frames will be shown, especially the cases in which the algorithm fails, and for these ones also an explanation of the causes will be provided, and also the cases in which there are some critical passages.

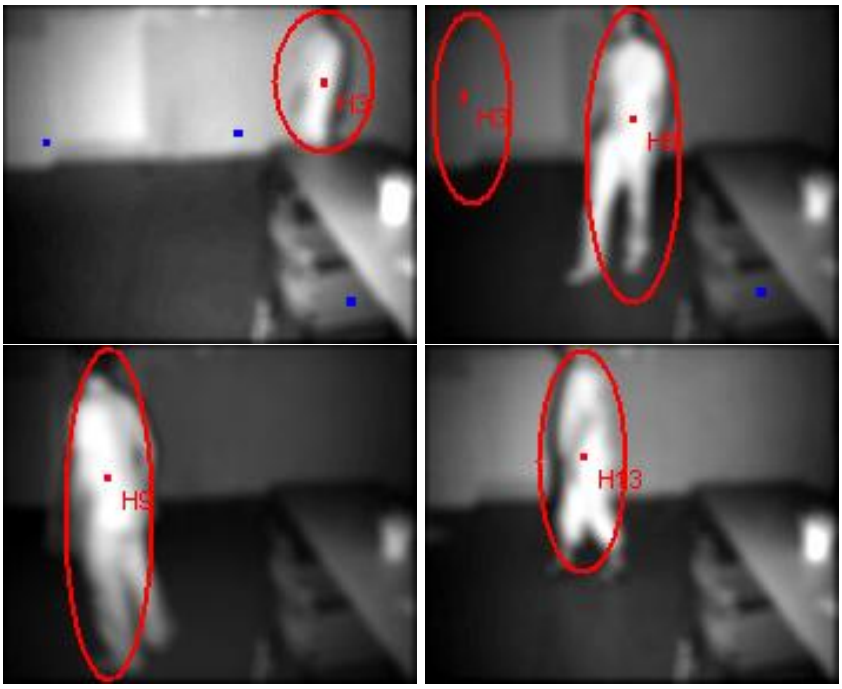

Figure 6.1: From frame 1 to 57 there is one person moving and the algorithm detects 1.09 people. This is caused by some false detections due to the not clear background model, that at the beginning has to steady. In fact the normal functions need some time before having a small variance and produce a more clear separation between the background and the foreground.

6.2 Test sequence one

The first test has been taken with a modulation frequency of 20 MHz, setting the auto-illumination and with an integration time of 100, that corresponds to 20,2 *ms*. In this way the wall in the background, that is less than ten meters far away from the camera can be correctly detected without producing overflow in the depth measure.

In this sequence there are two people walking, that sometimes stop, talk to each other and interact with the object in the room, like moving chairs or throw a pillow.

Below the sequence will be analyzed bit by bit and for each part the number of people detected will be shown and compared with the real one.

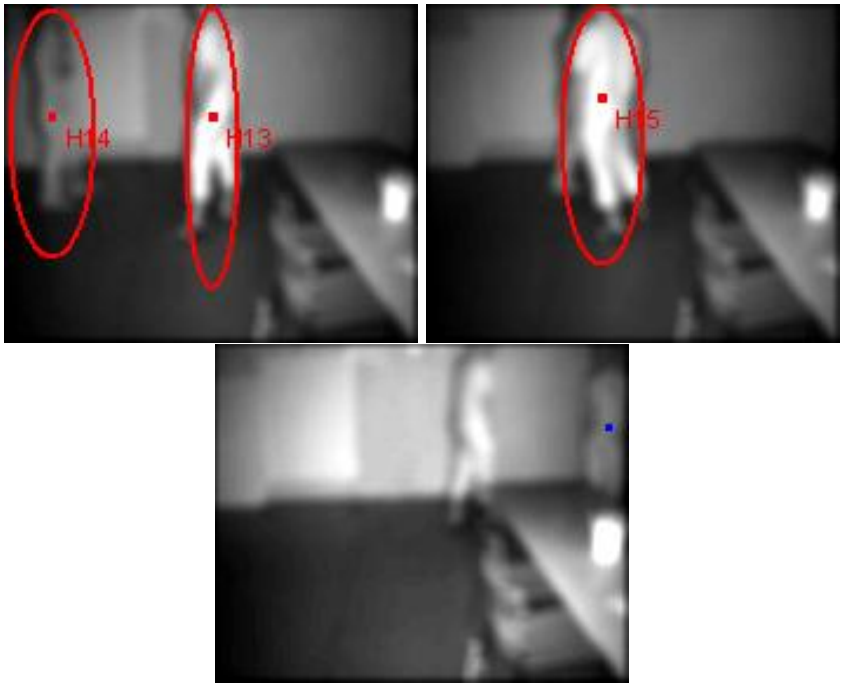

Figure 6.2: From frame 58 to 109 there are two persons moving and the algorithm detects 1.85 people. This result is acceptable considering that one of the people is occluded by the other one for some frames. Besides if people do no move for a while it can become harder to detect them, because they are slowly added to the background model. One example of this case can be seen in the third image.

Figure 6.3: From frame 110 to 120 there is one person in the scene and the detected are 1.

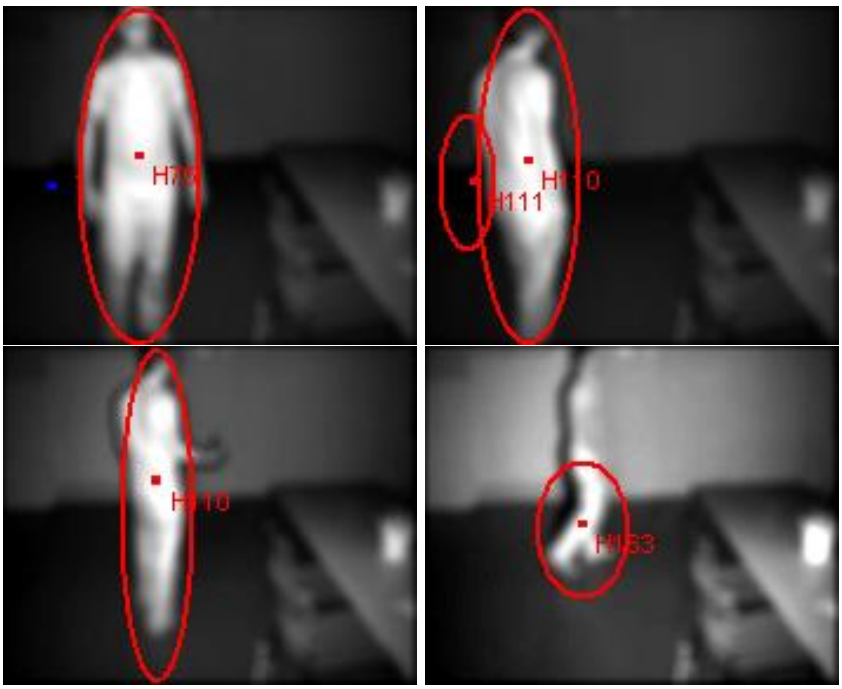

Figure 6.4: From frame 121 to 229 there is one person in the scene and the detected are 0.9. In some of these frames there are more than one ellipse for one person. This is due to the high integration time that creates instability when the person is to close to the device. It is also interesting to see that when a cluster is detected for the first time, it takes time to update all its parameters, and it could happen, like in the forth image, that the ellipse is smaller than the person, because it did not have much time to surround the body.

Figure 6.5: From frame 230 to 243 there are no people in the scene, and the output is zero people

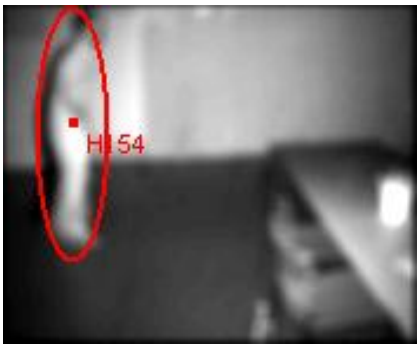

Figure 6.6: From frame 244 to 283 there is one person in the scene and the detected are 0.93.

Figure 6.7: From frame 283 to 296 there are no people in the scene, and the output is zero people

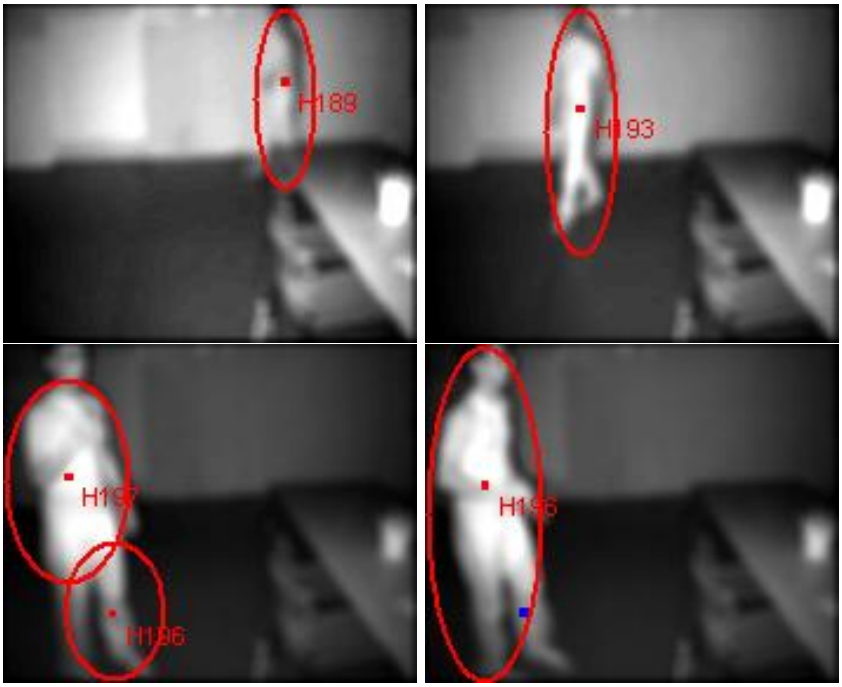

Figure 6.8: From frame 297 to 338 there is one person in the scene and the detected are 0.97. Also in this example there is a particular step of the algorithm. In fact as the person is close to the camera instead of having one big ellipse, his body is covered by two ellipses. Nevertheless in the next frame the algorithm merge the two ellipses because of their belong to the same person.

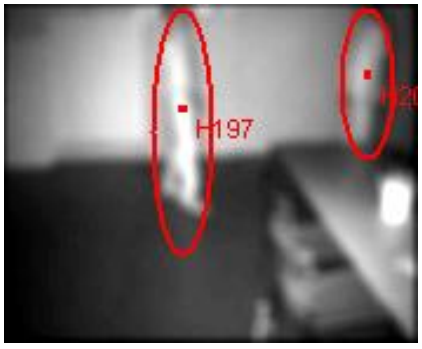

Figure 6.9: From frame 339 to 345 there are two people and the detected number is 1.65.

Figure 6.10: From frame 346 to 362 there one person and the algorithm output is 0.85. In the first image the body is not recognized because he is moving a panel near the wall, as a matter of fact the algorithm detects something non-human (blue-points) on the left of the person.

Figure 6.11: From frame 363 to 381 there are two persons and 1.75 are detected, also because there is an occlusion.

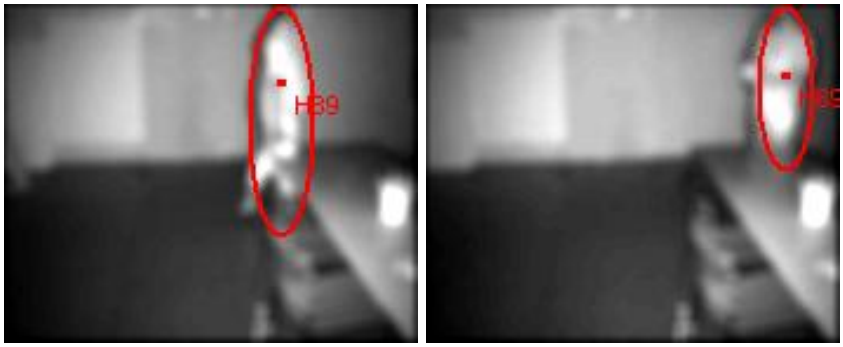

Figure 6.12: From frame 382 to 405 there is one person and 0.96 are detected.

Figure 6.13: From frame 406 to 427 there are no people in the scene, and the output is zero people

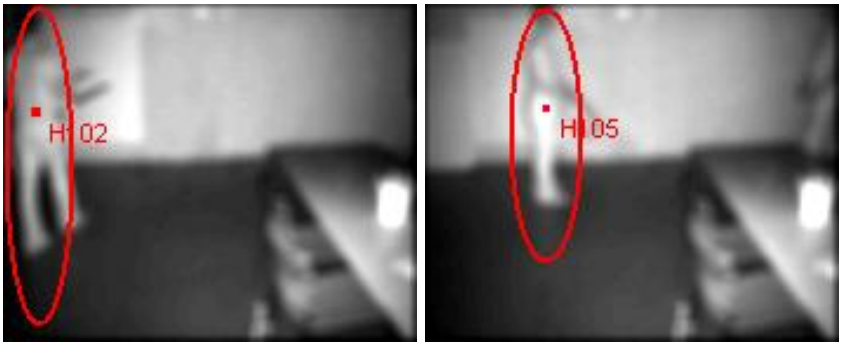

Figure 6.14: From frame 428 to 446 there is one person and 1 are detected.

Figure 6.15: From frame 447 to 491 there are two persons and 1.7 are detected. In this part of the sequence the two guys are throwing a pillow, that destabilize the scene, causing the lack of some blobs as shown in the figure.

Figure 6.16: From frame 492 to 535 there is one person and 0.98 are detected.

Figure 6.17: From frame 536 to 614 there are no people in the scene, and the output is zero people

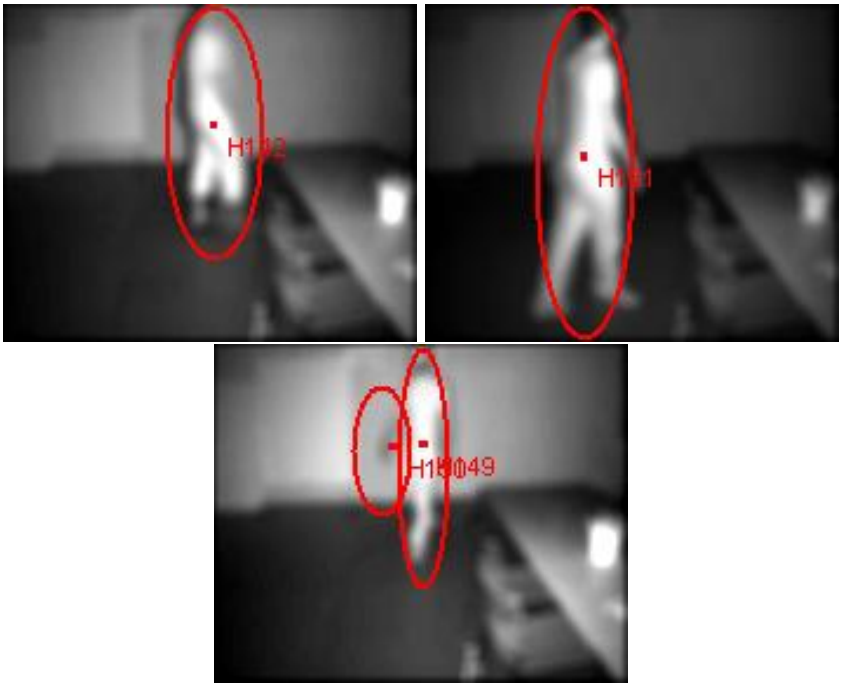

Figure 6.18: From frame 615 to 670 there is one person and 0.97 are detected. Also here there are some false positives, like in the third image. This one is due to the hand, that is detected as an independent blob from the body.

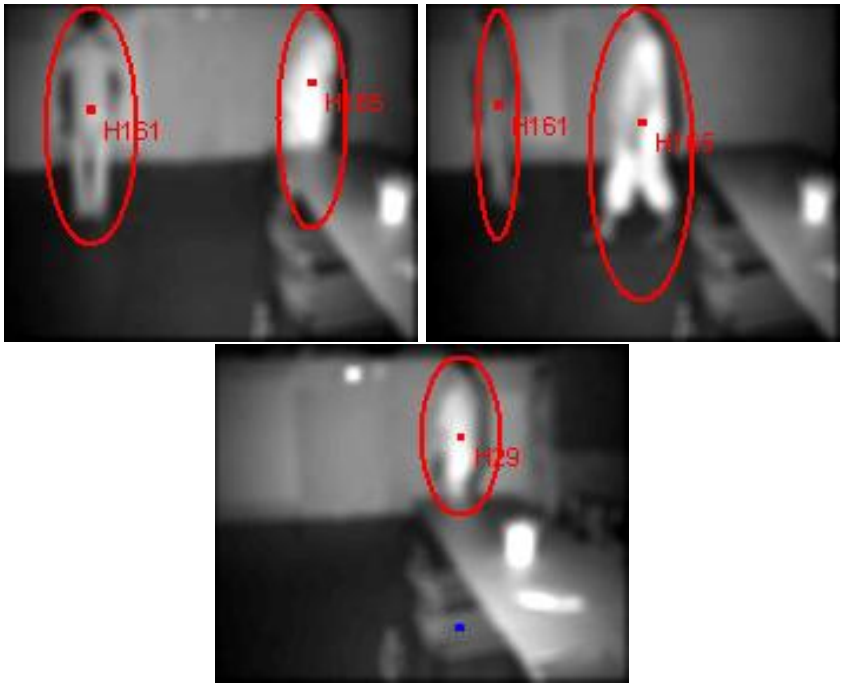

Figure 6.19: From frame 671 to 716 there are two persons and 1.75 are detected, because the tracking of the person close to the wall sometimes fails, maybe because the closeness to the wall, that makes harder the separation of the blob from the background.

Figure 6.20: From frame 717 to 796 there is one person and 1 are detected.

Figure 6.21: From frame 797 to 813 there are no people in the scene, and the output is zero people

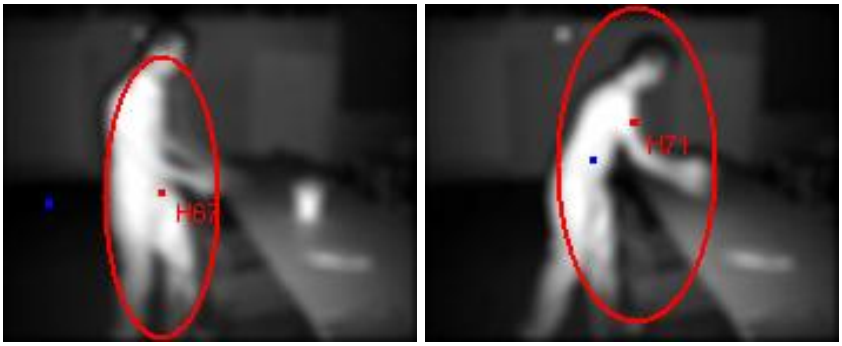

Figure 6.22: From frame 814 to 857 there is one person and 1.05 are detected. As the person moves a tray from the table, some false detection are present.

Figure 6.23: From frame 858 to 864 there are two persons and 1.78 are detected.

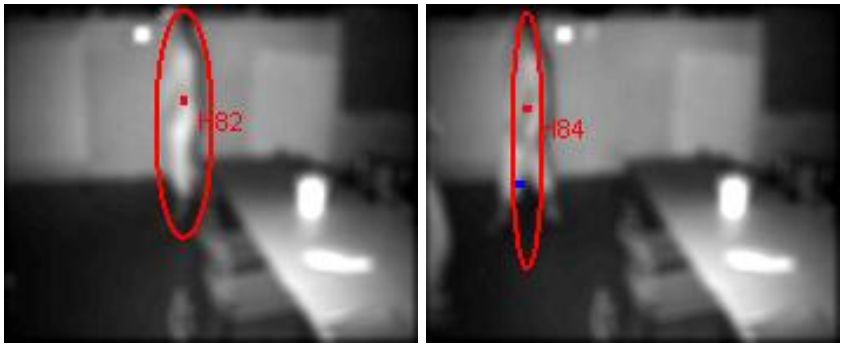

Figure 6.24: From frame 865 to 880 there is one person and 0.98 are detected.

Figure 6.25: From frame 881 to 894 there are no people in the scene, and the output is zero people

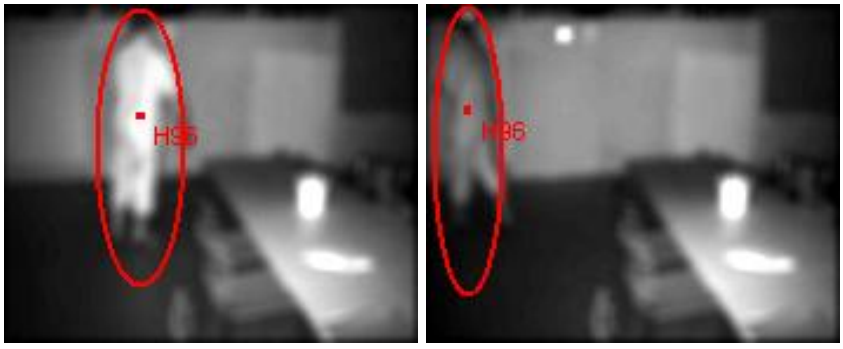

Figure 6.26: From frame 925 to 944 there is one person and 0.96 are detected.

Figure 6.27: From frame 945 to 962 there are no people in the scene, and the output is zero people

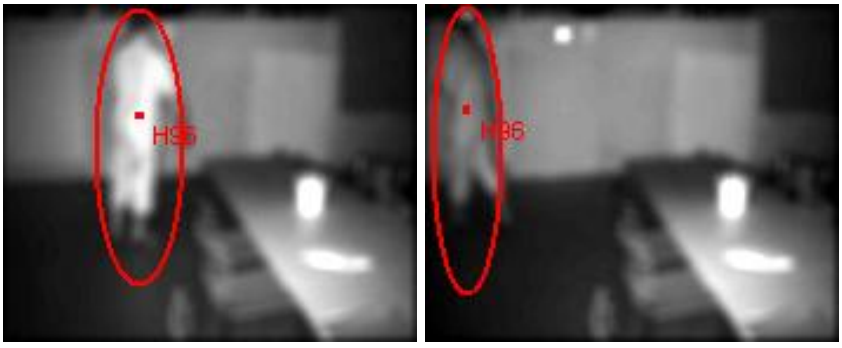

Figure 6.28: From frame 963 to 1006 there is one person and 0.97 are detected.

Figure 6.29: From frame 1007 to 1023 there are no people in the scene, and the output is zero people

Table 6.1: Analysis of the test sequence one

Frames	*True Positive*	*False Positive*	*False Negative*	*Accuracy*
1-100	121	14	10	0.896
101-200	72	6	9	0.923
201-300	62	4	9	0.939
301-400	88	4	19	0.956
401-500	101	2	16	0.980
501-600	35	3	2	0.921
601-700	86	10	16	0.896
701-800	99	3	15	0.970
801-900	73	7	4	0.912
901-1023	62	5	7	0.925

In the table 6.1 the sequence has been analyzed after having divided it in pieces of 100 frames each. For each part the people correctly tracked (*true positives*), the occurrences of objects classified as human (*false positives*) and the humans not recognized (*false negative*) have been counted. Regarding the *false negative*, almost all the occurrences of this class even if have not been classified as humans, have been detected as blobs. This may happen in the first frames in which a person is detected by the algorithm and its shape is not completely visible in the frame.

6.3 Test sequence two

In this second sequence a more extreme situation has been tested. The wall in the background is too far for the current modulation frequency and in the depth images it appears darker than the rest of the scene, because its distance is greater than the one reachable with the relative wave length.

Besides the auto-illumination of the camera was disabled, causing a very dark grey-scale sequence . For this reason the example will be shown principally using the depth images.

Besides it is interesting to notice that if people drift away from the camera too much they disappear in the depth images and it is very hard for the algorithm to detect them in this case.

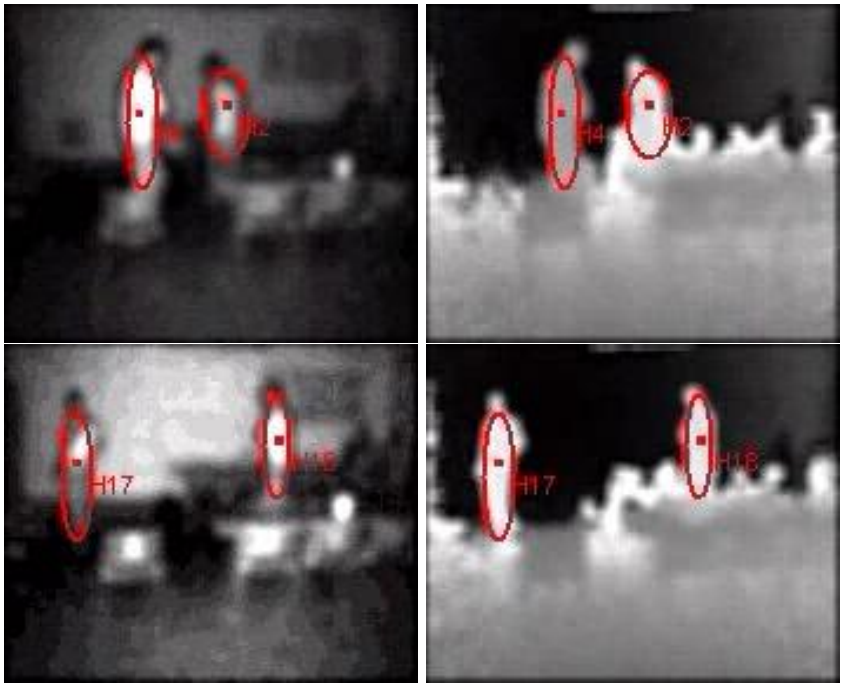

Figure 6.30: From frame 1 to 100 there are two people in the scene and the algorithm detects 1,55 people. In this figure there are two frames (the 33rd and the 96th) with the depth on the right. As it is possible to see the quality of the images is not as good as in the previous example. This is caused by the not correct setting of the camera parameters. The integration time is too small and for that reason the images are noisy, the background wall is dark because of the modulation frequency, too small too react it and because of the disabling of the auto-illumination the brightness images are very dark.

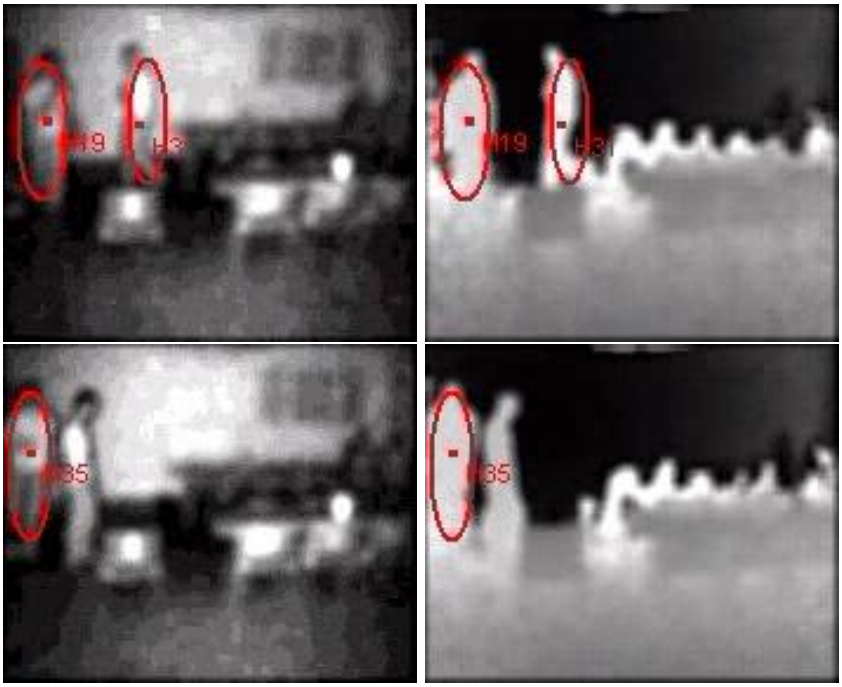

Figure 6.31: In this second part of the sequence the performance are similar to the previous piece. The people detected are 1.45 and it is not a bad value considering the bad quality of the images, that there is an occlusion and that one of the people is moving a chair, that crates instability in the probability images.

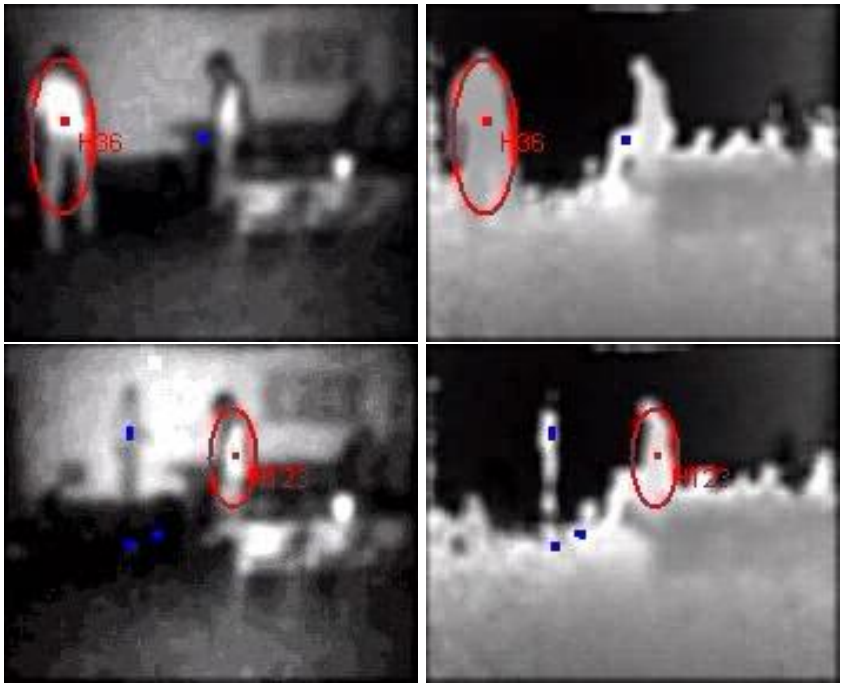

Figure 6.32: Between the frames 201 and 300 the number of detected people decreases, also because people move too close to the wall, too far from the camera to be detected in the depth image.

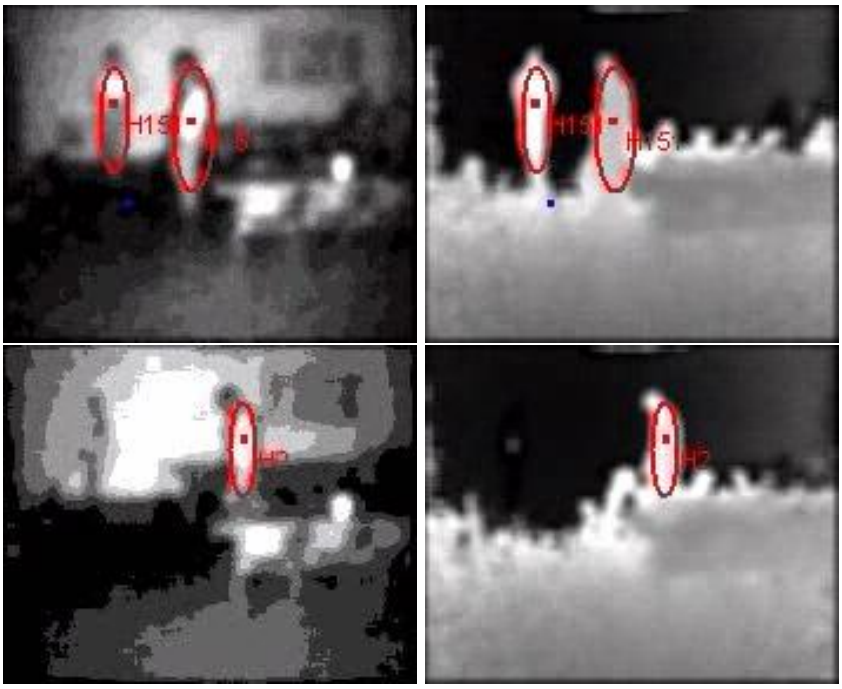

Figure 6.33: Also between the frames 301 and 400 the number of detected people decreases. Here it is very clear that one of the people almost disappears in the background of the image and for that reason he cannot be segmented from the background cluster.

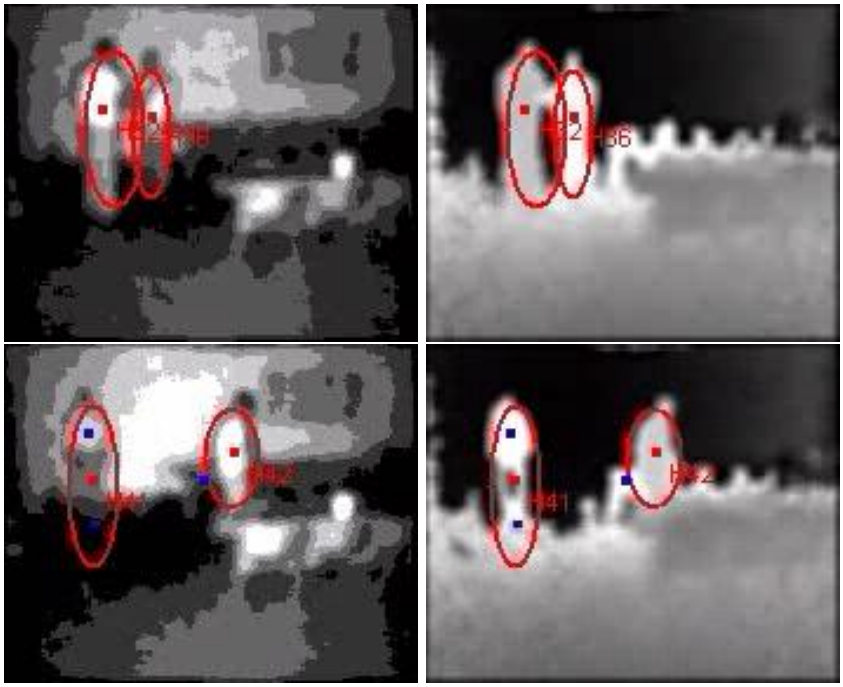

Figure 6.34: In the last part of the sequence the tracking results are much better, especially because the people are no more close to the wall and they can be detected by the camera.

Table 6.2: Analysis of the test sequence two

Frames	*True Positive*	*False Positive*	*False Negative*	*Accuracy*
1-100	155	1	48	0.760
101-200	147	2	53	0.735
201-300	131	4	69	0.655
301-400	126	5	74	0.630
401-500	59	6	141	0.295
501-682	292	10	72	0.802

As we can understand comparing the two test sequences to have good results it is important to set the camera parameters correctly and, as shown in the chapter 2, this depends on the scene.

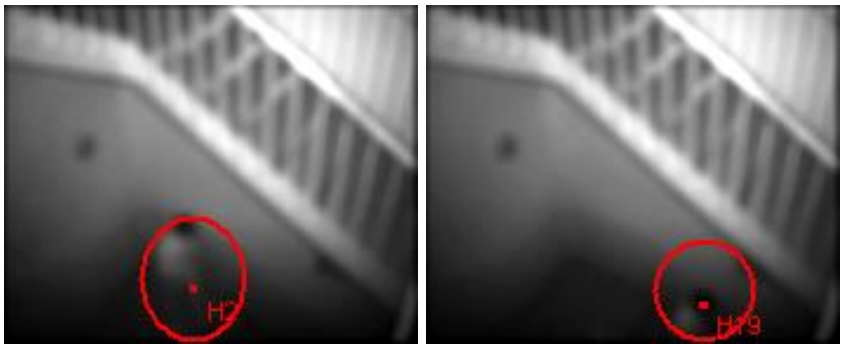

Figure 6.35: In the first 100 frames the accuracy is not very high: 0.60. As seen also in the other tests at the beginning the gaussians of the background subtraction need to steady and the background is not yet correctly generated

6.4 Test sequence three

In this test a top view sequence is analyzed. In this case the camera has been placed on the stairs to monitor people passing on a landing. As in this case the people are not taken frontally, the system for the human recognition was disabled. In fact the mask shown in the chapter 5 is useless and in this way people are tracked as normal object. Nevertheless this sequence has been taken with the correct parameters and for that reason the quality of the images is good and just few false negatives are generated.

Table 6.3: Analysis of the test sequence three

Frames	*True Positive*	*False Positive*	*False Negative*	*Accuracy*
1-100	32	0	16	0.604
101-200	48	0	4	0.923
201-300	15	2	3	0.833
301-400	22	0	2	0.916
401-500	38	1	3	0.927
501-600	7	2	3	0.700

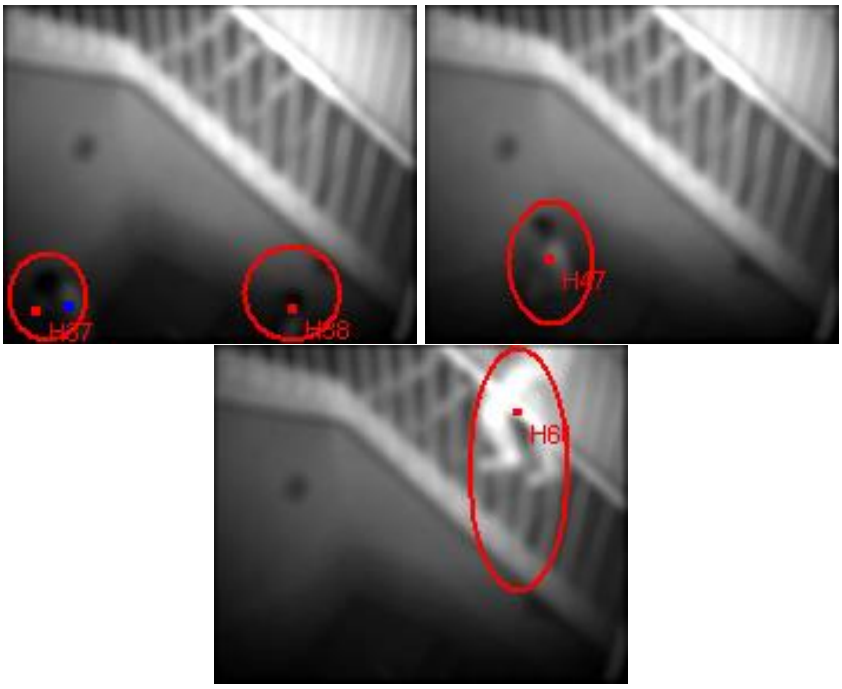

Figure 6.36: From the frame 101 to 200 the accuracy is 0.92, much better than the first 100 frames. As shown in the third image the ellipse sometimes could be longer than what it is required. This is caused by the perturbation had on the stairs below the person. This problem cannot be found in the other tests where people move horizontally on the floor and they can never walk one on each other, as in the case of the stairs. As a matter of fact for the other tests the corresponding problem is a blob with a width greater than normal.

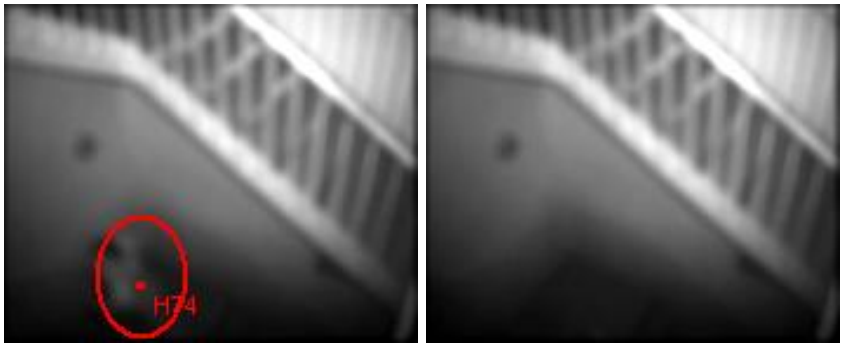

Figure 6.37: From the frame 201 to 300 the accuracy is 0.83, but in this part the frames without people walking are much more than in the other case, just 18 people appeared in different frames.

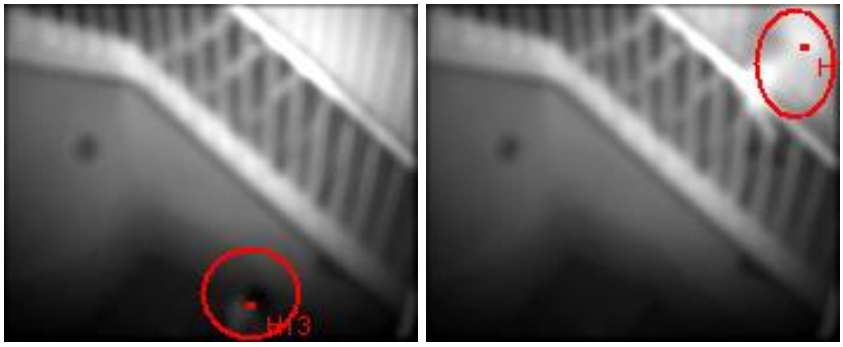

Figure 6.38: From the frame 301 to 400 the accuracy is 0.91 and the false negatives are just two.

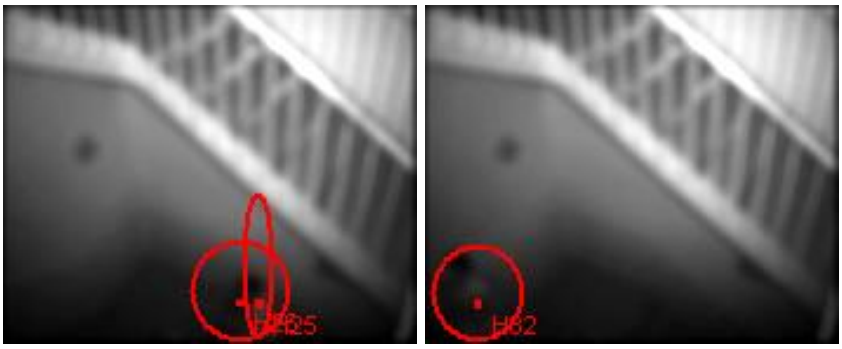

Figure 6.39: From the frame 401 to 500 the accuracy is 0.93 and the false negatives are just three. As showed in figure sometimes a blob can generate two ellipses but this happens it lasts just one frame. In the following one the blobs are merged to create one ellipse. As seen also for the other tests, this problem is due to the detection of two initial centers on the same blob. Anyway this is not a problem, it is a reasonable behavior, as long as the two blobs generated are merged because of their similarity, as it happens in this case in the following frame.

Figure 6.40: In the last 100 frames there are just 10 human bodies, among those the system recognize 7, and, as shown in figure, also here there are some small false detection. These kind of false positives would not be present in a frontal sequence where the system for human recognition is enabled. In that case probably that small cluster would have been deleted as there is no human shape inside it.

Figure 6.41: Because of the reflection problem in the depth images there is noise on the right corner. Nevertheless the mixture of gaussians method is quite stable and able to face with it. As a matter of fact it does not create many problems during the tracking, also because in the mixture of gaussians for each pixel it is possible to associate more values (one for each normal function) and a noise pixel has in this case two or three main values.

6.5 Other tests

In this test the camera is not mounted correctly and there are some reflection due to the shelf, as explained in the chapter 2, on the right of the frames. For that reason this part of the scene is very noisy and false positives might be generated.

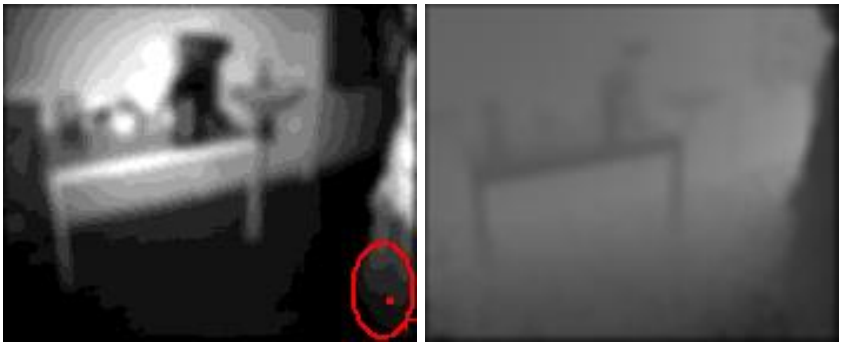

Figure 6.42: Whereas in this other frame the noise creates more problems and because of it a false positive is detected.

Chapter 7

Conclusions

In this thesis the prototype of a time-of-flight camera called SwissRanger and produced by Mesa has been analyzed and used to perform people tracking using it as a stationary camera. In the first step the camera has been tested and many sequences have been taken tuning its parameters to improve the quality of the images for this application.

After that, according to the state of the art for people tracking and to the performance of this device, the system has been divided in the following steps:

1. background subtraction
2. blob detection
3. blob tracking
4. human recognition

7.1 Background subtraction

The most popular background subtraction methods present in literature have been analyzed and three of them have been implemented for the time-of-flight

camera, so that also the depth information is used. The methods implemented are the mixture of gaussians, the kernel density estimation and maximum-value algorithm.

After having implemented them some tests have been performed to choose the parameters giving the best performance for each of them and, analyzing these tests the best the mixture of gaussians method has been chosen to be used in the following experiments.

The motivations for this choice can be read in the third chapter where the results for the background subtraction are shown and motivated.

7.2 Blob detection and tracking

By means of the background subtraction the foreground has been extracted from the background and now it has to be divided into blobs.

To do this a method inspired to Pece's tracking algorithm [11] has been implemented and extended to be used with the probability images and with the depth information. The foreground is segmented in ellipses, whose growth depends on a two-dimensional gaussian. The calculation for the parameters is done through the expectation-maximization algorithm as shown in the third chapter, with which the the ellipses containing the blobs increase or decrease their size according to the probability values coming from the background subtraction, the position of the pixel in the frame and their depth difference.

This process is repeated for each frame where the state of the EM initial iteration is taken as the last one of the previous frame. In this way, after the updating, each blob can be tracked frame by frame.

7.3 Human recognition

The content of the blobs tracked can be divided into human and not-human. To perform this step the blobs are analyzed according their orientation, dimensions and content.

As shown in the fifth chapter the dimensions and orientation are derived by the blob covariance matrix. Whereas the content is analyzed by multiplying it for

a mask created to generate high value if a human shape is present. Masks like this one are used in the Viola-Jones detection algorithm.

Each blob is given a probability to be a human according to these values and with this probability the blob is considered human or not-human.

7.4 Experimental results

In the last part of the thesis some tests have been performed. The test sequences chosen and shown in the sixth chapter have been taken with different camera parameters, also to test extreme situations, like using too short wave lengths or too big integration times.

Appendix A

Implementation code

All the work belonging to this thesis has been developed using Matlab. The choice of using Matlab comes from the fact that the SwissRanger provides a Matlab interface that allows to get easily the images taken with the device.

In this appendix a brief explanation of the most important functions implemented is offered.

A.1 Setup and background subtraction

1. **SwissRanger**: with this script it is possible to take a grey-scale and depth sequence by setting the camera parameters and the sequence length and save them in Matlab matrices.
2. **MixModel**: it computes the probabilities to belong to the foreground for each pixel of the sequence using the mixture of gaussians methods with the set learning rate and number of gaussians.
3. **KDE**: also this function computes the probabilities to belong to the foreground for each pixel of the sequence, but using the kernel density estimation method.

4. **BSMAximum**: this is the third method to perform background subtraction, using the maximum-value method with a given window of frames.

A.2 Detection and tracking

1. **PeceTracking**: this is the main function of this sections. It takes the sequence and the probabilities coming from the background subtraction and perform the tracking. Of course to do that it calls the functions to perform the EM iterations, to calculate the centroids, the covariance matrices ... etc

2. **EMStep**: this perform a step of the expectation maximization algorithm.

3. **CalculateFBackground**: calculate for each pixel the probability to belong to the background.

4. **CalculateFCluster**: calculate for each pixel the probability to belong to the a foreground cluster.

5. **CalculatePosterior**: calculate the posterior probability for each cluster.

6. **CalculateSigmas**: update the values of the covariance matrix.

7. **CalculateCentroids**: update the values of the clusters centroids.

8. **FindNewClusters**: this is the first step in the tracking algorithm. It finds the new clusters present in the current frame.

9. **MergeAndEliminate**: if there are the conditions explained in the previous chapters, it merges or eliminates the foreground clusters.

A.3 Human recognition and optimization

1. **OptimizeEllipse**: this function tries to change the ellipse dimensions to improve the fitting.

2. **WeightBody**: this function performs the multiplication of the human blob with the mask, to state the similarity for the blob to contain a human.

These are just the most remarkable functions. Also other functions have been implemented for instance to manipulate matrices or just to perform the experimental test. These ones are not reported because they do not have a direct correspondence with the algorithm.

Bibliography

[1] F. Xu, K. Fujimura. Human detection using depth and gray images. *IEEE Conference on Advanced Video and Signal Based Surveillance*, pages 115–121, 2003.

[2] D. Harwood A. Elgammal and L.S. Davis. Non parametric model for background subtraction. *Proc. ECCV 2000*, pages 751–767, June 2000.

[3] B. Fruhstuck C. Beleznai and H. Bischof. Human detection in groups using a fast mean shift procedure. *International Conference on Image Processing*, vol.1:349–352, 2004.

[4] W.E.L. Grimson C. Stauffer. Adaptive background mixture models for real-time tracking. *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, vol.2:246–252, 1999.

[5] T. Darrell C. Wren, A.Azarbayejani and A.P.Pentland. Pfinder: Real-time tracking of human body. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 19:780–785, 1997.

[6] A. Thomas D.J. Lee, P.Zhan and R. Schoenberger. Shape-based human intrusion detection. *SPIE International Symposium on Defense and Security, Visual Information Processing XIII*, pages 81–91, 2004.

[7] D. Hogg. Model-based vision: A program to see a walking person. *Image and vision computing*, 1:5–20, 1983.

[8] B.P.L. Lo and S.A. Velastin. Automatic congestion detection system for underground platforms. *Proc. ISIMP*, pages 158–161, 2001.

[9] H.Fujiwara M. Seki, T.Wada and K.Sumi. Background subtraction based on cooccurrence of image variations. *Proc. ECCV 2003*, 2:65–72, 2003.

[10] B. Rosario N.M. Oliver and A.P. Pentland. A bayesian computer vision system for modeling human interactions. *IEEE Trans. on Pattern Anal. and Machine Intell.*, vol.22(no.8):831–843, 2000.

[11] Arthur E.C. Pece. Generative-model-based tracking by cluster analysis of image differences. *Robotics and Autonomous Systems*, vol.39((3-4)):pp. 181–194, 2002.

[12] M.Piccardi R.Cucchiara, C.Grana and A.Prati. Detecting moving objects, ghosts, and shadows in video streams. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 25:1337–1442, 2003.

[13] C. Stauffer and W.E.L. Grimson. Adaptive background mixture models for foreground segmentation. *Proc. IEEE CVPR*, pages 246–252, 1999.

[14] K. Sumi T. Haga and Y. Yagi. Human detection in outdoor scene using spatio-temporal motion analysis. *International Conference on Pattern Recognition*, vol.4:331–334, 2004.

[15] Y.Shirai T. Tsukiyama. Detection of the movements of persons from a sparse sequence of tv images. *Pattern recognition*, 18:207–213, 1985.

[16] D. Toth and T. Aach. Detection and recognition of moving objects using statistical motion detection and fourier descriptors. *International Conference on Image Analysis and Processing*, pages 430–435, 2003.

[17] Sang Min Yoon and Hyunwoo Kim. Real-time multiple people detection using skin color, motion and appearence information. *International Workshop on Robot and Human Interactive Communication*, pages 331–334, 2004.