ECSE-529
Computer and Biological Vision

Project Report

# Security Event Detection Using Bayesian Learning and Particle Filtering

by

Olivier St-Martin Cormier

December 2, 2010

ECSE-529
Computer and Biological Vision

Instructor: M. D. Levine

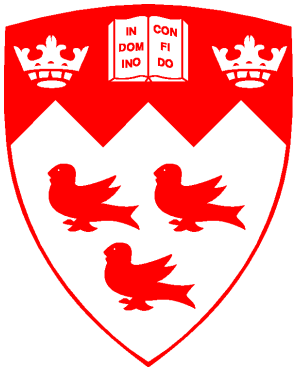
Project Report

# Security Event Detection Using Bayesian Learning and Particle Filtering

by

Olivier St-Martin Cormier
olivier@cim.mcgill.ca
ID: 260234106

December 2, 2010




Electrical and Computer Engineering Department
3480 University Street Room 410
Montreal, Quebec
Canada H3A 2A7

Abstract

*This project involves implementing a method to detect human beings and pieces of luggage in image sequences recorded in public environments. The goal of this project is to recognize potential security events such as loitering, luggage theft, and unattended luggage. The human tracking stage is accomplished by using a Bayesian multi-blob human tracker known as BraMBLe. This project demonstrates some of the critical aspects that need to be considered for such applications including the effect of camera positioning and the importance of proper training sequences. Throughout this report, it becomes clear that a perfect detection system is probably not possible to implement, but that the results are surprisingly accurate.*

*Several videos demonstrating this project in action can be found at*
http://cim.mcgill.ca/∼ olivier/ECSE529

# Contents

# List of Figures

## 0.1 Introduction

In recent years, public environment security and surveillance have become a growing concern. The obvious solution to this problem has been to install an increasing amount of security cameras. As more and more of these cameras are installed, the number of human operators required to monitor the constant streams of video become problematic. Computer vision scientists from around the world work to make the quantity of information more manageable by using various methods to determine when security events happen and to attract the human operator's attention toward these potential threats.

There is a large quantity of published papers on the topic of human tracking. For this reason, only those published in the past decade were considered. Amongst the proposed methods are some based on Kalman filtering ([3],[1]) to determine the position and trajectory of humans over time, but often overlook potential occlusion problems. Other methods use data from sensors more advanced than standard cameras. Those often involve infra-red sensors ([4]) or depth information acquired from laser scanners ([7]). Sadly, only image data is provided for the current project. The paper chosen ([9]) uses Bayesian learning ([6]) to detect and track people in images, which, according to the author, provides robustness towards partial occlusion problems.

## 0.2 Human tracking

The first step of the proposed project is to detect humans. For this task, the paper chosen [9] uses a modified version of a Bayesian multiple-blob human tracker called BraMBLe. It was modified to allow for extended use of the system by allowing the background and foreground models to be dynamically updated. These modifications are not required for the current project as the program is only meant to run for a short amount of time and that the environment conditions are fairly constant. For this reason, the BraMBLe tracker was implemented according to [6], without the modifications suggested in [9]. From the four provided camera orientations, the fourth was chosen because it is the one closest to the orientations of the cameras used in [9] and [6].

### 0.2.1 Image Filtering

To comply with the methodology of [6], the image is converted from the RGB color space to the YCbCr color space using the equations 1 through 3. In these equations, $R$, $G$, $B$ stand for the

intensity of the red, green and blue channels, respectively; $Y$ stands for the luma value, and finally, $Cb$ and $Cr$ stand for the blue and red chroma difference values, respectively.

$$Y = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B \quad (1)$$

$$Cb = -0.169 \cdot R - 0.331 \cdot G + 0.499 \cdot B + 128 \quad (2)$$

$$Cr = 0.499 \cdot R - 0.418 \cdot G - 0.0813 \cdot B + 128 \quad (3)$$

To get a manageable amount of useful information about the image, a 5 pixels by 5 pixels grid is placed on the image. At each position of this grid, 2 filters are applied to each of the three color channels, resulting in a response vector containing six values. The first filter is a Gaussian filter with equation shown in 4 and the second is a Laplacian of Gaussian filter, or, as they call it, a Mexican hat filter, with equation shown in 5. As can be seen in their equations, both of these filters are centered around the origin ($\mu_x = \mu_y = 0$) and are radially symmetric ($\sigma_x = \sigma_y = \sigma$).

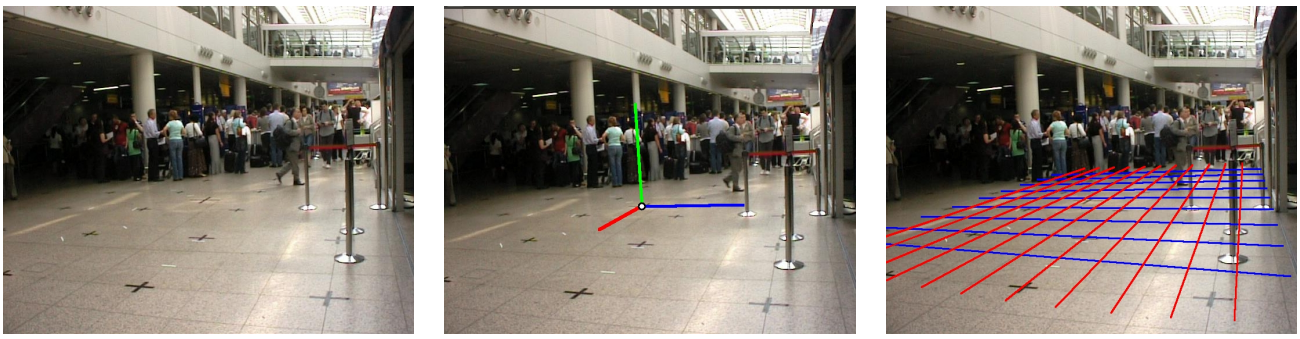
$$F_{Gaussian}(x, y) = \frac{e^{-\left(\frac{x^2+y^2}{2\sigma^2}\right)}}{2\pi\sigma^2} \quad (4)$$

$$F_{MexicanHat}(x, y) = \frac{e^{-\left(\frac{x^2+y^2}{2\sigma^2}\right)}}{\pi\sigma^4} \cdot \left(1 - \frac{x^2+y^2}{2\sigma^2}\right) \quad (5)$$

As per the instructions in [6], the size of both filters was set to 5 pixels by 5 pixels and the standard deviation ($\sigma$) was set to one third of that size, as suggested in [10].

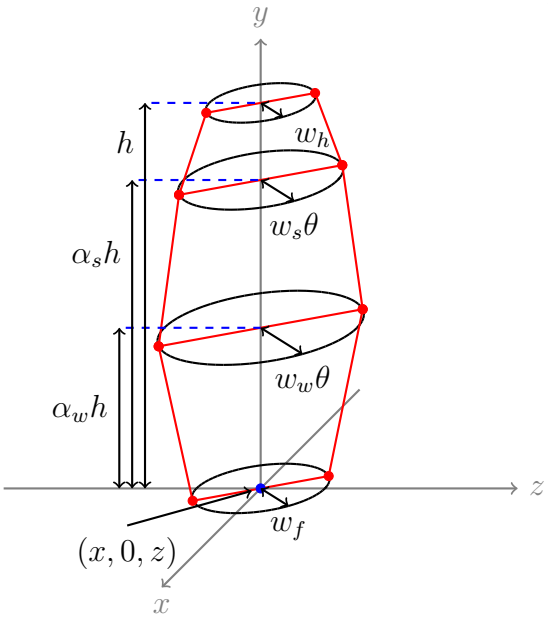
### 0.2.2 Coordinate System

The calibration data for the cameras used when capturing the sequences is provided as an xml file containing all of the pertinent parameters. The calibration was found using the Tsai camera model. This camera model allows one to find the two-dimensional coordinates, on the image plane, equivalent to any given real world three-dimensional coordinates. This conversion is made using the equations found in [5]. The only tricky part is that it is required to solve an under-determined third order system of equations, which inevitably results in multiple possible solutions. For simplicity, an existing implementation of the method was used [11]. Figure 1 shows that the method works fairly well by placing the X, Y, and Z axis over an empty image (X is Red, Y is Green, and Z is Blue). From figure 1(c), it can be noted that the precision, while sufficient for the needs of this project, is not perfect, as the lines are not exactly aligned with the floor markers.



(a) Empty Image (b) XYZ Axis over the image (c) Ground Plane of the image

Figure 1: Result of mapping 3D points to the 2D image plane using the Tsai camera calibration method.




| Parameter | Description |
|---|---|
| $x$ | Floor position along the x-axis |
| $z$ | Floor position along the z-axis |
| $w_f$ | Radius of the feet circle |
| $w_w$ | Radius of the waist circle |
| $w_s$ | Radius of the shoulder circle |
| $w_h$ | Radius of the head circle |
| $h$ | Total height |
| $\theta$ | Rotation Parameter |
| $\alpha_w$ | Ratio between the height of the waist and the total height |
| $\alpha_s$ | Ratio between the height of the shoulder and the total height |

Figure 2: Graphical representation of a generalised cylinder and definition of all parameters

### 0.2.3 Human template

The BraMBLe tracker uses a generalised cylinder model to approximate a generic human shape. Such a cylinder, with all of the associated parameters, is shown in figure 2. The $\theta$ parameter is used to approximate rotation of the human by scaling the radius of the waist and shoulder circles. It is merely an approximation, but prevents having to compute sines and cosines to draw the cylinders onto the image plane. In order to implement temporal tracking of objects, three extra parameters are associated with each generalised cylinder that are not shown in figure 2. Those are a unique identifier, to distinguish each object from the others, and two velocity components, in the X and Z directions, used to predict the next probable position of the human.

To determine the projected shape of this model onto the image, a plane (light blue) orthogonal to both the ground plane and a line from the camera to the floor position of the cylinder is defined, as seen in figure 3. The points of intersection between the circles representing the generalised cylinder and the orthogonal plane are shown as red points. These points correspond to the vertices of the polygon that will be used as the human shape template for the Bayesian learning algorithm.

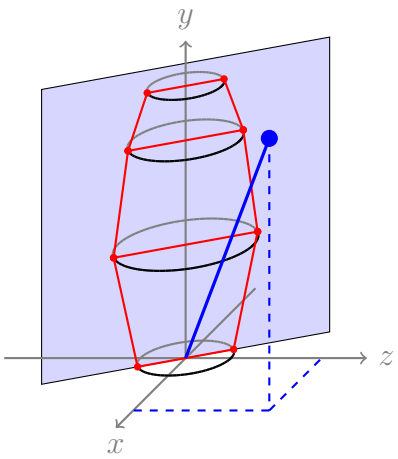



Figure 3: Generalised cylinder intersected by a plane (pale blue) orthogonal to the x-z plane and the line between the camera position (blue dot) and the floor position of the cylinder.

An approximate rendering of such a generalised cylinder is shown in figure 4. The contour of the shape is shown in red, the main axis of the cylinder is shown in green and locations of filter responses that are inside the area of the cylinder are shown with small blue circles. Determining whether a given two-dimensional coordinate is inside the polygon formed by the generalised cylinder

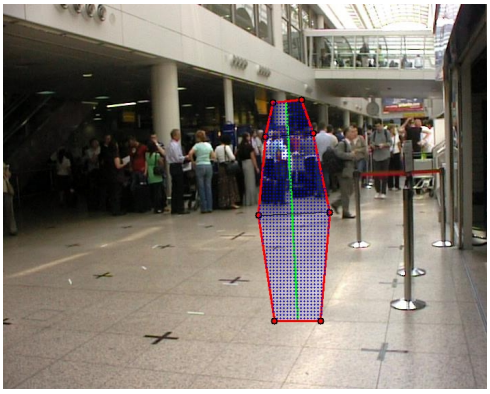


Figure 4: Approximate rendering of a generalised cylinder

is a computational geometry problem commonly referred to as "point in polygon inclusion". It is solved by using the Jordan curve theorem [8]. The method requires one to draw a straight line segment (horizontal or vertical for simplicity) from the point in question to the edge of the image and count the number of times this line intersects with the boundaries of the polygon. If the number of intersection is odd, the point is inside the polygon and if the number of intersections is even (including 0 intersections), then the point is outside.

### 0.2.4 Background-Foreground Segmentation

In order to determine the probability of a given filter response being generated by a foreground object, Bramble uses two distinct models: one for the background and one for the foreground. Both of these models are learned using standard k-means algorithms, but on different types of data.

Learning the background model is done using a set of training images (in the current case approximately 900 frames) containing as little foreground activity as possible (ideally none). At each of the position of the filter grid, the k-means algorithm is applied to find a mixture of four Gaussian components. This allows for periodic changes in the background to be learned and included in the models. An example of such a periodic event would be a blinking light, for which each of the two states (on and off) would be taking into consideration by one of the four Gaussian components. Basically, what this means is that each position of the image has a different trained background model with four Gaussian components, each with its own mean and covariance matrix.

Given any location $g$ and its associated filter response vector $z_g$, the probability of $z_g$ being generated by the background can be written as in equation 6, in which $\mathcal{N}$ stands for normal

(Gaussian) distribution, $\mu_g^k$ is the mean of the $k^{th}$ learned Gaussian component at location $g$, $\Sigma_g^k$ is the covariance matrix of component $k$ at $g$, $\delta_b$ and $\tau_b$ are two small constants. According to [6], adding a small perturbation ($\delta_b I$) to the covariance matrix "is sufficient to suppress false foreground responses to shadows". Another good reason for adding this perturbation is to prevent the determinant of the covariance matrix of ever being zero, which in turn prevents possible division by zero. Similarly, the addition of $\tau_b$ prevents problem which could arise when taking the log of the probabilities later on, by making sure that the probability is never 0. According to [6], the two constant values should be $\tau_b = 3 \cdot 10^{-13}$ and $\delta_b = 100$. These values were found to work well. From equation 6, it can also be noted that all of the components have the same weight, which makes learning the model slightly simpler.

$$p(z_g|background) = \sum_{k=1}^{4} \frac{1}{4} \mathcal{N}(\mu_g^k, \Sigma_g^k + \delta_b I) + \tau_b \tag{6}$$

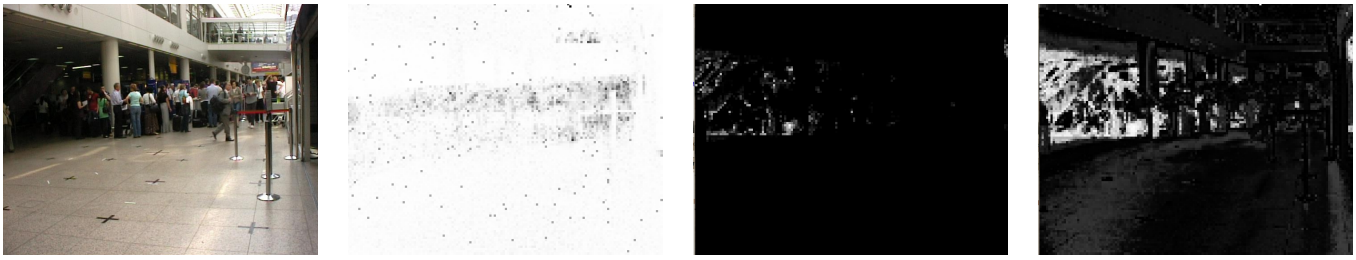
The foreground model is trained in a different way. Only sixteen Gaussian components are learned for the whole image. To make sure that the training focuses on foreground objects, only location whose filter responses have a weak probability of being generated by the background are included in the training data. Again, a few hundred frames of training were used.

Once the model is learned, computing the probability that a given filter response vector $z_g$ at a position $g$ has been generated by a foreground object is similar to what was shown for the background model. Equation 7 shows how to compute such a probability. The biggest difference between equations 6 and 7 is that the mean and the covariance matrix do not depend on the location, only on the Gaussian component. As indicated in [6], a value of $\tau_{foreground} = 3 \cdot 10^{-13}$ yielded the best results.

$$p(z_g|foreground) = \sum_{k=1}^{16} \frac{1}{16} \mathcal{N}(\mu_{foreground}^k, \Sigma_{foreground}^k) + \tau_{foreground} \tag{7}$$

It is important to remember that the normal distributions described in 6 and 7 are multivariate normal distributions. In both equations, $\sigma$ is always set to a diagonal approximation of the covariance matrix.

Figure 5(b) demonstrates that even if there is no one walking in the foreground, the background probabilities near the center of the image are quite low. Similarly, in figure 6(b), part of the torso of the person walking through the scene has a high probability of being generated by the background. This is because the BraMBLe tracker's methodology assumes a static background



(a) Original Image (b) Background Probability (c) Thresholded Foreground Probability (d) Log-likelihood of an object being in the foreground

Figure 5: Probabilities from learned background and foreground models applied to an empty image

during the training stage. This was not the case with the training data available. The background training data provided is "empty" in the sense that there are no human beings located in the main part of the scene, but there is a crowd walking around in the background, near the center of the image. This confuses the learning algorithm and make the learned model for these pixels unreliable. This region of the image will be referred to as the "dead zone", where reliability is very low. Because of this zone, figures 5(d) and 6(d) show that the segmentation between foreground and background is not perfect, as some parts of foreground objects are detected as background and vice-versa, but the results should be sufficient to compute the likelihood of human positions. One can also note that the results are only slightly worse than to those of [6] in terms of noise and misclassification. This difference of quality can be attributed to the fact that the background training sequence used in [6] was truly static.

(a) Original Image (b) Background Probability (c) Thresholded Foreground Probability (d) Log-likelihood of an object being in the foreground

Figure 6: Probabilities from learned background and foreground models applied to an image containing foreground objects

The effect of the "dead zone" is clearly shown in figure 7, which shows a sequence of images when someone walks across the image. During the first few frames, the body is almost fully

detected as foreground (the head is partly misclassified). Then, as the person moves toward the center of the scene, its head and torso become misclassified due to the unreliable "dead zone".

### 0.2.5 Computing the likelihood of an hypothesis

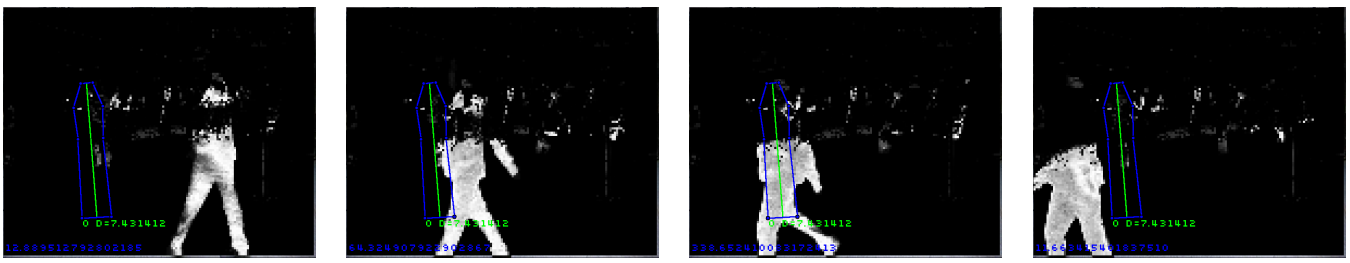
An hypothesis, in the context of this project, is defined as f[inite set of possible human locations and parameters (a set of generalised cylinders). The likelihood of an hypothesis being accurate is based on the probabilities found in the background/foreground segmentation part.

The procedure from [6] is followed to compute the likelihood of an hypothesis. The first step of this procedure is to make sure that the hypothesis is at least physically plausible by discarding it, if the distance between the floor coordinates of any pair of generalised cylinders is smaller than 0.5 meter. This prevents multiple persons from occupying the same space. The second step is to sort the humans using their distance from the camera as the sorting key. To reduce computational cost, one can use the square of the distance as the sort key, thus preventing the computation of multiple square roots. No specific sorting method is specified in [6], so the sorting was made using the bubble sort method. Next, one should generate an array to hold a label ($l$) for each filter position ($g$). These labels will be referred to as $l_g$. Initialize this array with zeros (no label) and set the initial likelihood to 0. Now, one should find the filter locations inside each generalised cylinder, in order. (the blue circles in figure 4) At each of these location, if $l_g$ is 0, then update $l_g$ with the index of the current generalised cylinder and add the value of $y_g$ to the likelihood. If $l_g$ is not 0, then that filter location has already been considered by a generalised cylinder closer to the camera. This only happens when a closer cylinder occludes another cylinder further from the camera and prevents adding $y_g$ twice to the likelihood. After considering each generalised cylinder, the likelihood has been found.

To demonstrate the concept in a simple manner, figure 8 shows the likelihood of an hypothesis for scene configuration. The likelihood when the person is directly inside the polygonal projection of the generalised cylinder is shown in 8(c) to be very high, as it should be. Similarly, when the person is completely outside the polygon,as shown in 8 (a) and (d), the likelihood is very small, but non-zero. The reason why the likelihood is non-zero is due to the noise in the foreground/background probabilities, mostly in the "dead zone" discussed earlier. Finally, 8(b) shows that the likelihood is somewhere between the maximum from (c) and the minimum from (a) and (d). This makes sense as the person is only partly inside the polygon.

(a) Sample frame 1 (b) Sample frame 2 (c) Sample frame 3

Figure 7: Effect of the "dead zone" clearly shown by the dissapearing head and torso of someone walking across the screen

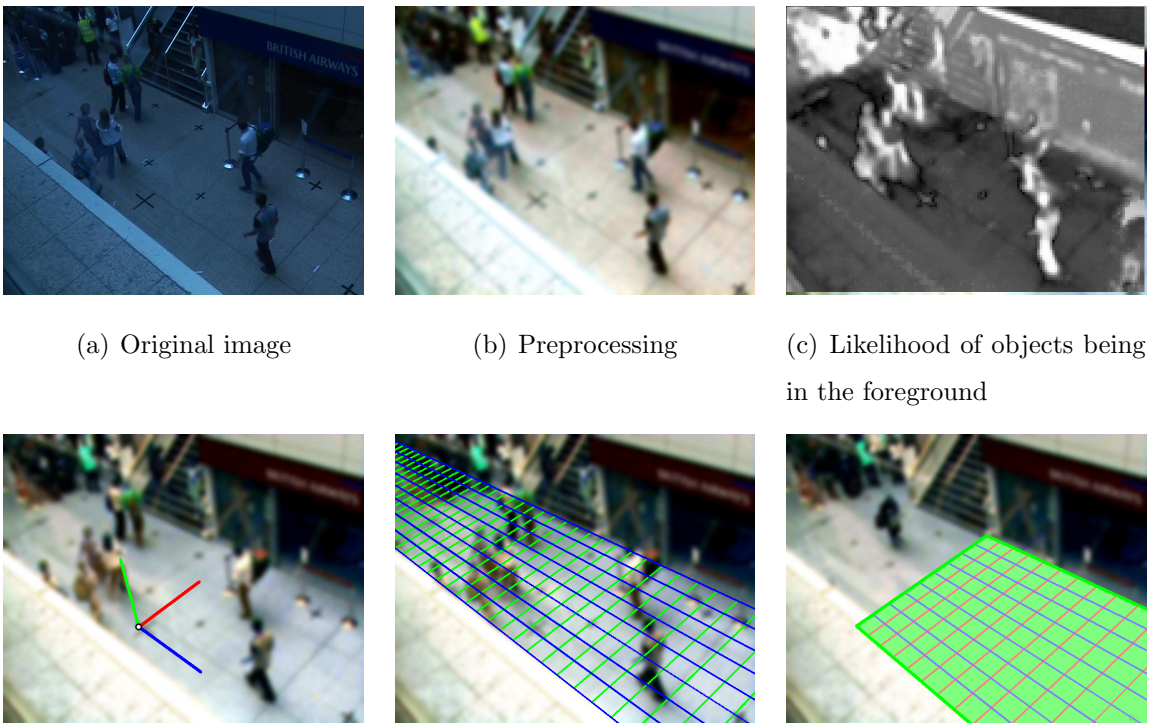


(a) Likelihood=12.889 (b) Likelihood=64.324 (c) Likelihood=338.652 (d) Likelihood=11.663

Figure 8: Evolution of the likelihood of a sample hypothesis for different scene configurations

### 0.2.6 Camera Problems

At this point, it became clear that the "dead zone" might create a bigger problem than anticipated. Fortunately, multiple camera orientations are provided. It was found that the first and second cameras have a similar problem, but that the part of the scene where the action takes place contains less clutter during the training sequence of the third camera (figure 9(f)). To further optimize the results, the images from the third sequence were preprocessed (figure 9(b)) by applying two filters: an auto-levels filter to make colors more distinguishable and a gaussian blur filter to ideally smooth out some of the noise and get better defined foreground blobs. The background and foreground were trained for this new camera (figure 9(c)) and the Tsai calibration was applied to the image to get a coordinate system (figure 9(d)) and a ground plane estimation (figure 9(e)).



(a) Original image

(b) Preprocessing

(c) Likelihood of objects being in the foreground

(d) Coordinate System

(e) Estimated ground plane

(f) Portion of the floor considered by the tracker

Figure 9: Third camera preprocessing, calibration and probability test

### 0.2.7 Particle Filtering

The BraMBLe tracker uses particle filtering to determine where humans are located. The idea of particle filtering, as used by BraMBLe, is to generate a large amount of possible scene configura-

tions, each called a particle, and from the likelihood of these particles, try to determine the real configuration of the scene.

#### 0.2.7.1 Particles

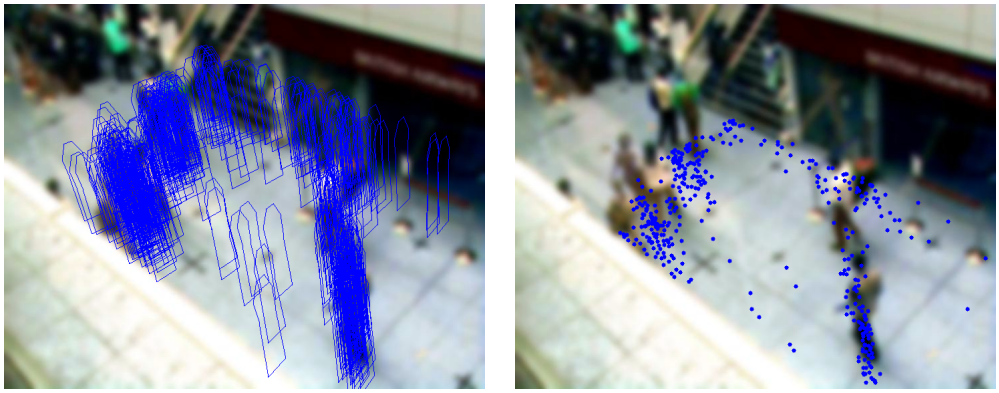
Particles contain a finite number of generalised cylinders; they each represent a possible configuration of objects in the scene. For the system to work properly, particles need to be updated at each time step (video frame). The update rule is quite simple; each generalised cylinder has a one percent chance of exiting the scene (disappearing) and there is a two percent chance of a new cylinder appearing. This is what allows an arbitrary number of objects to be tracked at the same time.

The generalised cylinders contained by the particle also need to be updated at each time step. The same equations as those presented in [6] were used. Furthermore, there was no need to try different values for the means and covariances of each parameters, because these values have a clear physical meaning and are logical. For example, the average height of a human is approximately 1.8 meters, as noted in [6].

The likelihood of a particle is defined as the sum of the likelihoods of the generalised cylinders it contains. Once multiple particles exists and that their respective likelihoods are known, it is possible to display those that have a higher probability of being accurate. The value chosen as the display threshold varies according to many factors. As the likelihood of a particle is the sum of other non-normalized probabilities, the likelihood depends on the number of cylinders per particle as well as the probability density function of the foreground/background segmentation. It was found that, in most cases, values below 2.5 allowed too many cylinders to be drawn. On the other hand, values above 3 did not allow enough cylinders to be drawn to determine the position of the humans accurately. Figure 10(a) shows the rendering of a set of particles using a threshold value of 2.75. For clarity, instead of drawing the full projections of the cylinders, figure 10(b) shows only the floor position of the possible humans. In figure 10(b), it is possible to see that the density of points is higher around the true position of the humans.

### 0.2.8 Marginalization of the results

To get a single position for each human from the set of points shown in 10, one needs to find the center of the zones with higher density of generalised cylinders. The instructions for this step in



(a) Most probable human locations and shapes

(b) Floor position of the most probable human locations

Figure 10: Display of high likelihood particles

[6] are fairly vague, and [9] does not go into any details about how this step was implemented. Considering the type of data, using a k-means clustering algorithm should be able to find the correct locations of humans. The clustering was made using ten clusters and thus a maximum of ten humans can be tracked at any given instant, but this number can easily be changed to allow for greater flexibility. Once all of the points are labeled according to the cluster they belong to, the probability of that cluster being a human is defined as the sum of the likelihoods of the individual points.

While being fairly efficient, this method often yields false positives. Fortunately, these usually appear and disappear very quickly. For this reason, most of them are discarded by adding a frame counter to each Gaussian cluster and only drawing the human shape if the counter is larger than a certain value. A value of two frames was found be be the smallest value capable of discarding most of these false positives.

### 0.2.9 Limitations of the human tracker and possible improvements

The most important problem with the tracker is that if two humans are close to one another, their respective identification numbers can be swapped. This problem is also mentioned in [6]. It could be solved by applying a method similar to the luggage identification method detailed in [9], which uses a learned mixture of Gaussian distribution for each piece of luggage to identify them based not only on their location, but also on their appearance. Another, simpler solution, would be to compute an approximate trajectory vector for each tracked human. This would allow to distinguish

between two persons crossing paths by knowing the direction in which they are traveling. The second problem is when a single human is detected as two different humans. Again, if each detected human had an estimated velocity, one could notice that two detected humans would be following the same path with the same velocity and assume that they are a single human being.

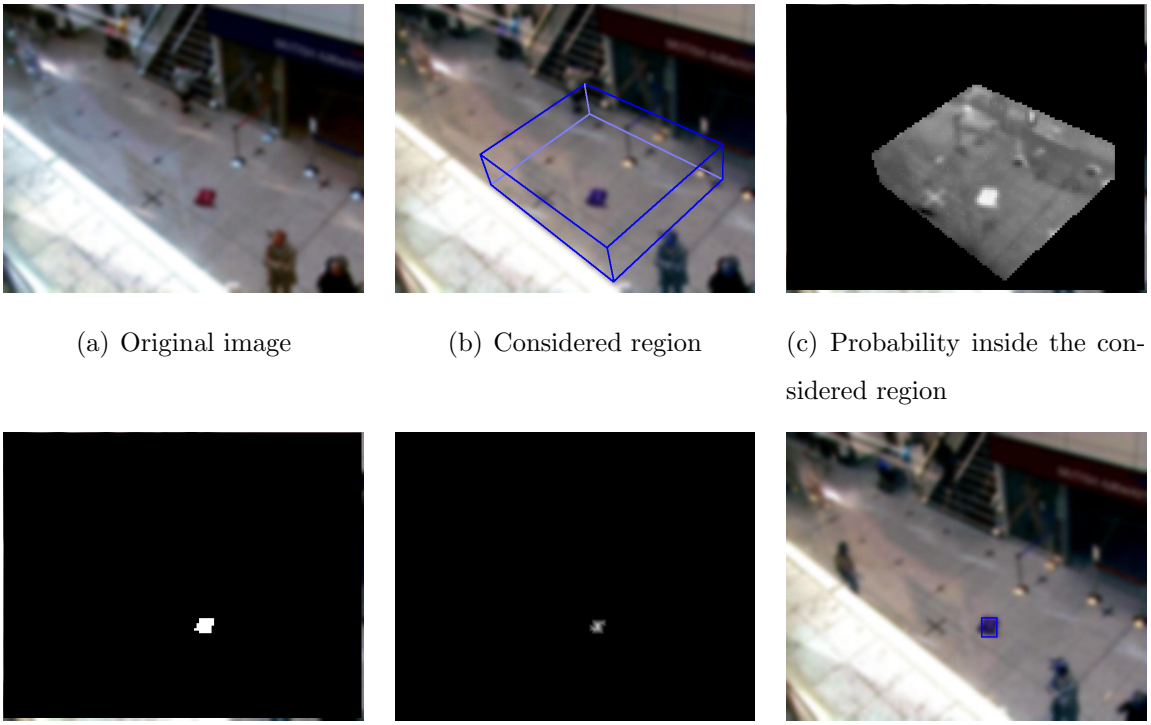
## 0.3 Luggage Detection

In the framework of this project, we are only interested in detecting abandoned luggage or theft. Both of these events involve bags being placed on the ground. For this reason, only stationary luggage needs to be detected. This greatly simplifies the task. To reduce the computational complexity further, only a finite region of the image is considered as potential location of luggage, as seen in figure 11(b).

Luggage detection requires a few steps. The first is to discard the probabilities outside of the considered region, as shown in figure 11(c). Then, thresholding the remaining probabilities allows to find blobs representing likely foreground objects (figure 11(d)). The instruction in [9] say that luggage should be found by finding blobs with an area greater than a certain value. It was found that a minimum distance value of nine *[assuming circular blobs with radius 9, the minimum area is approximately 250 pixels]* was able to find bags. To find such regions, a grass-fire distance transform is applied to the thresholded image, as seen in 11(e). From the distance transformed image, one can directly assess the size of blobs and find their centroids and bounding boxes, as seen in figure 11(f).

This, obviously, only works when all foreground blobs are pieces of luggage. To make sure humans in the foreground are not considered as bags, the probabilities inside regions occupied by humans are set to zero, as shown in figure 12. Some pixels from the human remain, but they do not constitute a large enough area to be detected as a bag.

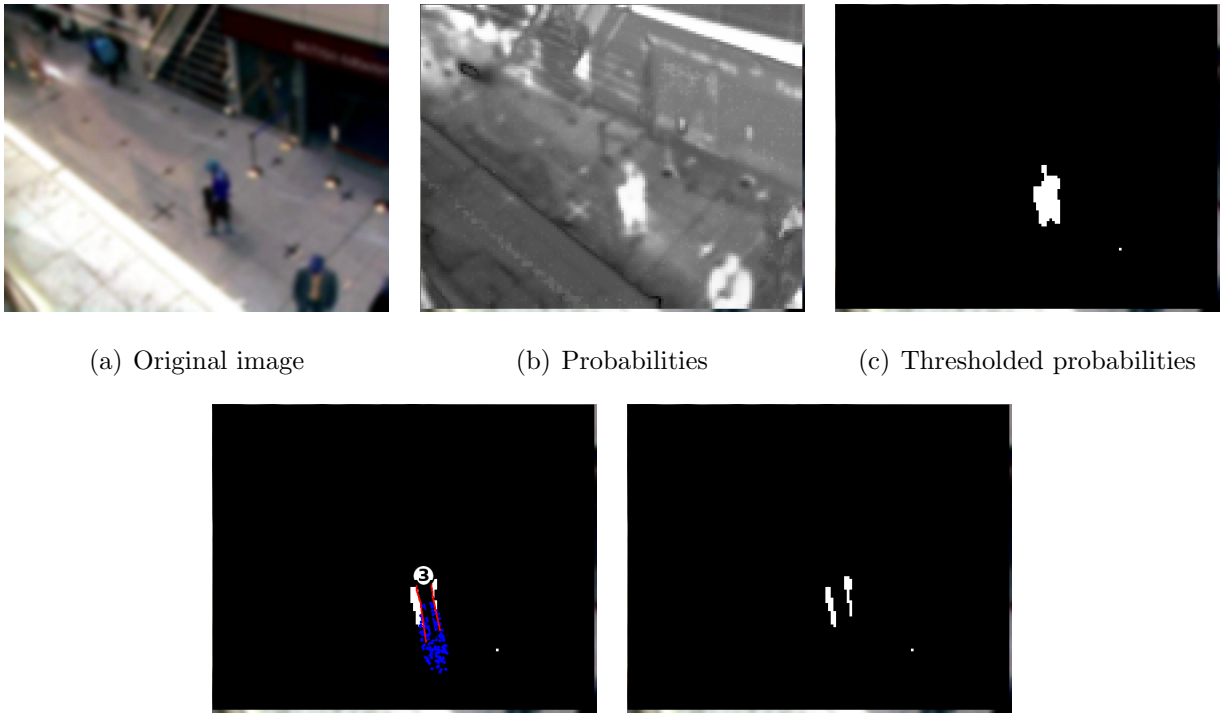
Using a few carefully chosen frames from the seventh scenario, it is possible to see a bag and two humans being properly recognized (figure 13(a)). A few frames later, we can see the person labelled "three" picking up the bag (still recognized as a bag) (figure 13(b)) then walking away (figure 13(c)), when the bag is no longer being detected as such. Finally, the person leaves the screen (figure 13(d)) and neither the bag nor the person is being located in the image. This sequence shows that the system seems to work on basic test sequence.



(a) Original image (b) Considered region (c) Probability inside the considered region

(d) Thresholded probabilities (e) Distance transform (f) Approximate bounding box

Figure 11: Luggage detection



(a) Original image (b) Probabilities (c) Thresholded probabilities

(d) Human detection (e) Remaining region

Figure 12: Removal of probabilities known to be generated by a human


(a) Bag and humans detected correctly

(b) Human 3 picks up bag, bag still detected


(c) Bag picked up, bag not being detected

(d) Human left scene with bag

Figure 13: Luggage detection test

## 0.4 Results

To detect security events and make information more appealing, an overlay was added to the image. The top left part shows the current frame number, and the bottom part has three colored indicators for the three types of events to be detected.

### 0.4.1 Loitering Detection

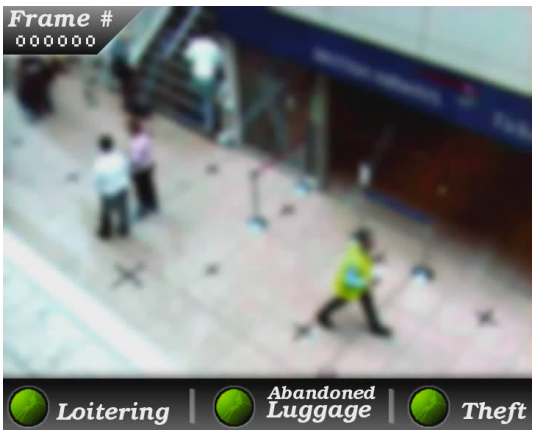
Loitering is to be detected in scenarios one and two. It was correctly detected only in scenario two, where a few key frames are shown in figure 14. The yellow loitering alarm is set when a person loiters for 750 frames, or half the time defined as loitering in the PETS instructions. The red loitering alarm is set only after a person is detected for 1500 frames, equivalent to one minute. The reason why the program does not properly detect loitering in scenario one is that there are many people walking across the screen. As mentioned before, when two humans cross path, the program has difficulty distinguishing between the two and their labels often get swapped. This prevent the program from detecting the loiterer for the full 1500 frames required.

As there is no ground truth available, it is hard to assess the accuracy of the detection. Noting that, in the second scenario, the loitering alarm (red) is activated after 1726 frames, it is possible to determine that the loitering human was first detected between frames 226 and 227. Figure 15 shows that in frame 227, the human labeled 0 is first detected. One can notice that the program only detects the person once a large portion of the body is visible on screen, and so the true frame number when the loitering alarm should be activated is probably a bit before frame 1726.

### 0.4.2 Abandoned Luggage Detection

The greatest problem when detecting bags is that if the bag is too large, it can generate enough foreground probabilities to be detected as a human. Such a case happens in scenario 8, as seen in figure 17(a). As detected bags are static objects, if a bag is detected as a human, it eventually leads to the activation of the loitering alarm.

The second problem about trying to detect abandoned luggage is that bags are detected as blobs on the two-dimensional image plane. According to the PETS definition of abandoned, the state of the alarm depends on the distance between the owner and the bag. As we do not know the position of the luggage in three-dimensional space, determining the distance between a bag and a

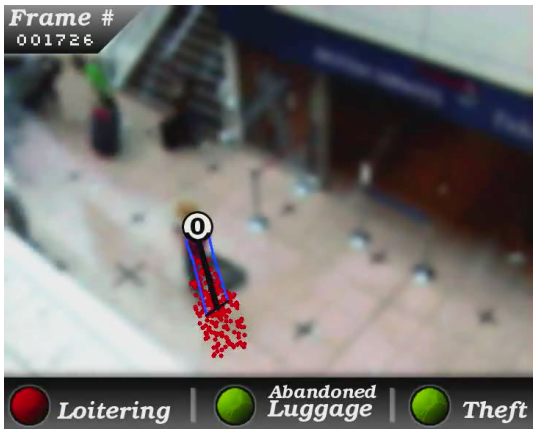



(a) Frame 0, no alarm


(b) Frame 975, no alarm


(c) Frame 976, yellow loitering alarm


(d) Frame 1725, yellow loitering alarm




(e) Frame 1726, red loitering alarm

Figure 14: Some key frames from the second scenario, showing loitering detection in action


(a) Frame 226 (b) Frame 227

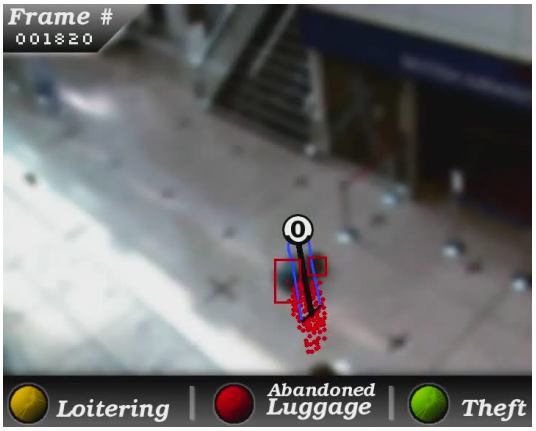
Figure 15: First detected frame of the loiterer from scenario 2

human is difficult. It would be possible to find the real world coordinates of the bag, but it would involve some fairly complex computations. For this reason, the measure of distance between bag and human was made on the two-dimensional plane. While not being as accurate, it provides a good enough approximation for the present project. It was decided that a bag is abandoned when none of the detected humans has a projected floor position inside a radius of 50 pixels from the centroid of the bag. Figure 16 shows that the alarm is properly set when an abandoned bag is detected.

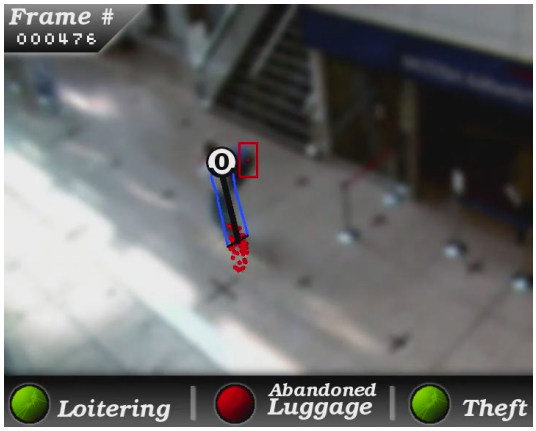

(a) No luggage, no alarm (b) Abandoned luggage, alarm set

Figure 16: Proper function of the abandoned luggage alarm

False alarms can happen in many cases. One such case is when a human is carrying a piece of luggage and stays idle for too long. This situation happens in scenario 8, as shown in figure 17(b).




(a) Piece of luggage being detected as a human




(b) False abandoned luggage alarm

Figure 17: Problematic cases of the luggage detector

## 0.5 Conclusion

By carefully reviewing the results presented earlier, it is clear that the method presented in [9] is valid as it has been shown to work on quite a few cases. It has also been shown that the method might not be robust enough to be usable in general cases, but that once all of the parameters and threshold values are determined, the results are sufficient. Overall, while not entirely solving the surveillance problem, the combination of Bayesian learning and particle filtering method presented here is a step in the right direction for automated security systems. The report will conclude with figure 18, which demonstrates proper detection of four distinct humans and one piece of luggage. This shows how promising such systems are.


Figure 18: Proper detection of four distinct humans and one piece of luggage

# Bibliography


[1] M. Bertozzi, A. Broggi, A. Fascioli, and A. Tibaldi. Pedestrian localization and tracking system with kalman filtering. pages 584–589, 2004.

[2] M. Bertozzi, A. Broggi, M. Felisa, G. Vezzoni, and M. Del Rose. Low-level pedestrian detection by means of visible and far infra-red tetra-vision. pages 231–236, 2006.

[3] Robert Bodor, Bennett Jackson, Nikolaos Papanikolopoulos, and Human Tracking. Vision-based human tracking and activity recognition. pages 18–20, 2003.

[4] C. Dai, Y. Zheng, and X. Li. Layered representation for pedestrian detection and tracking in infrared imagery. 2005.

[5] B.K.P. Horn. Tsai's camera calibration method revisited. 2000.

[6] M. Isard and J. MacCormick. Bramble: A bayesian multiple-blob tracker. 2001.

[7] H. Ryosuke and S. Nobuaki Ishihara. A novel system for tracking pedestrians usingmultiple single-row laser-range scanners. 2005.

[8] M.I. Shamos and D. Hoey. Geometric intersection problems. *Foundations of Computer Science, 17th Annual Symposium on*, pages 208–215, 1976.

[9] M. Spengler and B. Schiele. Automatic detection and tracking of abandoned objects. 2003.

[10] J. Sullivan, A. Blake, M. Isard, and J. MacCormick. Bayesian object localisation in images. *International Journal of Computer Vision*, 44, 2001.

[11] R. Willson. Tsai camera calibration software, October 1995.