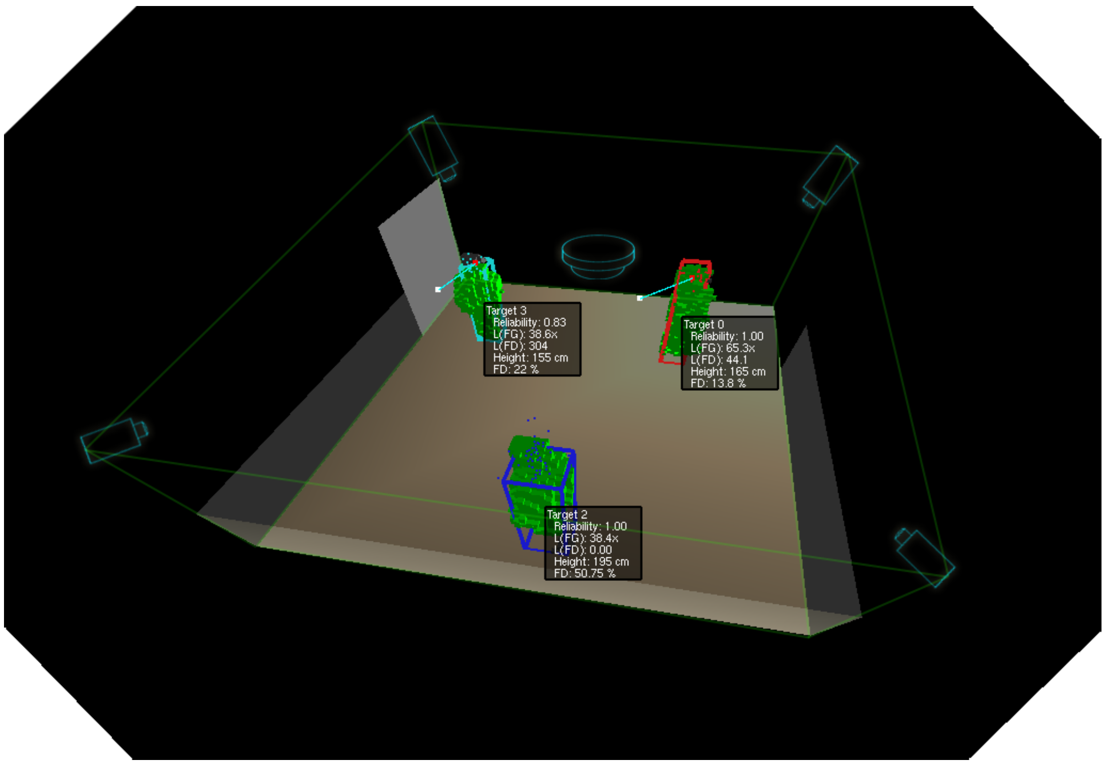
Local Positioning System
Based on Face Detection



AALBORG UNIVERSITET

Aalborg University
Institute of Electronic Systems
Vision, Graphics and Interactive Systems
Project Group 926, 2010

Institute of Electronic Systems
Fredrik Bajers Vej 7
DK-9220 Aalborg Ost
Phone +45 99 40 86 00

AALBORG UNIVERSITET

Title:	Local Positioning System Based on Face Detection
Theme:	Vision, Graphics and Interactive Systems
Project period:	September 6th to December 22nd 2010

Project group:
926

Group members:
Antoine Breton
Audrey Cabec
Océane Esposito
Grazina Laurinaviciute
Alexandre Majetniak
François Rosé

Supervisor:
Zheng-Hua Tan

Co-Supervisor:
Aristodemos Pnevmatikakis

Abstract:
The problem of sensing and tracking people is one which has encouraged much research. One growing need and application to this kind of system is the supervision of living environments for elderly.
In this project, we base our system on an existing one that tracks persons in a multi-camera environment, where tracking is based on two different modalities: foreground and feature points. In our system, we keep the information provided by the foreground and add a new modality: face.
Firstly, foreground is found in 2D on each camera frame and then converted to generate voxel-based 3D foreground. The face detection algorithm is then run, also for each frame, in order to confirm the existence of a human target. Finally, a particle filter based on partitioned sampling is used to combine information from both modalities.
The system is implemented to run on a single computer using video recordings. Tests will be performed on the CLEAR 2007 data set in order to compare it with the previous system. Results from the CLEAR 2007 data set are not presented in this report but the renderer of our system is described in details and can give an idea of how the system runs.

Copies:	3
Pages:	65
Finished:	20th Dec. 2010

Preface

This report documents group 926's work on the 9th semester of the *Vision, Graphics and Interactive Systems* specialisation at the Institute of Electronic Systems, Aalborg University. The work was done during the period from September 6th to December 22nd 2010.

The report is divided into 5 parts: *Introduction*, *Modalities*, *Tracking System*, *Implementation* and *Evaluation*. The first part motivates the project and describes existing systems for face detection. It concludes in a problem formulation. The different modalities that will be used in the system are then described in the second part. The design of the tracking system is explained in the third part, and its C++ implementation in the fourth. In the last part, the performance of the system is evaluated and a conclusion is drawn.
A bibliography listing all the relevant literature sources can be found at the end of the report. The references are made using the syntax [number].

We would like to thank our supervisor at Aalborg University Zheng-Hua Tan for allowing us to work on this project. We would also like to thank our co-supervisor Aristodemos Pnevmatikakis from Athens Information Technology (AIT) for his assistance throughout the project and for accepting to test our system on the CLEAR data set.

Antoine Breton

Audrey Cabec

Océane Esposito

Grazina Laurinaviciute

Alexandre Majetniak

François Rosé

Aalborg, December 22nd 2010

Contents

Part I

Introduction

Contents

This part of the report presents our motivation and the need for many face tracking applications. It also defines our project goal by a problem formulation. It explains the most relevant face detecting systems which already exist and the testing part with CLEAR Data Set.

Chapter 1

Motivation

People tracking is a process which can be used in very different domains like surveillance, augmented reality, traffic control and medical imaging. The common goal is to allow an automatic system to sense the presence of people somewhere in a room. It can assist people in their work by making it easier. However, it is a very long process because of the amount of data in a video.

Security systems are often used for monitoring people. They usually use a single camera with very simple algorithms which can detect if a person is moving in a scene. This basic system can be improved by a simple idea: when a specific event occurs, an alarm is triggered and it can be dealt with in a particular way. Finally, if we are able to differentiate between several events in a video, we can deal with them in many different ways. The more efficient the algorithm is, the less staff you need to secure a place.

However, the goal of this application is to assist elderly people in their everyday life. It is very important to help these people because their number is increasing significantly in the developed countries. Most of the time, old persons live alone and it can be a big problem when they get hurt at home. This program tries to improve their security and guarantee their independence.

When an elderly falls over, most of the time he or she cannot reach the phone to call for help. Fall detection is one of the goals of this project. It is achieved by tracking the position and determining the mobility of this person on a scene. Knowing the body posture in real-time, we can provide help instantaneously for someone in trouble.

In order to have an efficient system, we need clear information about the scene. In most cases, a single camera is used, which is not sufficient since an obstacle can easily obstruct a person. Passive sensors can be used to get more information but cameras provide more information about what is happening on a scene. This system uses five cameras and their information is combined to track a person and determine its body posture.

We decided to work on an existing project and improve it to make it more reliable. This project has been realized by Martin and Rasmus Andersen [1] during their last year at Aalborg university.

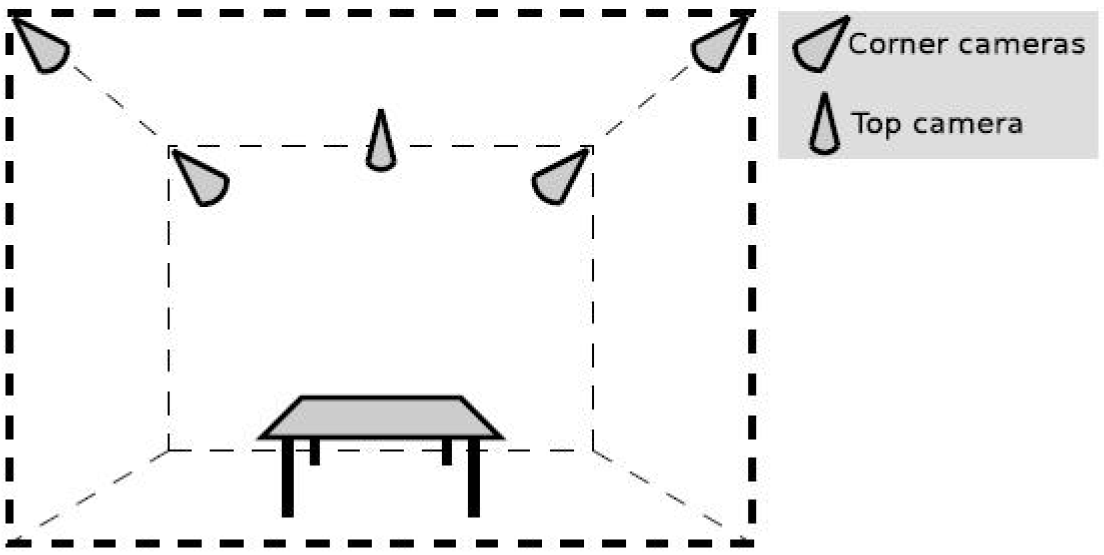


Figure 1.1: Camera Setup [1]

1.1 Setup and Data Set

The system is composed of 5 cameras. 4 of them are placed on top corners of the room and the last one is located in the middle of the ceiling, pointing down. An illustration of the room is shown in Figure 1.1.

1.1.1 Development Data Set and Setup

A test setup was established in the laboratory of Athens Information Technology (AIT) to assist in the development process. The 4 corner cameras resolution is 1600x1200 and the top camera's resolution is 1024x768. Each camera's recording rate goes up to 15 fps. One video sequence of 6.57 min duration has been used in order to test our system. All cameras are connected to network computers.

1.1.2 CLEAR Data Set

CLEAR 2007 data set [6] is chosen to produce results that can be compared to performance of previously developed systems. This international workshop contains 40 video recordings of meeting rooms, equipped with five cameras like in Figure 1.1. A number of background frames without people in the room are available for each recording. Recordings are taken from five locations and each one is approximately 5 minutes long. In each video we can see people moving around, sitting at a table and falling.

CLEAR 2007 workshop defines how to measure the performance of a system. It uses four different measurements:

- *Misses*: An object is missed if it is not tracked whithin 50 cm accuracy.

- *False positives*: A false positive is yielded when we are more than 50 cm wrong.
- *Mismatches*: A mismatch occurs when an object switches to another one.
- *Position error*: An object has to be tracked within 50 cm to be declared "matched".

  These four measurements are combined to obtain the MOTP. This shows the precision of the tracker system. It also gives a measure of tracker's configuration errors - false positive, misses and mismatches.

Chapter 2

Modalities and Tracking Systems

This part is going to give an overview of some measurements that we can use to make an efficient tracking system. In a second part we are going to discuss about the best modalities that our system can have and to conclude, we will make a choice about our tracking system concerning these modalities.

2.1 Existing Tracking Modalities

There are a lot of different modalities that can be used in a tracking system. They can be divided in two distinct categories: sound and visual.

Sound
When a person moves in a room, it interacts with other persons and with the environment by making sounds. Thus, by using multiple microphones, this person can be tracked. Some examples of tracking systems using noise are shown in [30] [14] [20].

Visual
There are four different characteristics that can be used in an image.

- Firstly, it is possible to track a person using the color [28]. The most relevant color-based method is the one which uses Gaussian distribution in a color mixture model.
- Secondly, the contours of a person is an effective mean for detection, as seen in [35]. This method presents a big problem because the outline of a person can be very easily occluded when the person is behind a chair for example.
- Thirdly, tracking of people can be accomplished using face detection [37] and recognition. In this paper, they are using a skin-color model to detect human faces in a video. Then, to track a person, a motion model and a camera model are used respectively to handle head motion and to predict camera motion. Faces are only

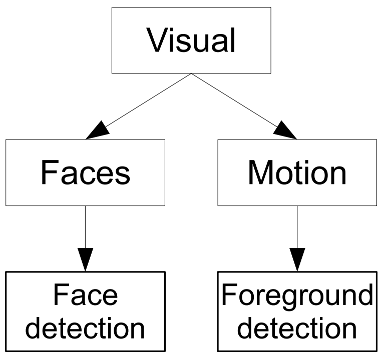


Figure 2.1: Modalities that we are using in the system.

present when real people are in the room into consideration and this kind of detection can make the difference between an object and a person when both are moving. However, some false positives can appear. The best face detection algorithm has been developed by Paul Viola and Michael Jones and is based on a boosted cascade of simple classifiers [33].

- The last visual cue to detect a person in a room is motion and they are used in [5] [11]. Two methods have been developed to find motion in a video. For the first one, a frame by frame difference is computed to obtain a foreground detection. The second method uses the optical flow and feature points. The optical flow is defined by the motion of some relevant points on a scene. By tracking these points, it is possible to track a person in a room. These techniques have a limit because when a person is not moving, he or she can not be detected.

2.2 Choice of Modalities

To have the best person tracking system, we have to combine at least two modalities. This is a very important part in the project because we have to choose the modalities very carefully, otherwise the system can rapidly become very slow.
The first type of cues, the sound modality, is discarded because we want to have a purely vision-based system. We are improving an existing system, thus we must deal with previously made choices. The foreground detection is kept, as it is an effective cue, and it works fine in the previous system. It will be the item for the motion category. On top of that, we decided to implement another visual item: face detection. These choices are represented in Figure 2.1.

To conclude, we will combine **foreground detection** and **face detection** to track persons in a room in order to improve the previous project which combined foreground detection and feature points.

Chapter 3

Face Detection

Before developing the system for face tracking, it is important to know a fundamental computer vision problem which is face detection. Over the past few years and with the rapid increase of computational powers, lots of advances have been done in the different approaches of this topic in computer vision literature. In this chapter, an overview of the different existing techniques in face detection is presented.

3.1 Face Detection Problem

Face detection recently aroused a growing interest from researchers [38] [40] [32] [34] because of the multiple applications using it: biometrics, video surveillance, energy conservation... In his paper [38], Ming-Hsuan Yang gave a definition of face detection: "*Given an arbitrary image, the goal of face detection is to determine whether or not there are any faces in the image and, if present, return the image location and extent of each face.*"
Face detection has been resolved by using a lot of different approaches which can be grouped into three categories:

- the knowledge and template-based technique will be presented in section 3.2
- the feature invariant approach will be discussed in section 3.3
- the appearance-based technique will be detailed in section 3.4

The last part (3.5) will talk about the algorithm developed by Paul Viola and Michael Jones. This method revolutionized the face detection research field.

3.2 Knowledge and Template-Based Techniques

The knowledge-based and template-based techniques are two very close methods. The knowledge-based one encodes what constitutes a typical face by rules referring to human

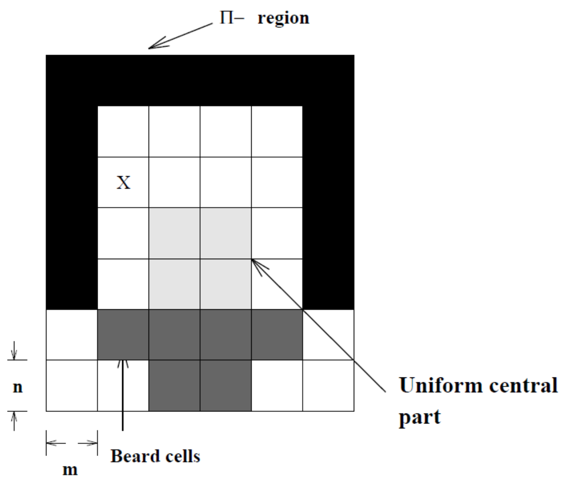


Figure 3.1: Abstract face model at the resolution level of the quartet image [16].

knowledge; for example all faces have two eyes, symmetric to each other, a nose and a mouse. It is also known that the center part of a face has uniform intensity values and the difference between the average intensity values of the center part and the upper part is significant. The template one uses a storage of several standard patterns to describe the face as a whole. They are both used mainly for face localization. Yang and Huang proposed a method using these approaches for face detection [36]:
"*First step: apply the rule* ***the center part of the face has 4 cells with a basically uniform intensity*** *to search for candidates. Second Step: local histogram equalization followed by edge equalization and then edge detection. Third Step: search for eye and mouth features for validation* ".
This is a simple method but it remains difficult enumerating templates for different poses.

Kotropoulos and Pitas [16] proposed an abstract model for the face at the resolution level of the quartet image. It is shown in Figure 3.1. A hierarchical knowledge-based system is then designed in order to detect facial features by establishing rules applied to the quartet image.

Face detection using knowledge-based method or template-based method is easy to implement with simple rules and it is well appropriate for face localization in tidy background. However it is difficult to translate human knowledge into precise rules: detailed rules fail to detect faces and general rules may find many false positives. Besides, it is really complicated to extend this approach to detecting faces in different poses because it is impossible to enumerate all the possible cases.

3.3 Feature Invariant-Based Technique

Feature based methods detect faces by using facial features and a combination of them. Facial features (eyes, eyebrows, nose, and mouth) are extracted using image processing

operations such as thresholding, edge detection or morphological operations. Relationships between them are described by statistical models built after the extraction of the features, and then the existence of a face is verified. Neural networks, graph matching, and decision trees can also be used to verify candidate faces. There is another powerful feature which can be used for face detection: skin color, which is considered as an invariant and effective feature. Normally, each person has her own skin color but research has shown that the main difference between human skin colours was the difference in the lighting intensity component. Thus faces can be detected using a color space such as HSV or YCrCb.
In their article [19], Lanitis, Taylor and Cootes model the shape of facial features. In a new image, facial features are located using an Average Search Model. This process is shown on Figure 3.2:

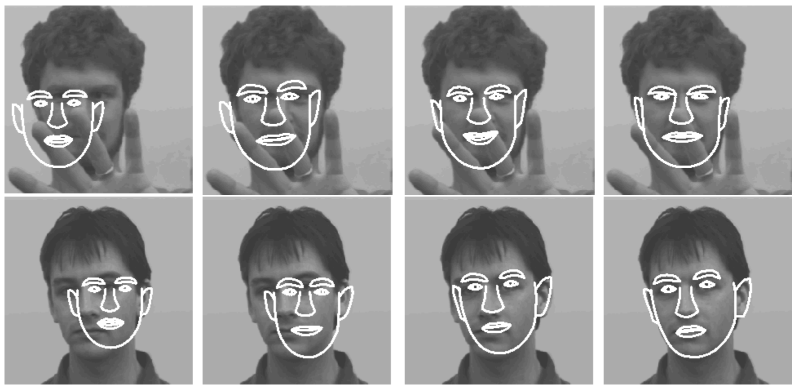


Figure 3.2: Examples of the Average Shape Model fitting procedure [19].

The feature invariant-based technique presents the advantage that features are invariant to pose and orientation change. Therefore it is very difficult to locate facial features due to several corruptions like illumination, noise or occlusion and detecting features in a complex background is not an easy task either.

3.4 Appearance-Based Technique

In contrast to template matching, appearance-based methods [7] use a huge number of examples such as images of faces or facial features, with different variations such as face shape, eye and skin color... The problem of face detection is to know whether there is a face or not in the image, so we can consider it as a classification problem with two classes: non-face and face. The non-face class contains images that have representations of anything that is not a face, while the face class is for all face images. Statistical analysis and machine learning techniques are used to find the statistical properties of the pixel brightness patterns on images belonging to both classes. To detect a face in an input image, the whole image is scanned and image regions are identified as "face" or "non face" based on these probability functions. Well-known appearance-based methods used for face detection [27], [24], [22] are eigen faces , LDA , neural networks, support vector machines and hidden Markov models.

3.5 The Viola and Jones Method

Paul Viola and Michael Jones developed a very successful appearance-based face detection algorithm [33] based on three main ideas: the integral image, classifier learning with AdaBoost, and the attentional cascade structure. It is one of the first techniques which can effectively detect faces in an image in real time. The algorithm consists of scanning the whole image and calculating some characteristics in overlapped rectangular areas. One particularity of this method is the fact that a lot of characteristics are used but they are very simple. The first innovative idea is to introduce integral images which enable to have a faster calculation of the characteristics. The second innovative and important idea is the selection of these characteristics by using a boosting algorithm - AdaBoost, representing the characteristics as classifiers. Then, the Viola and Jones method combines the boosted classifiers into a cascade process which is faster and more efficient for the detection.
To sum up, the method is divided into two steps: the first one is training the classifier, based on a huge number of positive examples (that is to say, objects of interest: faces, for example) and negative examples. This training step is done only once and is performed off-line. The second step for the detection is applying this classifier to unknown images.

We decided to use this method in our project to implement the face detection part because it is very efficient and the method is mentioned in literature a lot [39] [29]. Besides, it is implemented in OpenCV under a BSD license so we can use it in an easy way.

Chapter 4

Problem Formulation

4.1 Problems

The actual tracking system works but needs some improvements because in specific cases, some problems occur. In many situations, the system can lose a target very easily.

Firstly, when a person stays immobile for a period, the target is lost by the foreground tracker.

Secondly, when a person is faded into the background but still marked by feature points, the tracking system has some issues. Namely, the feature point tracker keeps the track without a problem, but it fails when a person is mobile nearby, because the foreground and the feature points parts of the tracker try to reach two different goals. While the foreground part tries to move the target to the area with foreground, the feature points part tries to keep the target at the originally tracked person. The feature points part always loses this fight because feature points located far away from the target are erased.

Thirdly, the initialisation procedure can be improved to avoid initialising targets on moved objects like chairs. Finally, when many people are very close to each other, they overlap in most camera views. No points are initialised in this situation to avoid initialising feature points on a wrong person.

With the CLEAR data set, we are going to be able to compare our algorithms to the previously developed ones. It means that this system must be able to succeed in these tasks in typical situations. The CLEAR metrics define what a good performance is:

- **Entering and leaving**: In our system, we only allow the fact that people are entering and leaving the scene with nobody on the first frame. The system does not support the situation where people are present in the first frame.
- **Mobility**: The system must be able to track a person, irrespective of whether he is moving or not.
- **Multiple persons**: The system has to track up to 5 persons successfully.

- **Precisions**: The tracking system needs to give a precision of 5-8 cm.
- **Real-time execution**: The system must be able to run in real-time.
- **Reasoning**: Body posture estimation has to be carried out simultaneously with person tracking.

Problem formulation:

How can a system be developed that, based on combined information from adaptive foreground estimation and face detection, is able to track multiple persons in a multi-camera environment and reason about their mobility and body posture?

Part II

Modalities

Contents

As our goal was to modify and improve the previous project realized by Martin and Rasmus Andersen [1] we based a significant part of our project on their work. One main part is foreground detection. In order to implement this type of detection, a lot of methods based on pixel background models have been developed. In this part, an overview of previous realizations on foreground detection is presented. Moreover, we describe a face detection approach, designed by Viola and Jones, which we combine with foreground detection to improve the system.

Chapter 5

Foreground Estimation

Foreground processing has already been done by Martin and Rasmus Andersen [1] and this chapter is a short analysis of it. The term foreground means that there are non-stationary persons or objects in the scene. Most of the methods for the foreground estimation in 2D are based on pixel background models [9] [11], which means that the system does not distinguish objects in the scene, but distributions of pixels. Those pixels are analysed, then noise and shadows are handled afterwards. The 3D representation of foreground can be realized by combining several 2D foreground masks and using preprocessing with combination of informations from the cameras. The last step enables to reduce efficiently the amount of noise produced with the method used to calculate the 2D foreground. The main advantage of using several cameras for tracking comes from allowing to track on the floor plan in dimensions that are independent of the camera positioning.

5.1 Foreground Estimation in 2D

5.1.1 Adaptive Background Estimation

Two approaches exist for the adaptive background estimation: parametric and non-parametric [21]. The first one assumes that the background is distributed as a predefined distribution whereas the second one can handle arbitrary distributions. Our system uses five cameras to work, it means that we have a huge amount of data to calculate so, we need an adaptive background which does not require too many memory to store the data so the parametric approach is chosen.

5.1.2 Gaussian Mixture Model (GMM)

A mixture model represents each RGB pixel by several simple distributions such as a Gaussian distribution. The Gaussian Mixture Model is the most used mixture model [3] and it is generally chosen for the basis distribution in the mixture model, besides the Gaussian distribution gives better results than the Laplacian one [15]. Using mixture models pixels means that only one distribution is going to be used to model the background color. For the foreground objects and persons which are passing in front of the background, some additional distributions are applied.

5.1.3 The GMM Background Estimation Algorithm

The first step of the algorithm is to find if a pixel value matches a distribution. So, for each new frame, each color of a pixel is compared against the distributions of the pixel. If the color belongs to the existing distribution, the pixel is matched to it. The second step is to update the weight of the distribution and sort it according to the background likelihood. Each pixel is marked as foreground or background. The background distribution will be present at each step of time whereas the foreground could have some huge variance.

For this algorithm, it is not required to use high resolution images. By using lower resolution images, the GMM algorithm runs significantly faster and reduces the time consumption.

5.1.4 Shadow Removal

In this section, the shadow removal algorithm is explained. Most of the shadow has to be removed because it is considered as noise in the image. To achieve this goal, a "per-pixel" shadow removal is performed. To handle shadows, we use separate channels for brightness and colour informations. Only the pixels marked as foreground are used. To remove the shadow, we need to introduce two different values: D_b and D_c. D_b is the brightness distortion and it is a scalar which indicates the brightness of a particular pixel compared to the background pixel. D_c is the colour distortion and it is a scalar which indicates the colour difference between a particular pixel and the background pixel. τ_{D_b} and τ_{D_c} are respectively D_b and D_c thresholds. The thresholds are meant to prevent too many dark pixels from getting marked as shadows.

Shadow removal algorithm [10]:

For each foreground pixel:

- Calculate D_b between the most likely background model and the pixel
- If $1 > D_b > \tau_{D_b}$ then
  - If $D_c < \tau_{D_c}$ then a shadow pixel is found

5.1.5 Noise Removal

The last task for 2D foreground detection is noise removal. Different approaches exist to remove the noise such as neighbourhood averaging [8], BLOB analysis [31], and a pixel persistence map (PPM) combined with binary decision trees [18]. In our case, the BLOB analysis is used because it is the simplest and the fastest technique. Indeed, it provides good results in many situations. The limitation of this technique is that sometimes it detects false background when the background has high similarities with the foreground. In this system, the BLOB analysis removes small areas of foreground and background. A BLOB which has an area with less than a predefined threshold is removed.

5.2 Combination of 2D Foreground Masks into 3D Foreground

First of all, the way of spanning the 3D space has to be chosen. It can be both spanned in a discrete [17] or a continuous way [4], but due to its simplicity and great effectiveness for removing the noise, the discrete voxel-based representation of the 3D space is chosen.

Secondly, voxels are projected to the image planes of all cameras and then the corresponding foreground masks are checked. If there are enough cameras detecting significant foreground, these voxels can be considered as foreground. To do that, three possibilities have been analysed:

1. **A solution based on centre pixels**: the center pixel of the voxel indicates whether or not the projected voxel mostly contains foreground, which in other terms means, if it is located on the border of a foreground area, on the foreground area, or out of it. Testing only one pixel saves computation time but is very sensitive to noise since one affected pixels yield to a wrong conclusion.

2. **A solution based on blurring filter**: before using the center pixel technique, a blurring filter is applied to the foreground masks. Its purpose is to add resistance to noise. However, the blurring filter produces a kernel for each voxels and the kernel is gathered around the centre pixel. Because of camera distances issue and various voxel sizes, the kernel might not have the correct size, and therefore will not be located on the foreground, which leads to false detection.

3. **A solution based on distance transform**: distance transform is the alternative solution to the blurring filter. The center pixel is still used instead of the whole voxel, but this time, the nearest foreground is found using a circle enclosing the projected voxel. It still leads to false positives but remains more efficient. It does not include noise reduction.

Noise is already suppressed during combination of different cameras into 3D. That way, the blurring filter is not needed, and distance transform becomes the best choice.

5.2.1 Hierarchical Grid Structure

To represent foreground in 3D space, coherent volumes have been structured to indicate the presence of people. To avoid having to test all the voxels to determine if they intersect foreground, the idea is to divide space into hierarchies of voxels. Testing a large voxel, if it contains foreground, its children will be tested. If not, they are skipped.

Algorithm for doing the conversion of the 2D foreground masks into a grid of foreground voxels [1]:

- We are using a grid of voxels on N hierarchical levels to span the room.
- Each center and corner of 3D voxel are projected on all levels to the image plane of each camera. The radius of the enclosing circle C is determined by the corners.
- **IF** a voxel can not be seen by a sufficient number of cameras **THEN** remove it.
- **FOR** each camera: Perform distance transform of the foreground mask.
- Assume that S is a group for all voxels of the highest hierarchical level.
- **FOR** each voxel in S:
    1. **FOR** each camera:
        - **IF** the value of the center of the projected voxel in the distance map is below the radius of C
        - **THEN** foreground is detected
    2. **IF** all cameras that can see the voxel detect significant foreground:
        - **IF** the voxel has any child
        - **THEN** repeat the algorithm with S consisting of all children of the voxel
        - **ELSE** mark the voxel as a foreground voxel

Chapter 6

Viola Jones Face Detection

This chapter describes the Viola-Jones algorithm providing face detection which we add to the existing system [1].

The algorithm, presented in [33], provides a solution to the problem of spotting image regions that contain faces. The algorithm takes a grey scale image as input and produces a set of rectangles on output, each of which, in an ideal case, corresponds to a human face in the supplied image.

The whole procedure is based on machine learning, whereby in the beginning a large data set of images is used to train the algorithm, i.e. make it acquire the necessary criteria for classifying any arbitrary input image region either as a face, or as a non-face region. Criteria take the form of features, which can be calculated from a single input image region, and their corresponding thresholds.

The trained algorithm is then applied to an input image, analysing its subregions of different sizes and positions. Some of them are accepted as faces and constitute the output of the algorithm.

6.1 Features

Decisions about face presence in the image are taken based on image features, that can be computed for every image subregion. The features used in the Viola-Jones face detection algorithm are reminiscent of Haar basis functions. It classifies images based on the value of simple features. There are three varieties of them: two-rectangle, three-rectangle and four-rectangle features. These are shown in Figure 6.1.

- The value of a 2 rectangle feature is the difference between the sum of the pixels within two rectangular regions.
- The value of a 3 rectangle feature is computed by the sum within 2 outside rectangles subtracted from the sum in a center rectangle.

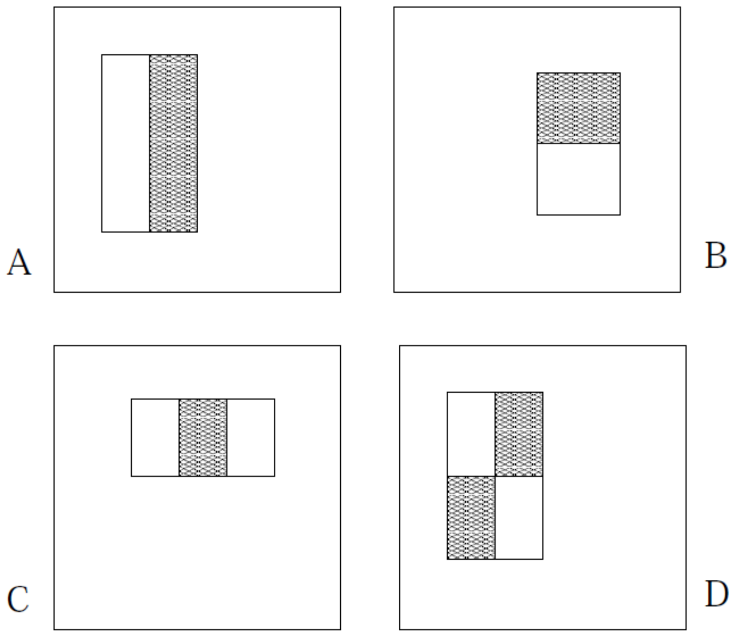


Figure 6.1: A and B show a two rectangle feature, C illustrates a three rectangle feature and D is a four rectangle feature. The sum of the pixels which lie within the white rectangles are subtracted from the sum of pixels in the grey rectangles [33].

- The value of a 4 rectangle feature is computed by the difference between diagonal pairs of rectangles.

They can be efficiently computed due to an innovative image representation, called an integral image.

$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y') \tag{6.1}$$

6.2 Training of the Algorithm

Now that the features have been defined, the training of the Viola and Jones algorithm can be described. This requires a data set of N images x_i, each having a label y_i assigned to it. $y_i = 1$ when an image is considered a face, and $y_i = 0$ otherwise.

6.2.1 Weak Classifier

Central to understanding the performance of the Viola and Jones algorithm is the concept of a binary classifier. It is a function which either discards or accepts its input variable (this corresponds to values 0 or 1, respectively). Dealing with images of faces, an image region is considered as a variable, and its acceptance or rejection means that it either contains a face or not.

By the term weak classifier we refer to a single feature and its corresponding optimal threshold. The threshold is learned as follows: first, the feature value is computed for each image x_i from the training data set. Based on this computation, the image set $\{x_i\}$ can now be divided into two parts - negative and positive, according to some threshold applied to the feature value - and the resulting division can be compared with the original labels y_i. The threshold we are looking for is the one that leads to misclassifying the minimum number of example images.

6.2.2 Strong Classifier

Weak classifiers, taken individually, cannot be trusted enough, when a decision about face presence in the image region is taken, as they tend to misclassify input significantly. However, their appropriately chosen linear combination - a classifier itself - performs the task much better. This combination is called a strong classifier and is obtained by applying the Ada Boost algorithm to the training data set.

6.2.3 Ada Boost Algorithm

The Ada Boost algorithm takes a feature set and a training set as input and produces a strong classifier as output. In the beginning, each feature's optimal threshold is chosen, as described in 6.2.1.

Afterwards, a sequential process of T rounds is run. Each round selects a single feature out of the initial set. The selection criterion is the minimum classification error with respect to weights, assigned to the data set items.

The algorithm is presented in Figure 6.2. Each round starts with a set of normalized weights, associated with each image x_i, marking their relative significance in this round. Higher weights correspond to the images that were misclassified in the previous round(s), which leads to subsequent selections of a classifier that compensates for previous mistakes.

Each feature's threshold divides the data set into two parts and this division together with image weights and true classification labels y_i is used to obtain a classification error for this feature in the current round. The smallest classification error leads to its corresponding feature being selected in this round.

The process results in a strong classifier, expressed as a linear combination of T weak classifiers.

6.3 Cascade Structure of the Algorithm

Usually, for each input image, there is a huge number of subregions to analyse and the great majority of them does not contain faces. However, the right regions should be selected as quickly as possible. Therefore it is necessary not to spend too much time on dealing with irrelevant subregions.

- Given example images $(x_1, y_1), ..., (x_n, y_n)$ where $y_i = 0, 1$ for negative and positive examples respectively.
- Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively, where m and l are the number of negatives and positives respectively.
- For t = 1,...,T:
  1. Normalize the weights
     $$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$$
     so that w_t is a probability distribution
  2. For each feature j, train a classifier h_j which is restricted to using a single feature. The error is evaluated with respect to $w_t, \varepsilon_j = \sum_i w_i \mid h_j(x_i) - y_i \mid$.
  3. Choose the classifier h_t, with the lowest error ε_t.
  4. Update the weights:
     $$w_{t+1,i} = w_{t,i} \times \beta_t^{1-e_i}$$
     where $e_i = 0$ if example x_i is classified correctly, $e_i = 1$ otherwise, and $\beta_t = \frac{\varepsilon_t}{1-\varepsilon_t}$.
  5. The final strong classifier is:
     $$m(i) = \begin{cases} 1 & \text{if } \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$
     where $\alpha_t = \log \frac{1}{\beta_t}$

Figure 6.2: AdaBoost Algorithm [33]

In order to achieve this goal, given the particularities mentioned, a cascade structure of the Viola-Jones algorithm is chosen, meaning that each input region is processed consecutively by a number of nodes, called strong classifiers. Each node is a binary classifier and performs the same task - either classifies the region as a face or rejects it - however, their criteria differ. Nodes further down the cascade will analyse a subregion only if it has not been rejected by the previous ones. The nodes are arranged in the order of increasing complexity, so that the front ones spend less time analysing a single image subregion and therefore determine more quickly if a face is present. Strong classifiers further down the cascade, on the other hand, inspect each subregion more thoroughly before taking a decision. The result is as desired: less time is spent dealing with irrelevant regions, as they are discarded in the beginning of the cascade, and those that pass successfully to the end of the cascade are most likely to contain faces and are given the most attention.

Only subregions having traversed the whole cascade successfully are finally acknowledged as faces and are further used in our system in face likelihood computations.

Part III

Tracking System

Contents

The purpose of our system is to be able to track the centroid of a person's head present in the room. This goal is achieved by making use of the statistical method called **particle filter**. Each person in the scene being observed is associated with a **target** - a representation, suitable for the purposes of tracking. Each target, in turn, has its corresponding **particle set**. This part is devoted to the discussion of these concepts.

Chapter 7

Particle Filter

Particle filter is an advanced statistical method used to estimate the state of a Bayesian model. This method uses the available observations and turns them into probabilities. Then the particle filter evaluates different state hypotheses, also called particles, and a model estimation is built. This method is an alternative to Kalman filter developed in 1960 by R.E.Kalman [12]. If the particle filter is correctly defined, it can be faster than a Kalman filter. Also Kalman filters are limited by some assumptions. For example, Kalman filter assumes that probability density functions are only Gaussian. On the other hand, a particle filter provides a more general approach, often more efficient to manage real word statistical models. In computer vision systems, particle filters are very useful to track objects and people using video data. The first system using particle filter in this domain was developed by Isard and Blake in 1998. This application is well known under the name CONDENSATION algorithm [2].
Our system cannot assume that probability functions are Gaussian. Moreover, a particle filter can be improved to combine efficiently several modalities using partitioned sampling. In this chapter we will describe the basic particle filter algorithm, then we will explain how we use partitioned sampling to combine foreground measurement with face detection.

7.1 Bayesian Probability, Theory and Notations

Bayesian probability is a way of using information from the past and current time step measurements to predict the current time state of a process by evaluating a probability density function. The mathematical notation for a state at time k is $\mathbf{x}_k$ and the observations at time k are denoted $\mathbf{z}_k$.
In Bayesian estimation, the true state is assumed to be the result of an unobserved first order Markov process [30], and the measurement is the observation corresponding to the true state. In other words, the state at time k depends only on the state at time $k-1$ and the measurement is related to the true state. Then some probabilities are defined:

- $p(\mathbf{x}_k)$ is the prior probability of $\mathbf{x}_k$, the probability of the state being equal to $\mathbf{x}_k$ before the data $\mathbf{z}_k$ is observed.

- $p(\mathbf{z}_k|\mathbf{x}_k)$ is the conditional probability of observing $\mathbf{z}_k$ given that the hypothesis $\mathbf{x}_k$ is true. This probability is also called the likelihood.
- $p(\mathbf{x}_k|\mathbf{z}_k)$ is the posterior probability, the probability that the state hypothesis is true, knowing $\mathbf{z}_k$.

The purpose of the Bayesian approach is to evaluate the posterior probability distribution $p(\mathbf{x}_k|\mathbf{z}_k)$. In general, algorithms based on this approach do it in 2 steps. First, the system gives a **prediction** using the previous state and data. Then it takes the newest measurement into account in the **update** step. These steps are described by the following equations based on Bayes'theorem 7.1.

$$\textbf{Bayes'theorem:} \quad p(\mathbf{x}_k|\mathbf{z}_k) = \frac{p(\mathbf{z}_k|\mathbf{x}_k)p(\mathbf{x}_k)}{p(\mathbf{z}_k)} \tag{7.1}$$

$p(\mathbf{z}_k)$ is a constant, so we can simplify Bayes'rule to: $p(\mathbf{x}_k|\mathbf{z}_k) \propto p(\mathbf{z}_k|\mathbf{x}_k)p(\mathbf{x}_k)$

$$\textbf{Predict:} \quad p(\mathbf{x}_k) = p(\mathbf{x}_k|\mathbf{z}_{k-1}) = \int p(\mathbf{x}_k|\mathbf{x}_{k-1})p(\mathbf{x}_{k-1}|\mathbf{z}_{k-1})d\mathbf{x}_{k-1} \tag{7.2}$$

The equation 7.2 holds because of the underlying first order hidden Markov model assumption. $p(\mathbf{x}_k|\mathbf{x}_{k-1})$ is sometimes called motion model or state evolution model and $p(\mathbf{x}_{k-1}|\mathbf{z}_{k-1})$ is the posterior probability at time $k-1$.

$$\textbf{Update:} \quad p(\mathbf{x}_k|\mathbf{z}_k) \propto p(\mathbf{z}_k|\mathbf{x}_k)p(\mathbf{x}_k|\mathbf{z}_{k-1}) \tag{7.3}$$

$p(\mathbf{z}_k|\mathbf{x}_k)$ is a likelihood, also denoted $L(\mathbf{x}_k|\mathbf{z}_k)$. This likelihood reflects the probability of seeing $\mathbf{z}_k$ for a given state $\mathbf{x}_k$.
In many cases the state space is huge and it is often impossible to obtain the true state because calculating $p(\mathbf{z}_k|\mathbf{x}_k)$ and $p(\mathbf{x}_k)$ cannot be done. The solution is to estimate $p(\mathbf{x}_k|\mathbf{z}_k)$. There are many algorithms used to solve this kind of problem. One of the most famous is the Kalman filter. It is used in many applications because it is a quite simple algorithm able to solve complex problems. The main problem of this filter is the assumption that transition and observation model are considered as Gaussian. An alternative to Kalman filter is the particle filter algorithm [23], also known in the computer vision field as "multiple hypotheses tracking". This filter is able to solve more complex problems that cannot assume the probability density functions being Gaussian.

7.2 Particle Filter Algorithm (SIS and SIR)

Particle filters are designed to estimate the state of a process using a Bayesian approach. They are recursive algorithms. The main idea of this method is to generate many hypotheses which are represented by the particles. Each particle consists of a state value and a weight obtained by evaluating a likelihood function for its state. Then, given all the particles, we use an estimator to get the state estimation.

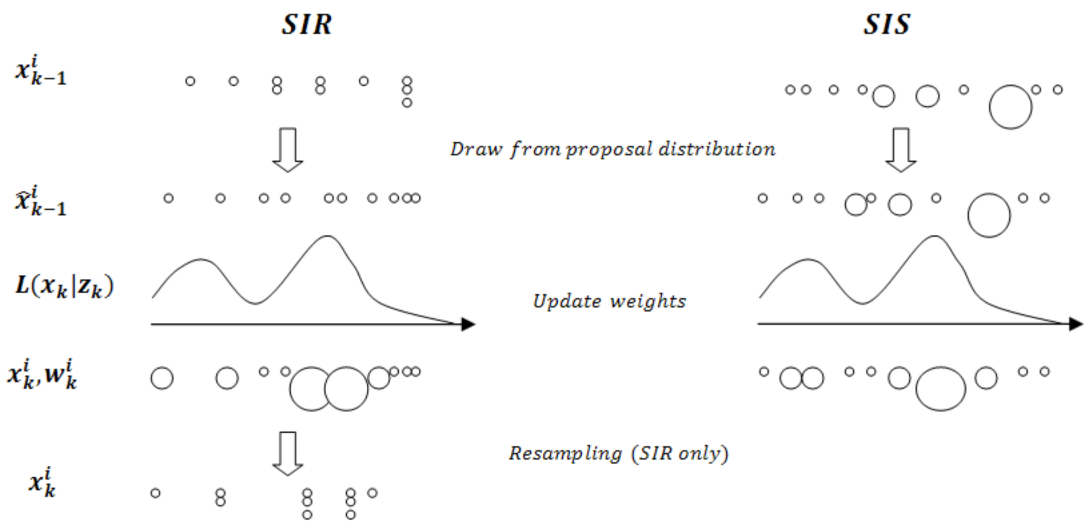


Figure 7.1: Illustration of SIS and SIR particle filters in 1 dimension. The particles are represented by circles and their size are proportional to their weights.

- **Prediction:** The first step is to generate a new set of particles for the current time step. Ideally the particles should be distributed according to the posterior distribution $p(\mathbf{x}_k|\mathbf{z}_k)$ but it is not feasible due to the requested calculation process. The solution is to use a proposal distribution $\pi(\mathbf{x}_k|\mathbf{z}_{1:k})$ instead. This approach is based on first order hidden Markov chain, which means the proposal distribution will only depend on the previous state and the measurements, i.e: $\pi(\mathbf{x}_k|\mathbf{z}_k) = \pi(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{z}_k)$. The definition of this proposal distribution affects the efficiency of the filter a lot.

- **Update:** For each particle a likelihood is calculated. Each particle's importance weight is updated by the equation 7.4. Then we normalize weights, i.e for each particles we divide its weight by the sum of all the weights.

$$\textbf{Update weights:} \quad w_k = w_{k-1}\frac{L(\mathbf{x}_k|\mathbf{z}_k)p(\mathbf{x}_k|\mathbf{x}_{k-1})}{\pi(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{z}_k)} \tag{7.4}$$

- **Resampling:** The purpose of resampling is to change the weight of all the particles to the same value $\frac{1}{Np}$, with Np being the number of particles. But it is important to make sure that resampling does not change the distribution described by the particles' weights. Therefore particles with higher weights are replaced by several identical particles and the particles with lower weights are deleted.
  The Sequential Importance Sampling (SIS) is the basic implementation of the particle filter without resampling. With SIS, the particle filter can easily produce one or several particles with very high weights and all the others will have very low weights. When this particular situation occurs, the particles do not represent the

posterior distribution correctly. This problem can be solved by occasionally resampling the particles when the problem appears. A good indicator of this problem is the variance of the importance weights. Thus it is possible to set a maximum acceptable value for this variance and resample when the variance is above this threshold.
Another common approach for particle filtering is to resample at each step of time, which is called Sampling Importance Resampling (SIR). In this case, the proposal distribution is set to the transitional prior: $\pi(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{z}_k) = p(\mathbf{x}_k|\mathbf{x}_{k-1})$. This choice simplifies the importance weight update equation 7.4 to:

$$w_k = w_{k-1} L(\mathbf{x}_k|\mathbf{z}_k)$$

Both SIR and SIS are described on figure 7.1

7.3 Partitioned Sampling

The concept of particles is the approximation of the posterior distribution. The number of particles used in the particle filter defines the accuracy of this approximation. So the choice of this number is a compromise between performance and accuracy. Thus the relation between state space dimension and the number of particles required is obvious. The bigger the state dimension is the more you need particles to obtain good performances.
For a system which uses M different measurement sources, it is possible to partition the particle filter into several stages. This strategy is very useful to combine modalities, and decreases the number of particles needed to achieve a good level of performance. Perez and Black use this approach called **partitioned sampling** in a tracker fusing sound and visual measurements [26].

Partitioned sampling assumes the M measurements $\mathbf{z}_k = (\mathbf{z}_k^1, \mathbf{z}_k^2 ... \mathbf{z}_k^M)$ are independent.Therefore the likelihood can be calculated separately from each sources:

$$L(\mathbf{z}_k|\mathbf{x}_k) = \prod_{m=1}^{M} L(\mathbf{z}_k^m|\mathbf{x}_k)$$

It is still possible to use this property in a simple particle filter but it is also possible to exploit it better. To do that, the state is split in M partitions too and the proposal distribution is redefined to $\pi^m(\mathbf{x}_k^m|\mathbf{x}_k^{m-1}, \mathbf{z}_k^m)$ i.e. each proposal distribution drifts only the m partition of the state without modifying the others. Then the weights are updated according to the following equation:

$$\textbf{Update weights:} \quad w_k^m = w_k^{m-1} \frac{L(\mathbf{x}_k^m|\mathbf{z}_k^m) p^m(\mathbf{x}_k^m|\mathbf{x}_k^{m-1})}{\pi^m(\mathbf{x}_k^m|\mathbf{x}_k^{m-1}, \mathbf{z}_k^m)} \ \textit{with } m = 1..M \tag{7.5}$$

Note: Things are a bit different with the first partition: $\mathbf{x}_k^0 = \mathbf{x}_{k-1}^M$ and $w_k^0 = w_{k-1}^M$.

```
Take a set of $Np$ particles from previous set of time and proceed for time k.
**Initialization:**
For each particle: $\{\mathbf{x}_k^0, w_k^0\} = \{\mathbf{x}_{k-1}^M, w_{k-1}^M\}$
Then for each partition: $m = 1...M$
    **Proposal distribution:**
        $\mathbf{x}_k^m \sim \pi^m(\mathbf{x}_k^m|\mathbf{x}_k^{m-1}, \mathbf{z}_k^m)$
    **Weights update:**
        $w_k^m = w_k^{m-1} \frac{L(\mathbf{x}_k^m|\mathbf{z}_k^m)p^m(\mathbf{x}_k^m|\mathbf{x}_k^{m-1})}{\pi^m(\mathbf{x}_k^m|\mathbf{x}_k^{m-1}, \mathbf{z}_k^m)}$
        Afterwards normalize weights to have their sum equal to 1.
    **Optional resampling:**
        Replace $\{\mathbf{x}_k^m, w_k^m\}$ by $\{\mathbf{x}_k^m, \frac{1}{Np}\}$
        According to the distribution represented by weights.
**End of time step k**
```

Table 7.1: Generic partitioned sampling framework

This particular strategy splits the state into partitions associated to different measurements and runs one particle filter for each partition with an optional resampling between each stage. The order of the partition can have an important impact on the performance of the filter because the first stages guide the particles and the last stage defines the weights used for the estimation of the state. Therefore, to benefit from the advantages of partitioned sampling, the measurement modalities must be ordered from coarse to fine.

Chapter 8

Targets

This chapter discusses targets, which are the representations of people in the scene being monitored. First, the targets' main constituents are presented, then operations related to their management are introduced.

8.1 Target Definition

Each target is a collection of relevant information about a moving entity (ideally, a person) in the room. This information is principally aimed at determining a moving entity's position, orientation and size. Its most important parts are target-associated particles, its state estimate and reliability.

8.1.1 Miscellaneous Fields

To begin the description of targets, several miscellaneous pieces of information stored by targets need to be brought out:

- Target's ID - unique identifier of the target in our system;
- Target's age - number of consecutive frames the target has been in existence;
- Matched blob - the 3D foreground blob, the target is assumed to correspond to it in the current frame; see 8.2.

8.1.2 Target-Associated Particles

Since the approach chosen for target tracking is filtering by means of particles, each target the system generates has an associated particle set. The choice of its size is inherited from the previous system [1] and equals 50.

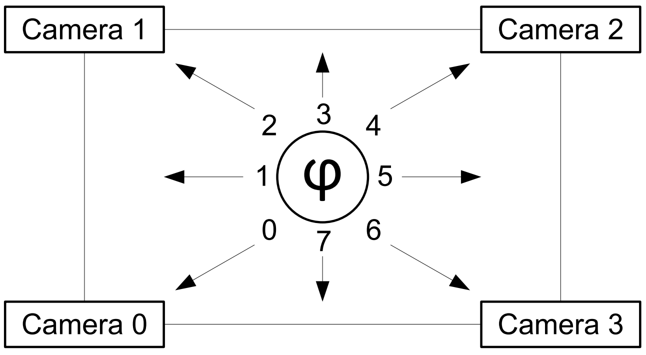


Figure 8.1: Representation of the values for the orientation φ. For example, if $\varphi = 3$, the target is viewed by cameras 1 and 2.

The state of every particle is defined so as to correspond to the objective of the system, i.e. being able to know the head centroid's position at every time instant. Therefore the state is:

$$x_i = (x, y, z, \varphi, \alpha). \tag{8.1}$$

Here the meaning of variables is as follows:

- (x, y, z) - 3D coordinates of the head in the world coordinate system;
- φ - the viewing direction of the person the target corresponds to, defined as a discrete value between 0 and 7. The camera-orientation associations are shown in Figure 8.1;
- α - the measure of the target's size on the 2D floor plan.

To conclude, at each specific instant, every target is associated with multiple hypotheses about its position, size and orientation.

8.1.3 Target State Estimate

Based on the states of target-associated particles, the state estimate for the target can be calculated. This is performed at each time step and the result is the core output of the system, i.e. the position of the person at a concrete time moment is said to be equal to this estimate.

Based on particular circumstances (refer to 8.4), our system uses either of two ways for obtaining this estimate: Maximum a posteriori (MAP) or Weighted average methods.

8.1.4 Target Reliability

The measure of a target's adequacy for being tracked and yielded as system's output is its reliability. It is defined as a numeric value between 0 and 1, higher values meaning higher adequacy.

Reliability was introduced in order to take account of the possibility of confusing real persons with noise when initiating new targets and existing targets becoming obsolete with time.

These issues are addressed by a reliability update mechanism and two reliability thresholds. The first threshold is the minimum reliability value, which is sufficient for a target to be kept alive. Newly-created target's reliability is initialized to this **elimination threshold** and is expected to increase for targets corresponding to real people in the scene, and to finally fall below the minimum for those representing anything else, e.g. non-human moving objects and noise. After the minimum has been transgressed, the target is discarded.

While the elimination threshold only provides a criterion for the continuation of a target's maintenance, the second threshold, called **reliable threshold**, when exceeded, makes a target not only kept alive but also reported as system's output.

Factors taken into account when updating target's reliability, are:

- existence of a match with a blob, detected in the current frame;
- the amount of evidence from both modalities, expressed as likelihoods;
- target's age;
- target's mobility;
- proximity to exit zones;
- sudden disappearance of foreground corresponding to the target.

The reliability computation mechanism, based on these factors, is explained in 8.5.2.

8.2 Target-Blob Matching

Having discussed the concept of the target, we can now turn to the target processing performed by the system.

Each time step starts with associating foreground blobs of the current time step with existing targets. First, the 2D distance is determined between every blob and every existing target. Then, the Smart Hungarian algorithm [25] produces associations based on the computed distance. The procedure might also yield blobs and targets without a match, so these output alternatives are possible:

- Matched blob-target pairs: each existing target is assigned a pointer to its matched blob, which is further utilized by the foreground detection part of the system.
- Unmatched blobs: new targets are initiated for large enough blobs.
- Unmatched targets: this will most likely (see 8.5.2) cause target's reliability to decrease.

8.3 Update

Our system combines two modalities for solving the tracking task, therefore we use the partitioned sampling approach to update the values of particles at every time instant. Namely, update is performed in two steps. At first, the foreground detection part drifts x, y and α components of the state vector, making use of target-blob matching sketched out in the previous section. Second, face detection part propagates z and φ components. Figure 8.2 describes the overall operation of partitioned sampling in our system.
Note that face detection will now be referred to as **FD**, and foreground as **FG**.

8.3.1 Foreground Detection Update

The particle update performed by the foreground detection part of our system can be outlined like this:

- Particles are drawn from FG proposal distribution, which combines a Gaussian random walk and a grid;
- Particle weights are updated according to FG likelihood;
- Weights are normalized;
- Resampling is performed.

Since this part was originally introduced by Martin and Rasmus Andersen and left unmodified in our system, the reader is referred to [1] for more details on it.

8.3.2 Intermediate Update

Before the face detection part starts updating z and φ components of particles, the necessary resampling of particles according to weights calculated using the FG likelihood is performed.

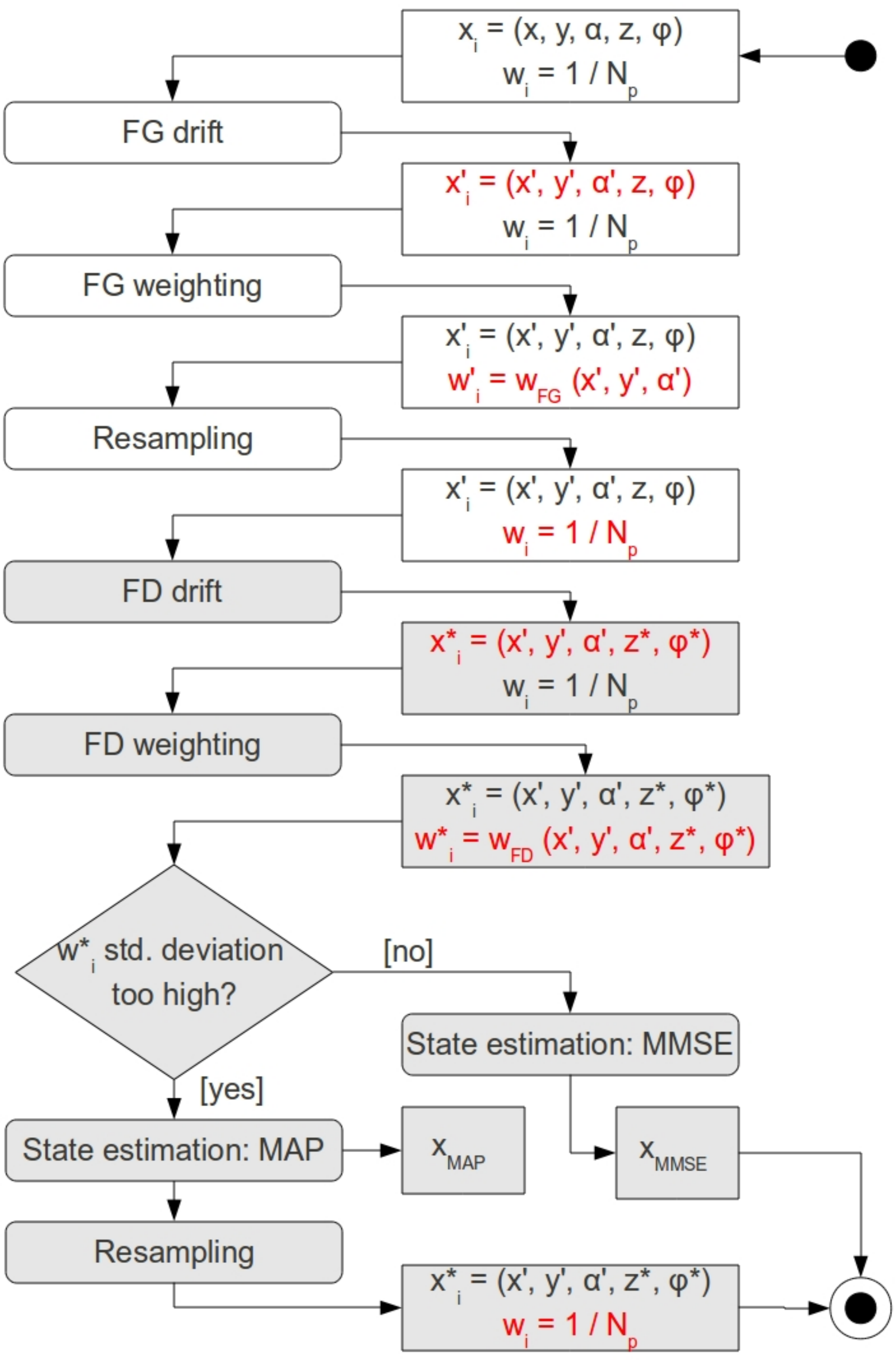


Figure 8.2: Partitioned sampling in our system

8.3.3 Face Detection Update

Similarly to the foreground detection update, face detection update can be described by a number of major steps:

- Particles are drawn from FD proposal distribution, which is chosen to be the same as the state evolution model. For each particle their z and φ values are updated as follows:
  - z values are drifted, with the displacement values being distributed according to a Gaussian distribution of zero mean and a predefined standard deviation. In certain circumstances z values are reinitialized before this drift is performed. Namely, they are reset to the target's maximum z value, i.e. its corresponding blob's top boundary, when there has been no face detection for this target for a predefined number of frames.
  - φ values are also changed by adding a random number from a Gaussian distribution with zero mean and a predefined standard deviation, taking modulus of 8 and making sure the obtained value is odd, so that the resulting orientation could ensure each particle being projected onto two camera frames.
- Bounding boxes for a target are calculated (see 10.2).
- Particle weights are updated according to FD likelihood. Since the proposal distribution chosen for z and φ update is equal to the state evolution model, the weight update formula is reduced to:

  $$w_k = w_{k-1} \times L(z_k|x_k).$$

  See more on FD likelihood in 10.2.
- Weights are normalized.
- State estimation is performed (explained in 8.4).
- Resampling is performed, if the standard deviation (defined as 8.2) of particle weights is too high.

8.4 State Estimation

As shown in 8.3.3, state estimation is performed after both parts of the system - foreground and face detection - have updated the particles of a target and before final resampling is done. Target state is estimated using either MAP, or weighted average.

8.4.1 Maximum a Posteriori Estimate

The first of the possible state estimation methods, used in our system, is the maximum a posteriori estimate. It can be described by the formula:

$$\hat{x}_{k|k}^{MAP} = \arg\max_{x_k} p(x_k|z_k).$$

(The notation $\hat{x}_{k_1|k_2}$ means the estimate x at time k_1 given measurements $z_{1:k_2}$.)

It is applied to the set of particles, so it actually has the effect of choosing the state of the particle with the highest weight as a state estimate.

8.4.2 Weighted Average Estimate (MMSE)

The second state estimation alternative is the minimum mean squared error (MMSE), defined as:

$$\hat{x}_{k|k}^{MMSE} = E(x_k|z_{1:k}) = \int x_k \cdot p(x_k|z_{1:k})dx_k.$$

With this estimator, the estimated state is the result of a weighted average. This approach combines (sums up) all particle states into a single value by using their weights as importance coefficients.

8.4.3 State Estimation in our System

In our system the decision on which state estimation method to use is based on the particle weight standard deviation value, given by the formula:

$$s = \sqrt{\frac{1}{N_p}\sum_{k=1}^{n}\left(\frac{1}{N_p} - w_k\right)^2} \tag{8.2}$$

MAP is used when s $> 0.1 \times \frac{1}{N_p}$, otherwise the weighted average is used, meaning that when particle weights differ too much, it is better to choose the state of the particle with the highest weight and when the weights are quite equal, weighted average would yield a better solution.

However, the original system of Martin and Rasmus Andersen [1] used only the weighted average approach to estimate state vector component values.

8.5 Management

Target management includes administrative tasks, related to the proper maintenance of targets, which have to be executed for each frame. The tasks include target initialization, reliability determination and target merging and are discussed in this section.

8.5.1 Initialization

The process of matching existing targets to foreground blobs discovered in the current time step might ouput unmatched foreground blobs, as was said in 8.2. These blobs become the reason for initiating new targets, because they bear witness to the existence of new persons in the room.

The state components x, y and α of a new target are initiated based on the corresponding blob parameters. z is chosen to reflect the average human height - 170 cm - and is constant for all new targets. Finally, φ is set to a random value from a uniform distribution between 0 and 7, expecting that the particle filter will make its value converge to the true one anyway.

Together with the target, its particles are initiated. They all have the same x, y, z and α state components equal, namely, set to those of the initial state of the target. However, φ values are drawn from a uniform distribution.

Target reliability initialization is covered in the next section.

8.5.2 Determining Reliability

After the creation of new targets, their reliability should be initialized, and that of old targets should be updated. In this section, the current time step target reliability computation mechanism is presented.

Goals, taken into account, when determining reliability, are [1]:

- Targets, matching **real people**, should become reliable as soon as possible;
- **Mistaken** targets, e.g. initialized on moved objects or noise, should not become reliable sooner than they fade into the background;
- **Stationary**, but old and reliable targets should retain their reliability for a long time, so that they could stay alive despite the lack of supporting evidence from the measurements.

Keeping these goals in mind, the reliability calculation was designed. The formula for updating the reliability is:

$$r_n = (1-k)r_{n-1} + k \cdot P_n \tag{8.3}$$

where r_n is the reliability of a target at frame n, with r_0 set to the reliability value $r_{eliminate}$ to just prevent the target from being eliminated;
P_n is 1, if the target is present, and 0 otherwise;
k is the reliability learning factor.

In order to implement it, every time step these key computations are performed:

- Reliability of new targets is initialized to the elimination threshold.

- It is determined if the target is **detected** in the current time step. It is considered detected, if there exists a spatially matching blob or either of likelihoods is above a threshold.
- Target mobility is evaluated.
- Reliability learning rate is set according to the formula:

$$k = \begin{cases} k_{min} & \text{for an immobile target} \\ min(\frac{1}{n} + k_{min}, k_{max}) & \text{for a mobile target} \end{cases}$$

  This formula reflects the third goal by minimizing the reliability change of immobile targets. Also, it means that the learning rate decreases from maximum to minimum value throughout target's existence, the decrease being larger (and therefore reliability more stable) for older targets. Parameters k_{min} and k_{max} are tuned in correspondence to background learning rate in order to achieve the second goal.
- As an addition to the previous system, face detection module makes a contribution achieving the first goal, stated above. Namely, if the FD likelihood is above a threshold, the learning rate is set to the maximum value, meaning the reliability will increase fast for real humans. Also, if FD likelihood is high, reliability is set to the reliable threshold, so that targets, corresponding to people, appear as system's output immediately.
- Reliability is calculated according to 8.3.

The resulting reliability is used to take a decision about whether to keep or kill a target. Namely, a target is removed if its reliability is below the elimination threshold, no likelihoods are available from any of the modalities, the target is in one of the exit zones or its corresponding foreground has suddenly disappeared.

8.5.3 Target Merging

Target merging is the final step in target processing, performed at each time step. It is necessary because it is possible that during the course of system execution different targets might end up tracking the same person.

Merging is performed by first checking all currently existing target pairs for possible merging. The criterion is the distance between two targets: if it becomes too small, the target pair is marked as requesting merging in the current frame. Then, all pairs having requested merging for a specific number of consecutive frames are merged.

Chapter 9

Foreground Likelihood

As we use particle filters to track the target, we need some likelihood functions $L(x|z)$ which measures the data z with a given state x. We decided to keep the foreground part from the previous project, so this chapter will give a short summary of the foreground likelihood function used by Martin and Rasmus Andersen in [1].

9.1 State Estimation

The likelihood function is given as $L(x_k|z)$. It measures how likely a state x_k is, given measurement data z.

When using foreground projected to the floor, the minimal possible number of states to estimate is the two coordinates (x, y). Besides, a size α is added, in order to determine the space occupied by the target on the scene. With these parameters, the shape of a state can either be a square or a circle. However the circle does not fit properly with the discrete representation of the foreground. Therefore, the size is represented as a square. Figure 9.1 illustrates this choice.

9.2 Likelihood Function

For designing the foreground likelihood function, the position (x,y) can be excluded as function parameters for a question of simplicity, thus there are three values which can be taken into consideration :

1. Volume : the number of voxels N(α) represents the volume of a person
2. Density : a person might not have a bigger volume than the volume V , inside his 3D bounding box :
$$F(\alpha) = \frac{N(\alpha)}{V(h, \alpha)} = \frac{N(\alpha)}{h.(2\alpha + 1)^2} \tag{9.1}$$
The height h corresponds to the maximum number of voxels in a one column in the area.

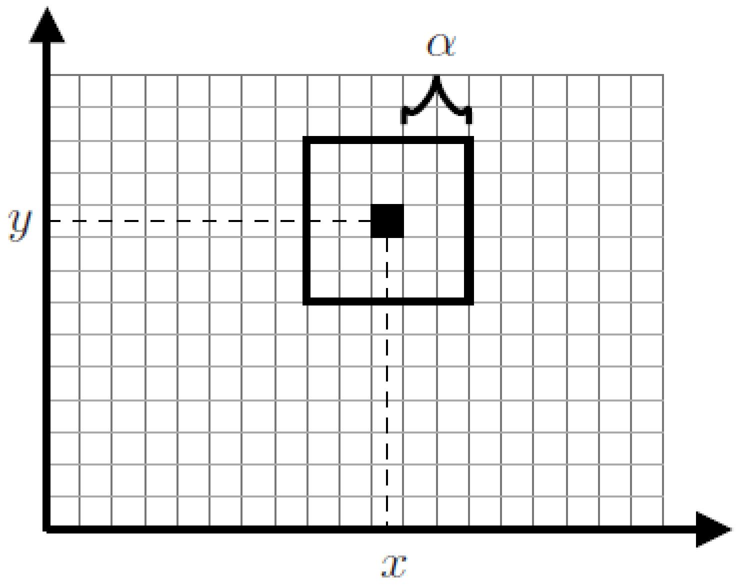


Figure 9.1: The foreground projected to the floor at the positions (x, y) and the size α used in the state [1].

3. Derivative of the density : to have a good state, we expect to be close to the center of a person and include most of her, that is to say F(α) might drop fast if α increases. It can be explained by the fact that the area around a person is without foreground. The derivative $\frac{\partial F(\alpha)}{\partial \alpha}$ can measure the change in F(α) :

$$F_d(\alpha) = \frac{\partial F(\alpha)}{\partial \alpha} = \frac{F(\alpha + k) - F(\alpha - k)}{2k} \tag{9.2}$$

$$F_d(\alpha) \approx \frac{1}{2kh}\left(\frac{N(\alpha + k)}{(2(\alpha + k) + 1)^2} - \frac{N(\alpha - k)}{(2(\alpha - k) + 1)^2}\right) \tag{9.3}$$

where h corresponds to the maximum height of the smaller area.

The main problem is that the likelihood function has to be able to separate two people standing close to each other. Besides, as the likelihood function needs to be evaluated once for each particle of each target, it is important to take into consideration the computation time. So, the final likelihood function will be :

$$L(\alpha) = -F(\alpha - k)^2 . \sqrt{N(\alpha + k)} . F_d(\alpha) \tag{9.4}$$

The function is weighted by squaring F(α - k) and the square root of N(α + k) is realized to bias toward single coherent people.

Chapter 10

Face Likelihood

The second particle weight update of the partitioned sampling is determined by the face likelihood. The challenge here is to find a way to extract and use the relevant information from the Viola-Jones face detector described in Chapter 6.
From the state of a particle (see Equation 8.1), only the size α is not used during this step. The 3D position of the head (x, y, z) and its orientation φ will be used in a function that measures how likely they are to be a face.

In order to determine this function, some preliminary steps are necessary. Starting from a particle's state, a process is defined that leads to the actual use of the function. We tested two different approaches, each one having its own process and final likelihood function. We will describe both in the following sections.

10.1 First Approach

Process

The first approach is represented in Figure 10.1. The following steps are applied for each particle.

1. The orientation φ gives the number(s) of the corresponding camera frame(s). There can be one or two cameras. We decide to represent the orientation φ with 8 values, from 0 to 7, this is described in 8.1.2.

2. A head area centered on the (x, y, z) position is defined in 3D. The size of this area is approximately 3 times that of a head. Its 8 corner points are then projected to the selected frame using the intrinsic and extrinsic parameters of the camera. This projection is an image segment in which we look for bounding boxes from the Viola-Jones detector. The selected bounding boxes (i.e. those located inside the segment) are then validated according to their width, which is expected to be roughly one third of the segment width.

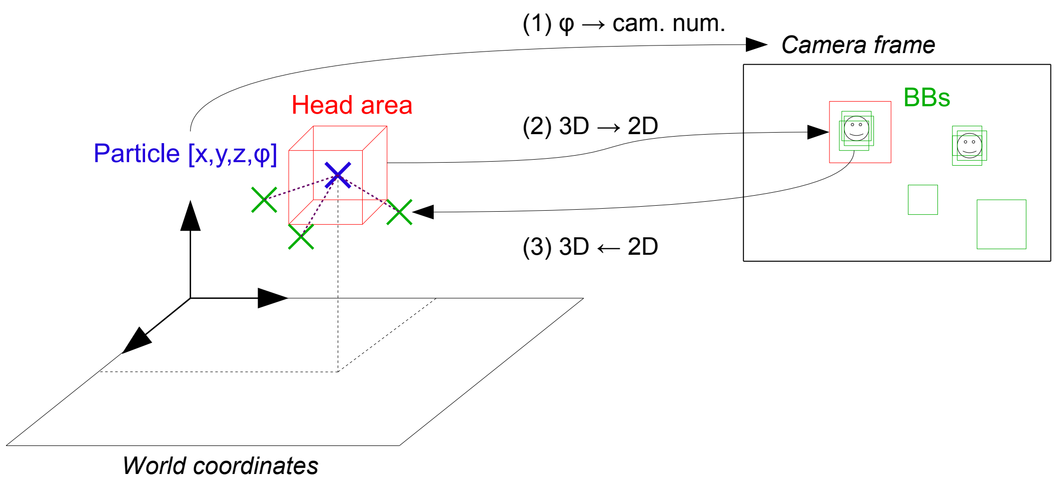


Figure 10.1: First face likelihood process: (1) The orientation φ gives the corresponding camera frame (2) A head area is defined in 3D and then projected in 2D (3) The selected bounding boxes are projected back to 3D

3. The validated bounding boxes are projected back to 3D by assuming that their real width is 15 cm [13]. Only the center of the boxes are projected, resulting into 3D points.

The contribution from a bounding box should be inverse proportional to its distance to the particle. We therefore define the likelihood as the sum of all of these contributions:

$$Likelihood = \sum_{i=1}^{N} \frac{1}{(||X_p - X_{BBi}||)^{K/2} + 1} \tag{10.1}$$

Results

The first problem we encountered was that our system was very slow. This is a consequence of running the Viola-Jones algorithm every time on the whole image for each frame of each camera. Secondly, mapping an image point into a real world point was not accurate enough. Indeed, this task presents several difficulties.
The forward problem of converting from 3D to 2D is a non-linear one, which means that its inversion is impossible and thus, approximations have to be made. The approximations don't give just one single point on the real-world coordinate system because of the uncertainty on depth. The latter is almost never resolved using just one camera and this is a problem because faces are often detected just by one camera. For the likelihood function, we calculated the distance from the particle under consideration, to the center of the BBs which have been projected to 3D. But here again one of our problems, because the 2D to 3D conversion is not precise, is that the resulting distance measurement is incorrect. We used a validation method based on an assumed standard face width

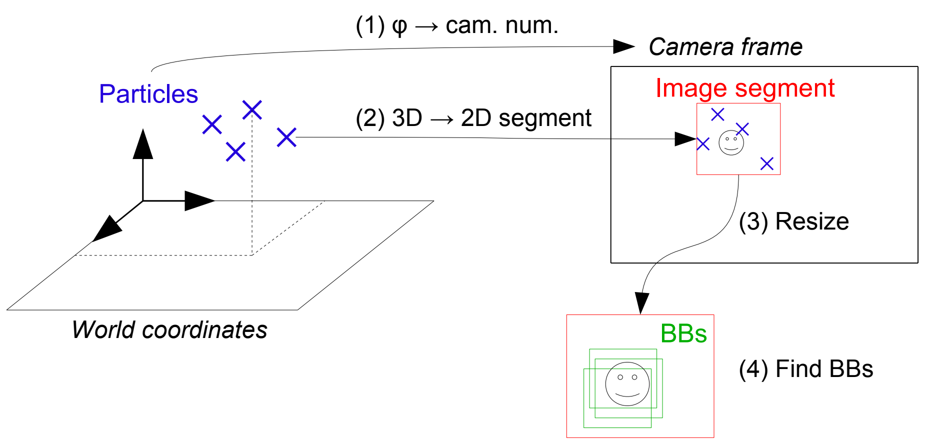


Figure 10.2: Second face likelihood process: (1) The orientation φ gives the corresponding camera frame for each particle (2) All particles are projected and define an image segment (3) The image segment is resized to a constant size (4) Viola-Jones algorithm is applied

of 15 cm, which means that the detected faces which corresponded to profile faces for example yielded terrible 3D position estimations.

To deal with these problems, we decided to take another approach. To reduce the execution time, we will only apply the Viola-Jones algorithm on a region of interest that we will determine beforehand. Then, we will stop doing the 2D to 3D conversion and we will use just the 2D coordinates for the likelihood function. Finally, we decided to implement another validation method.

10.2 Second Approach

Process

Given the problems encountered with the first approach, a second one was implemented. The new process is represented in Figure 10.2.

1. The first step is similar to the previous process, each particle is assigned one or two camera frame(s) according to its orientation φ.

2. All particles are projected to their camera frame(s). An image segment is defined on each frame by looking for the top/left/bottom/right most particles. This area is also expanded by a pre-defined scale (e.g. 1.4) to make sure no faces are missed during detection.

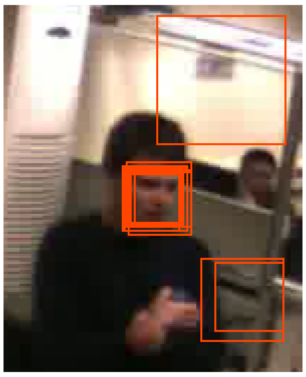


Figure 10.3: Example of false positives on an image segment. It is clear that the actual face area has a stronger response, which illustrates the use of grouping bounding boxes.

3. The image segment is resized to a constant width of for example 150 pixels. This is done in order to ensure that the number of bounding boxes returned does not depend on the proximity to the camera. Both the width and height of the segment are multiplied by a scale factor β. This factor is then defined as $\beta = \frac{150}{segment_{width}}$.
4. The Viola-Jones algorithm is applied on this resized segment. The resulting bounding boxes are turned back to their original position and scale on the camera frame by simply dividing them by β.

Grouping bounding boxes

After this process, the bounding boxes are grouped according to their common area ratio, defined as:

$$CAR = \frac{A[BB_1 \cap BB_2]}{min(A[BB_1], A[BB_2])} \tag{10.2}$$

Where $A[BB]$ is the area of a certain bounding box.

Two bounding boxes are assigned to the same group whenever their CAR is above a certain threshold, e.g. 40%. The multiplicity of a group is defined as the number of bounding boxes it contains. Any group whose multiplicity is lower than a defined threshold is then discarded. This helps combat false positives which have a lower response in terms of number of bounding boxes than actual faces. This phenomenon is shown in Figure 10.3.

Likelihood

The likelihood of a particle is now related to its 2D distance to each bounding boxes. This distance is also multiplied by the multiplicity of the group to which the bounding box belongs. In case a particle is projected on two camera frames, the likelihoods are summed.

$$Likelihood = \sum_{i=1}^{N} multiplicity \times exp\left(\frac{-||X_p - X_{BBi}||^2}{BB_w.BB_h.2.\sigma^2}\right) \tag{10.3}$$

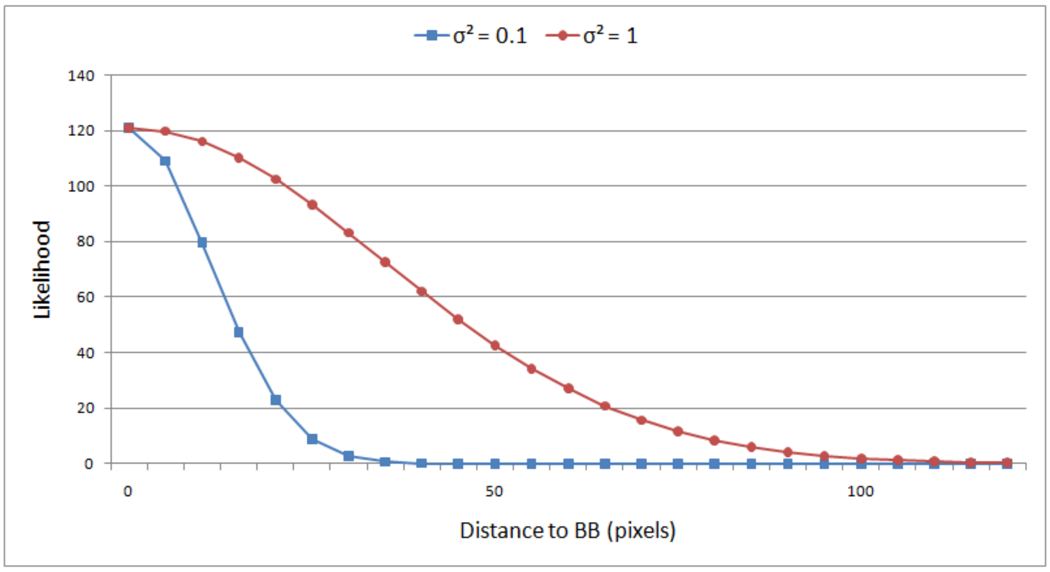


Figure 10.4: Comparison of two σ^2 values. Using $\sigma^2 = 0.1$ yields a more strict selection according to the distance. A bigger value such as 1 still allows likelihoods to be clearly distinguished, but is not selective enough for our tracker.

The σ^2 parameter defines how fast the likelihood decreases according to the distance. This is shown in Figure 10.4. We use a low value of 0.1 for σ^2 in order to be strict enough so that not every particle have good likelihoods. This way, every near-perfect match is also highly stronger in likelihood than others.

Moreover, the 2D distance is normalised by the area of the face estimate in order to not penalise the likelihoods of large faces against those of smaller ones. This is done because it's the distance in meters and not in pixels that dictates how good the match is. Since the distance in meters is a function of the distance from the camera, we normalise it as a function to the face size.

Results

All of these steps to calculate the likelihood form a good and sufficient face validation process, which eradicates nearly all false positives. All the parameters used in the implementation are shown in Table 10.1.

Parameter	**Symbol**	**Value**
Face validation parameters		
Expansion scale of image segment	-	1.4
Rescaled segment width	-	150 pixels
Minimum CAR for belonging to same group	-	40%
Minimum number of BBs for a group to be valid	-	8
Variance of the exponential distribution	σ^2	0.1
Viola-Jones parameters		
Scale factor	-	1.05
Minimum search size	-	14 pixels

Table 10.1: Parameters for then final face likelihood process

Part IV

Implementation

Contents

In the previous part, the algorithms and techniques of tracking persons using foreground and face detection were explained. In this part, the implementation of our algorithms in C++ is designed. It also gives information about the existing source code from the previous project.

Chapter 11

Software Design

This section documents the C++ design of the previous system [1] and explains how we improve it. There are two different versions of the source code. The first one needs five computers to run and the second one needs only one. In our project we are going to use the second version. The basic structure and the most important classes of the source code are described below.

11.1 General Description of the Existing Source Code

There are two different versions of the tracking algorithms: the first one is executed by five computers and runs in parallel. The second version of the system combines information from all cameras and treats them sequentially. The tracking algorithms' results are stored and display by *Renderer*. It runs in parallel and executes a separate thread while *Application* is executed in the main thread. The framework gives intermediate and final results of the tracking algorithms.

Application: This class calls and synchronizes the algorithms in PerCam and Sequential. When a class is created, it runs in the main thread which first calls PerCam. It is followed by running of the tracking algorithms. It ends by requesting DataManager to swap the ProtectedData buffers.

PerCam: This class contains all the algorithms which are used for each camera. All results are stored in the back buffer. 97% of the work for the program are spent on the algorithms perCam. Indeed, the thread in PerCamRunner wait for signal to start executing algorithms and signal to Application when algorithms are finished.

Sequential: This class contains all the algorithms which cannot be used in parallel. The results of these algorithms are stored in the back buffer.

Renderer: This class puts the most interesting things on the display. They include the videos, the 3D view, the reasoning results, and any debugging displays that have been enabled.

Configuration: The class Configuration keeps in memory all the parameters that the user can change easily. When the program starts, one or more Configuration files can be read. The program can run without this file by using default values. Three main parameters can be changed in the Configuration file:

- *Video source*: A path to a web-cam, a video file or to a folder of images.
- *Algorithms parameters*: Parameters of the tracking algorithms.
- *Intermediate display*: We can choose which result we want to display.

onKey: onKey function receives two keyboard commands: The first one can pause the video and the second one can rotate the 3D view of the room.

The program needs two other classes to work: *DataManager* and *ProtectedData*. These classes give Renderer access to the intermediate results from the tracking algorithms. ProtectedData also retains all the shared data.

11.2 Implementation of Face Tracking

We implemented the algorithm for the face detection part in the *Track* file because we have to manipulate the *Particle* and *Target* objects. Figure 11.1 shows how we update a target using face detection.

The description of the algorithm is done in Chapter 8. Functions have been added to the *Particle* and *Target* classes and we created a new class which is *BBGroup* in order to calculate, manipulate and save the BB groups (see the Equation 10.2). Besides, we save all the data resulting from the execution of the Viola-Jones algorithm, the calculation of the bounding boxes groups and the image segments, in the *ProtectedData* file so as to make them accessible to the different threads, more particularly the *Renderer* one, and also to use them in the calculations we are doing. On top of that, we added and set some parameters to the Configuration file used for the face detection modalities. The *Configuration* class has been updated by taking in account these new parameters and thus enables their access in any function.

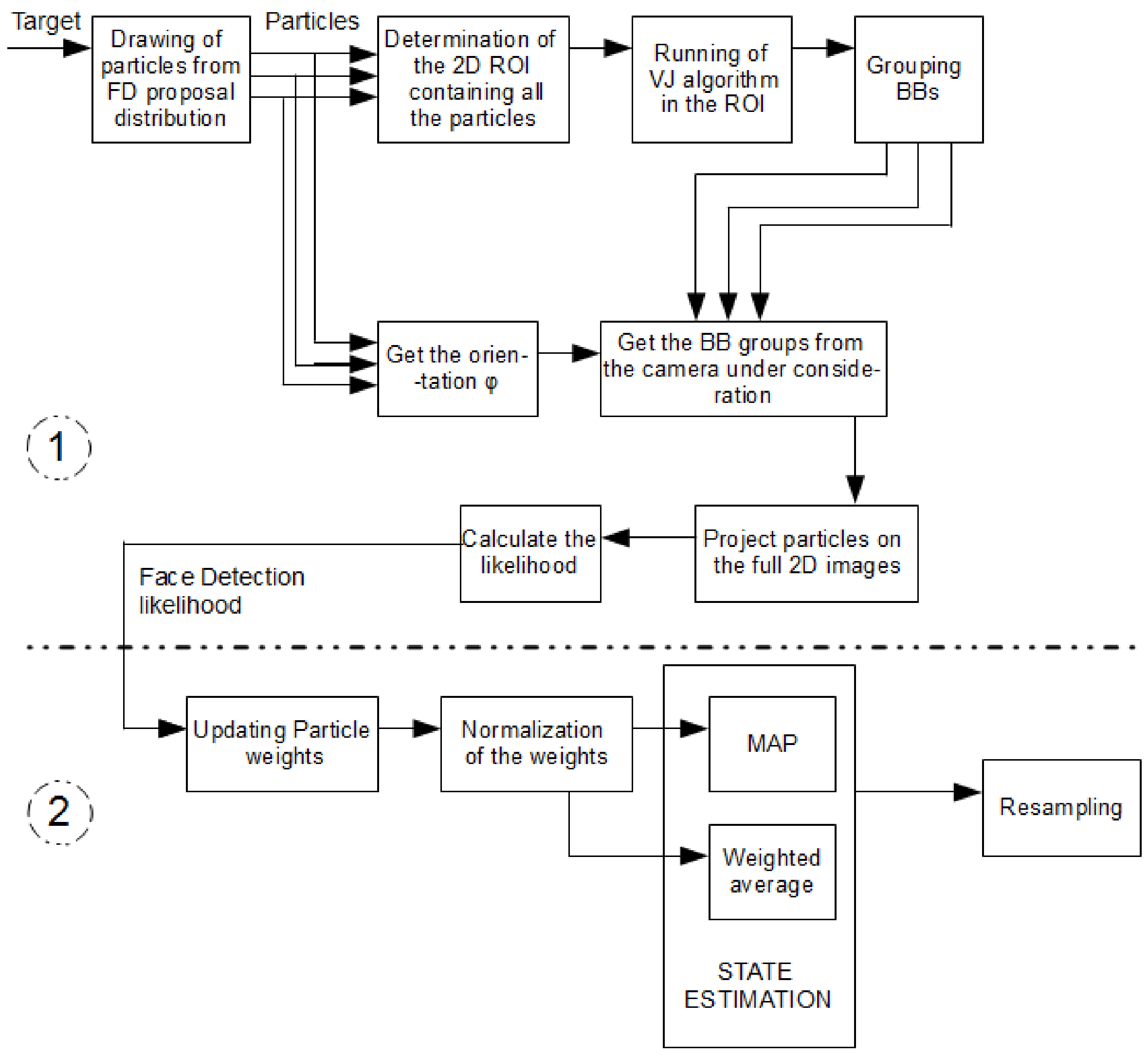

Target
Drawing of particles from FD proposal distribution
Particles
Determination of the 2D ROI containing all the particles
Running of VJ algorithm in the ROI
Grouping BBs
Get the orien- -tation φ
Get the BB groups from the camera under conside- ration
1
Calculate the likelihood
Project particles on the full 2D images
Face Detection likelihood
Updating Particle weights
Normalization of the weights
MAP
Weighted average
STATE ESTIMATION
Resampling
2

Figure 11.1: Cycle for updating a target using Face Detection. The part 1 and 2 are detailed respectively in Chapter 10 and Chapter 8.

Part V

Evaluation

Contents

This part will present the different results obtained while testing the system. It also describes the main tool used for evaluating the tracker: the renderer. To finish, we will make a conclusion about what we have done for this project and notably we will sum up the choices we have made and how we built the system to achieve our goal.

Chapter 12

Renderer

In this chapter, we present a useful tool we used to evaluate our tracking system: a renderer inherited from the previous project [1]. This renderer generates 2 types of displays:

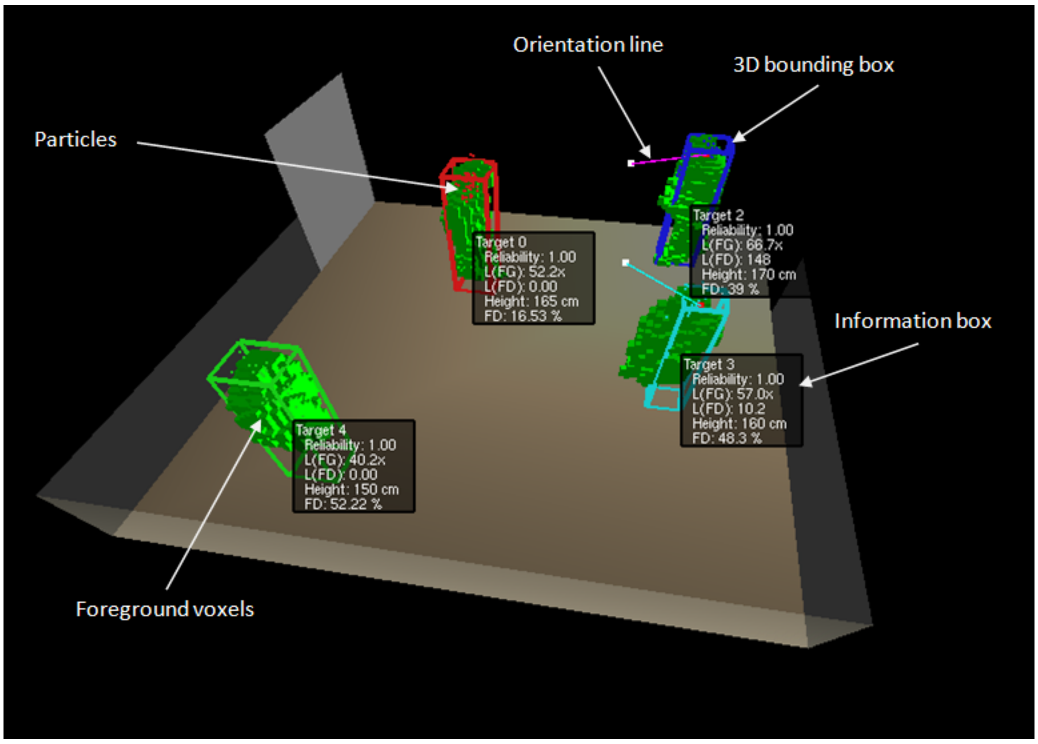
- A 3D rendering of the room showing computation results such as 3D foreground and particles.
- Camera frames with additional informations such as face detections and image segments (cf. 10.2).

The renderer saves these data in videos. Some videos are available in the CD enclosed to this report. They are based on videos we used to develop our system.

12.1 3D Renderer

This renderer generates a 3D view of the room using OpenGL. Different relevant informations are shown. Figure 12.1 illustrates the possibilities of the 3D renderer.

- **Foreground voxels:** They are displayed as green or grey cubes. Green voxels show the foreground is big enough to be considered as a target.
- **Particles:** They are represented as color points. The color is associated to a target. We use (x, y, z) of each particle state as the 3D position.
- **Orientation lines:** They are displayed only when faces are detected. It starts from the target estimated state (red point) and it shows the direction in which the target is most likely looking. There are two different colours: magenta means the target face is detected by 2 cameras and cyan means the face is only seen by 1 camera.
- **3D bounding boxes:** 3D cube that surrounds the reliable targets. The color is associated to a target.



```
Target 0
  Reliability: 1.00
  L(FG): 52.2x
  L(FD): 0.00
  Height: 165 cm
  FD: 16.53 %

Target 2
  Reliability: 1.00
  L(FG): 66.7x
  L(FD): 148
  Height: 170 cm
  FD: 39 %

Target 3
  Reliability: 1.00
  L(FG): 57.0x
  L(FD): 10.2
  Height: 160 cm
  FD: 48.3 %

Target 4
  Reliability: 1.00
  L(FG): 40.2x
  L(FD): 0.00
  Height: 150 cm
  FD: 52.22 %
```

Figure 12.1: Detailed screen capture of the 3D renderer

- **Information boxes:** These boxes provide relevant informations for each step of time such as values of likelihood, reliability, percentage of face detection throughout time and estimated height of the target head center.

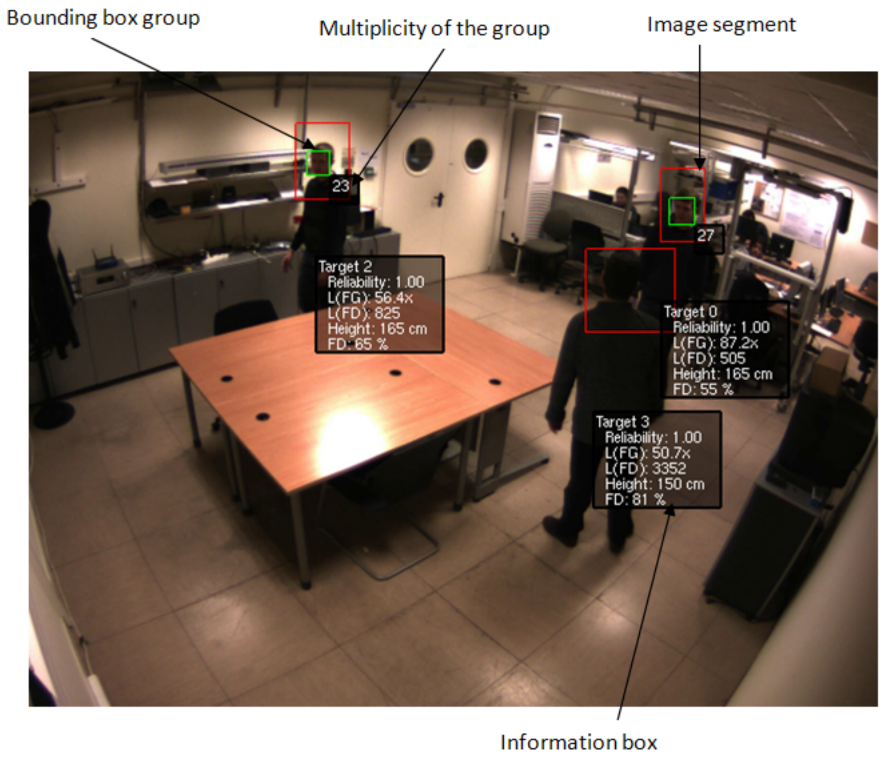
12.2 Frame Renderer

This renderer shows the cameras' frames with some informations we use to adjust parameters in the computation of the face likelihood. Figure 12.2 illustrates the features available in the frame renderer.

- **Image segments:** For every target, we estimate an area where the face might be found on each camera frame. These areas are represented by orange rectangles.
- **Bounding boxes:** Every time we detect valid faces, bounding boxes are grouped as defined in 10.2, and are then displayed as green rectangles with their multiplicity.

12.3 Interpretation

With the two renderers described previously, we are able to evaluate some features of our system. We can see when a target is initialized and how fast the system makes it reliable or discards it. We also see if the system misses a target. After the initialization,



```
Target 2
  Reliability: 1.00
  L(FG): 56.4x
  L(FD): 825
  Height: 165 cm
  FD: 65 %
```

```
Target 0
  Reliability: 1.00
  L(FG): 87.2x
  L(FD): 505
  Height: 165 cm
  FD: 55 %
```

```
Target 3
  Reliability: 1.00
  L(FG): 50.7x
  L(FD): 3352
  Height: 150 cm
  FD: 81 %
```

Figure 12.2: Detailed screen capture of the camera frame renderer

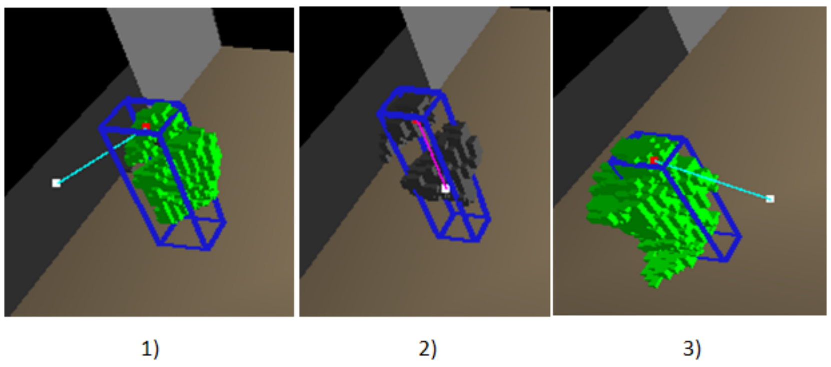


Figure 12.3: Illustration of how face detection can keep the track when foreground is failing. 1) Blob rendered with good foreground detection. 2) Tracking maintained by face detection for a while. 3) Rematch target with blob when foreground recovers.

we easily see if the track works and if the target is deleted at the right time. When we encounter problems with tracking or initialization, the renderer helps us a lot to find the issue, which is convenient. For example we can determine if false positives in face detection make the system lose the track or if a moving object like a door becomes a target. Our system is not meant to work in real time. With the computer we use for testing we can process between 2 and 3 frames each second. The speed depends on the number of targets because we use one instance of the particle filter for every target which corresponds to 50 particles each. We can also use the renderer to test our system on specific scenarios which demonstrates the relevance of face detection in a tracking system.

- **Scenario 1:** The target is correctly initialized and the track works from the beginning. Then the foreground detection fails to build a blob because the target does not move or because it cannot be seen by enough cameras. At this moment the face detection allows the system to keep the target reliable until the foreground detection recovers. Figure 12.3 gives an example of this scenario.

- **Scenario 2:** The target is a tracked person. This person seats and the target starts fading into the background. The person is facing a camera so there is a face detection which is used to keep the track. The challenge here is to find the face when the height of the head changes quickly.

- **Scenario 3:** Some targets are tracked by the system and there is a sudden lighting variation in the room. The system will fail to obtain relevant foreground data, and the whole room will be turned into foreground. The face detection module is not sensitive to lighting variation, so the system should track the target with faces only while the background model is updated.

We don't have the specific videos to test scenario 2 and 3 but we believe our system should be able to deal with these situations.

Chapter 13

Conclusion

We have presented in this report an efficient way to track persons in a room. A new algorithm was presented based on foreground and face detection. This kind of system is very relevant when you want to help elder people in the everyday life. To reach this goal, the installation and calibration of five cameras in a room is required. This system can reason about mobility and three different body posture: standing, sitting or fallen. The system was implemented upon an existing one [1] with the intention of being more reliable and powerful.

The framework of the tracking system is based on particle filter, also known as the Condensation algorithm [2]. More specifically, since two modalities are to be combined, the use of partitioned sampling was decided upon. This allows for a greater state dimension without the need for additional particles.
The two modalities considered are foreground detection as used in the previous system, and face detection based on the Viola-Jones algorithm [33]. This widely used method was chosen for its ability to use a specifically trained cascade, which we have at our disposal. It is already available in various coding libraries, such as OpenCV, which makes it easy to implement.
The partitioned particle filter can be seen as a cascade of two simple particle filters, each one being associated to a single modality. The modalities are ordered from coarse to fine, which corresponds to using foreground detection first to update the floor position of a target. The second modality - face detection - then evaluates how likely a target is to be human.

This project shows face detection is a good modality for human tracking even if faces are not always detected depending on persons' behaviour. The Viola Jones face detector has good performances and we have proposed a way to turn its results into likelihoods. Therefore it is possible and efficient to use it in a particle filter.
Finally our system is able to track people in a room. In some situations, mistakes occur but most of the time it works fine. Also we only used one set of videos for developing and testing. Our tracking application is now ready to be tested on more various situations.

Bibliography

[1] M. Andersen and R. Andersen. Multi-camera person tracking using particle filters based on foreground estimation and feature points. Master's thesis, AAlborg University, 2010.

[2] M. J. Black and A. D. Jepson. Recognizing temporal trajectories using the condensation algorithm. *International Journal of Computer Vision*, April 1998.

[3] T. Bouwmans, F. E. Baf, and B. Vachon. Background modeling using mixture of gaussians for foreground detection - a survey. *Recent Patents on Computer Science*, 2008.

[4] C. Canton-Ferrer, J. R. Casas, and M. Pardas. Towards a bayesian approach to robust finding correspondences in multiple view geometry environments. In *In International Conference on Computational Science (2) pages 281âĂŞ289*, 2005.

[5] C. Canton-Ferrer, J. Salvador, J. Casas, , and M.Pardas. Multi-person tracking strategies based on voxel analysis. *Lecture Notes in Computer Science*, 2009.

[6] CLEAR. Classification of events, activities and relationships. `http://www.clear-evaluation.org/`.

[7] S. Cooray and N. O'Connor. Facial features and appearance-based classification for face detection in color images. In *International Workshop on Systems, Signals and Image Processing*, 2004.

[8] X. H. Fang, W. Xiong, B. J. Hu, and L. T. Wang. A moving object detection algorithm based on color information. In *J. Phys.: Conf. Ser.*, 2006.

[9] N. Friedman and Russell. Image segmentation in video sequences: A probabilistic approach. In *Thirteenth Conf. on Uncertainty in Artificial Intelligence*, 1997.

[10] T. Horprasert, D. Harwood, and L. S. Davis. A statistical approach for real-time robust background subtraction and shadow detection. In *ICCV Frame-Rate WS*, 1999.

[11] P. Kaewtrakulpong and R. Bowden. An improved adaptive background mixture model for realtime tracking with shadow detection. *Proc. 2nd European Workshop on Advanced Video Based Surveillance Systems*, 2001.

[12] R. E. Kalman. A new approach to linear filtering and prediction problems. *ASME Journal of Basic Engineering*, 1960.

[13] N. Katsarakis and A. Pnevmatikakis. Face validation using 3d information from single calibrated camera. In *The 16th International Conference on Digital Signal Processing*, 2009.

[14] N. Katsarakis, F. Talantzis, A. Pnevmatikakis, and L. Polymenakos. The ait 3d audio visual person tracker for clear 2007. *Lecture Notes in Computer Science*, 2009.

[15] H. Kim, R. Sakamoto, I. Kitahara, T. Toriyama, and K. Kogure. Robust silhouette extraction technique using background subtraction. In *10th Meeting on Image Recognition and Understand (MIRU 2007)*, July 2007.

[16] C. Kotropoulos and I. Pitas. Rule-based face detection in frontal views. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1997.

[17] J. L. Landabaso and M. Pardas. Foreground regions extraction and characterization towards real-time object tracking. In *Proceedings of Multimodal Interaction and Related Machine Learning Algorithms (MLMI)*, 2005.

[18] J. L. Landabaso, M. Pardas, and L.-Q. Xu. Hierarchical representation of scenes using activity information. In *In Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, March 2005.

[19] A. Lanitis, C. J. Taylor, and T. F. Cootes. Automatic interpretation and coding of face images using flexible models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 19, No. 7, july 1997.

[20] A. Leone, G. Diraco, C. Distante, P. Siciliano, M. Malfatti, L. Gonzo, M. Grassi, A. Lombardi, G. Rescio, P. Malcovati, V. Libal, J. Huang, and G.Potamianos. A multi-sensor approach for people fall detection in home environment. *Workshop on Multi-camera and Multi-modal Sensor Fusion Algorithms and Applications*, 2008.

[21] L. Li, W. Huang, I. Y.-H. Gu, and Q. Tian. Statistical modeling of complex backgrounds for foreground object detection. *IEEE Transactions on Image Processing*, November 2004.

[22] C. Liu. A bayesian discriminating features method for face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 20, No. 1, 1998.

[23] S. Maskell and N. Gordon. A tutorial on particle filters for on-line nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 2001.

[24] B. Menser and F. Muller. Face detection in color images using principal component analysis. In *IEE Conference Publication*, 1999.

[25] J. Munkres. Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*, 1957.

[26] J. V. Patrick Pérez and A. Blake. Data fusion for visual tracking with particles. In *Proceedings of the IEEE*, February 2004.

[27] H. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 20, No. 1, 1998.

[28] K. Schwerdt and J. L. Crowley. Robust face tracking using color. *IEEE International Conference on Automatic Face and Gesture Recognition*, 2000.

[29] P. Smyth. Face detection using the viola-jones method. Technical report, Dublin City University, 2007.

[30] M. Stamp. A revealing introduction to hidden markov models, 2004.

[31] C. Stauffer and W. E. L. Grimson. Learning patterns of activity using real-time tracking. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2000.

[32] E. M. Verma, E. P. Rani, and E. H. Kundra. A hybrid approach to human face detection. *International Journal of Computer Applications*, VOL. 1, N0. 13, 2010.

[33] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *IEEE Conference on Computer Vision and Pattern Recognition*, number 511, Dec 2001.

[34] J. Wang and T. Tan. A new face detection method based on shape information. *Pattern Recognition Letters 21*, november 1999.

[35] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: Real-time tracking of the human body. *IEEE*, 2002.

[36] G. Yang and T. S. Huang. Human face detection in a complex background. *Pattern Recognition*, VOL. 27, N0. 1, 1994.

[37] J. Yang and A. Waibel. A real-time face tracker. *IEEE Workshop*, 1996.

[38] M.-H. Yang, D. J. Kriegman, and N. Ahuja. Detecting faces in images: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 24, No. 1, january 2002.

[39] C. Zhang and Z. Zhang. A survey of recent advances in face detection. Technical report, Microsoft Research, june 2010.

[40] Q. Zhang and E. Izquierdo. Multi-feature based face detection. In *IET International Conference on Visual Information Engineering*, 2006.