

Crowd Surveillance by Video Camera

Master's Thesis
Coen van der Veen

Delft University of Technology
Faculty of Electrical Engineering, Mathematics and Computer Science
June 2006



Graduation committee:

Drs. dr. L.J.M. Rothkrantz
Dr. ir. C.A.P.G. van der Mast
Ir. F. Ververs

Veen, Coen van der
Master's Thesis, June 2006
Crowd Surveillance by Video Camera

Delft University of Technology, The Netherlands
Faculty of Electrical Engineering, Mathematics and Computer Science
Man-Machine Interaction Group

Keywords: people tracking, crowd surveillance, occlusions, DirectShow, image processing,
computer vision

Acknowledgments

In the summer of 2004 I went to the Czech Technical University in Prague, Czech Republic. There, I started working on a project at the Center for Machine Perception (CMP, part of the Faculty of Electrical Engineering) headed by professor Václav Hlaváč. My personal supervisor there was Tomáš Svoboda. My task was to make improvements to the work done by a former Master student, Aleš Fexa [2].

My work was finished under the supervision of professor Leon Rothkrantz at the Delft University of Technology, in the Man-Machine Interaction (MMI) group of the Faculty of Electrical Engineering, Mathematics and Computer Science.

First of all, I want to thank Leon Rothkrantz for giving me the opportunity to go to Prague, for his advice and ideas for improvements and for his support and patience during the time writing this thesis.

Many thanks to Václav Hlaváč and Tomáš Svoboda for accepting me at the Czech Technical University and helping me during my time in Prague.

Finally lots of thanks to my parents for their never-ending efforts to make me finally finish this thesis. I wouldn't be finished yet without them!

Abstract

At the CMP (Center for Machine Perception), part of the Czech Technical University in Prague, a project has been running on crowd surveillance. The goal of the project is the detection and tracking of people. We tried to improve on the earlier results of that project, by making the existing system better usable for tracking of multiple people in a group simultaneously.

Tracking multiple people in groups can be done by using multiple single-person trackers at the same time. However, often these systems are not meant for multimodal applications, so the results are not satisfactory.

Another problem is, that using multiple single-person trackers simultaneously does not work in the case of occlusions. We have augmented the single-person tracking methods available (Condensation and Mean Shift) with capabilities for occlusion detection. The resulting implementation did not have to work in real-time. For test data material from CMP was used, together with some self-made animated videos.

We designed and implemented a demonstration and have tested the system. The model, implementation and test results, will be described in this thesis.

Table of Contents

Acknowledgments.....	iii
Abstract.....	v
1 Introduction.....	1
1.1 Problem background.....	1
1.2 Problem definition.....	2
1.3 Goals.....	2
1.4 Structure of this thesis.....	3
2 Related Work.....	5
2.1 Background subtraction.....	5
2.2 Tracking of people.....	6
2.3 Occlusion detection.....	7
3 Theory.....	9
3.1 Similarity Measures.....	9
3.1.1 Bhattacharyya Coefficient.....	9
3.1.2 Histogram Intersection.....	9
3.1.3 Correlation.....	10
3.2 Background Subtraction.....	10
3.2.1 The algorithm.....	10
3.3 Condensation.....	11
3.3.1 Object representation.....	11
3.3.2 The algorithm.....	11
3.3.3 Connection between background subtraction and Condensation.....	12
3.4 Mean Shift.....	13
3.4.1 Object representation.....	13
3.4.2 The algorithm.....	13
3.5 Occlusion Detection.....	14
3.5.1 Occlusion relation as hidden Condensation process.....	14
3.5.2 A more general approach.....	15
4 Design.....	17
4.1 Global Architecture.....	17
4.2 Background Subtraction Module.....	18
4.3 Tracking Module.....	18
4.3.1 Condensation.....	19
4.3.2 Modifications to the Condensation tracker.....	19

4.3.3 Mean Shift.....	20
4.4 Occlusion Detection Module.....	21
4.4.1 Compare two tracked objects.....	21
4.4.2 Compare N tracked objects.....	22
5 Implementation.....	25
5.1 Tools and API's used.....	25
5.1.1 DirectShow.....	25
5.1.2 GraphEdit.....	26
5.1.3 OpenCV.....	26
5.1.4 Flash.....	26
5.2 Class diagram and descriptions.....	27
5.2.1 CMPTrackingFilter.....	27
5.2.2 BackgroundSubtraction.....	28
5.2.3 Tracking.....	28
5.2.4 TrackedObject.....	28
5.2.5 CondenTracking.....	28
5.2.6 MStracking.....	28
5.2.7 CondenObject.....	28
5.2.8 MSObject.....	29
5.2.9 Ghost.....	29
5.2.10 HistModel.....	29
5.2.11 OcclusionDetection.....	29
5.2.12 OcclusionResult.....	29
6 Testing.....	31
6.1 Test material used.....	31
6.2 Condensation improvement with animated sequence.....	32
6.2.1 Goal.....	32
6.2.2 Results.....	33
6.2.3 Conclusions.....	34
6.3 Condensation improvement with camera sequence.....	34
6.3.1 Goal.....	34
6.3.2 Results.....	34
6.3.3 Conclusions.....	35
6.4 Occlusion detection by comparing two tracked objects using animated sequence.....	36
6.4.1 Goal.....	36
6.4.2 Results.....	36
6.4.3 Conclusions.....	37
6.5 Occlusion detection by comparing two tracked objects using camera sequence.....	37
6.5.1 Goal.....	37
6.5.2 Results.....	38
6.5.3 Conclusions.....	38
6.6 Occlusion detection by comparing N tracked objects using animated sequence.....	39
6.6.1 Goal.....	39
6.6.2 Results.....	39
6.6.3 Conclusions.....	40

6.7 Occlusion detection by comparing N tracked objects using camera sequence.....	40
6.7.1 Goal.....	40
6.7.2 Results.....	40
6.7.3 Conclusions.....	41
7 Conclusions and Recommendations.....	43
7.1 Conclusions.....	43
7.1.1 Improvement in tracking multiple people.....	43
7.1.2 Occlusion Detection.....	43
7.2 Recommendations.....	44
Bibliography.....	45

1 Introduction

Crowd surveillance by tracking people on camera sequences is the subject of this thesis. In this chapter, a short overview of the background of crowd surveillance, and its societal relevance is given. After that, a problem definition is formulated, together with the goals of our research. Finally the global structure of this thesis is explained.

1.1 Problem background

The problem of 'tracking', or detecting and following objects on a video sequence, is a subject of much research in the computer science world. Tracking in general can have a lot of applications, for example in the medical world, where the hands of a surgeon can be tracked so he can operate remotely. The tracking of people however, and especially the tracking of groups of people, has mainly one application: surveillance.



Illustration 1.1: How to use surveillance on a crowd like this?

Surveillance plays a very important role in the world, whether we want it or not. Security is such a large issue everywhere that it can't be ignored. There are of course a lot of social issues about it too, after all, privacy is one the basic democratic rights we have. But still, criminals have been arrested and terrorist attacks have been avoided by making use of images and videos, made by surveillance cameras.

That is why surveillance cameras will not disappear soon, and they can now be seen everywhere where there are a lot of people, or at places where problems could appear easily.

And these cameras have seen a lot of innovation too. 'Smart' cameras help a lot to detect unusual activities in an early stage. But they are nothing more than a tool to warn an operator when there are intruders, or when something else has been detected that can be identified easily (fire and smoke detection, for example).

Large crowds of people, for example at events like sports matches or festivals, or at subway and train stations, are maybe the most important to have under surveillance, because they are both very vulnerable and unreliable. However, surveillance of this kind of groups is still something that has to be done mostly by humans, because it is such a hard problem to solve. Why is this the case?

1.2 Problem definition

To track objects, a lot of algorithms have been developed, based on computer vision methods. Many of these are implemented and work very well, also in real-time applications. So it would not seem a very large problem to just modify these algorithms for tracking many individuals in a group at the same time.

When there is more than one person in the image, multiple trackers can be used in parallel to track these people separately. Unfortunately, problems arise when people are occluded, by each other, or by static objects like buildings and trees. Often trackers switch people around after they have got near to each other, or they lose track of the person at all. So there has to be found a solution to these problems.

Another problem is when one person is standing partly or completely in front of (occluding) an other person. The tracking system does not 'know' that one person is behind the other, so the person at the back can be 'forgotten'. Also the order in which the people stand measured to the camera is not known. These are things that have to be taken care of too in a system that must be capable to track multiple people in crowds.

1.3 Goals

The existing tracking system developed at CMP was not very good at the tracking of multiple people. The system got confused very easily and often, at the end all multiple trackers would wind up tracking the same person. Some modification needed to be made to make it behave better, and that was one of the goals of this research. The other goal was to make the existing tracking system also aware of people occluding each other, and to make it possible to give an estimation of which person is in front and which behind, also in cases when more than two people are involved in an occlusion.

To summarize the goals:

- Design and implement a modification to the existing implemented tracking algorithms so that they are more usable for tracking multiple people simultaneously.
- Design and implement an addition to these algorithms to classify tracked objects that are occluding each other.
- Test these improvements and document the test results.

1.4 Structure of this thesis

The rest of this thesis is organized as follows: in the second chapter an overview of the state-of-the-art in person tracking and especially multiple person tracking is given. The third chapter contains detailed descriptions of the theoretical algorithm that are relevant to our work. Chapter 4 shows the design of the global architecture of the system, and gives a description of the modules of this architecture. Chapter 5 focuses on implementation details, such as descriptions of tools and programs used, and the decomposition of the architectural modules in implemented parts of the system. The sixth chapter describes testing methods and test material; and given a comprehensive overview of every test done including the test results. The last chapter, chapter 7, gives conclusions and recommendations for further research.

2 Related Work

In this chapter, the current state-of-the-art in background subtraction, people tracking and occlusion detection is described. The general subject of “looking at people” is very broad, including not only computer vision aspects, but also cognitive aspects (reasoning on basis of image data), psychological (how do groups move?) and social (privacy is an important issue when you are dealing with surveillance). Comprehensive surveys of all of these aspects, including an overview of applications and descriptions of many implemented systems, are found in [3] and [11]. The more specific and restricted problems mentioned above are described here in more detail.

2.1 Background subtraction

Before tracking can begin, the tracking algorithm needs to know which objects to track. Some kind of motion segmentation needs to be done. One way to do this is by using a technique called *optical flow*. With this technique, every pixel has a corresponding motion vector that is updated every step. However, this is a computationally very expensive method, and so not so useful for real-time applications.



Illustration 2.1: An example of background subtraction

The other, preferred method of separating the background (static parts) from the foreground (moving parts), is to use background subtraction. A model of the background is kept in memory, and when there appears a moving object, that is not consistent with the background model, so the object is seen as foreground. Because changes in lighting or weather can influence the background, most often an adaptive background model is used, that is, one that

is updated every step. The method described by Grimson and Stauffer [12] has proven itself to be a powerful and reliable background subtraction method, using a mixture of Gaussians for modeling the adaptive background.

2.2 Tracking of people

State-of-the-art tracking methods can be divided in several ways. An important first distinction is the way in which the tracked persons are represented. When there is no predefined explicit shape model, some possibilities are a box, an ellipse (generally a more appropriate shape for tracking of humans), the contours of a blob [7], or the blob itself [14]. If there is an explicit shape model, a stick figure can be used, or every body part can have its own box [4].



Illustration 2.2: An example of a single person tracking system

Then the tracking algorithm itself must be chosen. Condensation [7] is a very well-known algorithm in the Computer Vision research with a lot of applications based on it that are already in use. It is actually an example of a *particle filter*, also known as a *Sequential Monte Carlo* method. The idea of a particle filter is that random sampling is used to estimate a Bayesian model. Large advantages of Condensation is that it is possible to use it in a very broad range of applications, and that it is usable more or less independent of the object representation. A problem is that particle filters are not very good for use in multimodal applications (tracking of more than one object at the same time), so Condensation does suffer the same problem. Improvements that have been suggested are to mitigate this problem, for example to use a MCMC (Markov Chain Monte Carlo) based particle filter [8], or to use a mixture of particle filters to create one multimodal particle filter [13].

A less known alternative for Condensation is tracking based on the Mean Shift algorithm [1]. Mean shift is an older theoretical method, but can be applied to tracking in a very promising way. The 'mean shift' is the estimated direction and distance in which the target moves, and this is computed by comparing an already defined model target with the current candidate target. The targets are defined by their color distribution (histogram). The advantages of this method is that no dynamic model is needed in advance. It is a fast and reliable procedure that has started to be used in many tracking systems.

2.3 Occlusion detection

Given that there is a reliable multiple-person tracking system in place, the occlusion detection can start to find the depth ordering of occluding objects. Several methods have been proposed to solve this problem.



Illustration 2.3: An example of occlusion detection

One solution, described in [5], adds occlusion handling to Condensation. A virtual object representing the occlusion relation between two objects is subjected to the same sampling process as all normal Condensation objects.

Another method is used when objects are represented as blobs [4], [9]. In these systems, when moving blobs merge into one and then split again, it is seen as the assembling and disassembling of groups of people. If the group consists of more people, it is segmented into its constituent people. Also the detection of body parts, like the head, can be used to count the number of people in the group. This method does not give real depth information, but it still can quickly find people in a group.

3 Theory

The algorithms on which this work is based, are explained fairly detailed in this chapter. A basic understanding of all these algorithms is useful, to have a better understanding later on.

In the first place some information about similarity measures is given. They are used multiple times in the algorithms so it should be known how they work.

A background subtraction algorithm that separates the foreground from the background is described. Then, a tracking algorithm does the real tracking of the foreground objects, that have been detected by the background subtraction. There are two algorithms described: Condensation and Mean Shift. Finally, the theory of the occlusion detection method is given.

3.1 Similarity Measures

Because color histograms are often used to give an indication of what is in a region of an image, a *similarity measure* (or *distance measure*) has to be defined to compare these histograms. Two normalized histograms $p = \{p^{(u)}\}_{u=1 \dots m}$ and $q = \{q^{(u)}\}_{u=1 \dots m}$ are considered, each having m bins.

3.1.1 Bhattacharyya Coefficient

The Bhattacharyya coefficient is a very popular method to compare distributions. For p and q , it is defined as:

$$\rho[p, q] = \sum_{u=1}^m \sqrt{p^{(u)} q^{(u)}} \quad (3.1)$$

3.1.2 Histogram Intersection

An alternative is the histogram intersection. It is defined as:

$$\rho[p, q] = \sum_{u=1}^m \min(p^{(u)}, q^{(u)}) \quad (3.2)$$

3.1.3 Correlation

A third possibility is to use the correlation of the histograms:

$$\rho[p, q] = \frac{\sum_{u=1}^m p^{(u)} q^{(u)}}{\sqrt{\sum_{u=1}^m p^{(u)} \sum_{u=1}^m q^{(u)}}} \quad (3.3)$$

where

$$p^{(u)} = p^{(u)} - \frac{\sum_{i=1}^m p^{(i)}}{m} \quad (3.4)$$

and the same for $q^{(u)}$.

3.2 Background Subtraction

As seen in the last chapter, background subtraction is the first step on the way to the tracking of objects. The algorithm of Stauffer and Grimson [12], mentioned there, will be explained here.

The background subtraction is a very important phase in the tracking process, the changing of parameters here can influence the tracking results heavily, so it is useful to know the basics of the working of the algorithm. However, because the background subtraction is not directly influencing the occlusion detection, it is not described in as much detail as the other algorithms.

3.2.1 The algorithm

The background of every pixel is modeled as a mixture of Gaussians, so for every pixel there are K normal distributions. Each normal distribution has a mean μ , a standard deviation σ^2 and a weight ω .

Because there is a multiple of distributions, every pixel has more than one background color. In this way, a slightly moving background will be detected. It also has the another advantage: when an object is standing still for a long time, it may be detected as being part of the background. With the approach used in this algorithm, the old background will reappear when the object moves again, as it still has its own distribution in the mixture of Gaussians. The weight may have been lower for a time, but now it gets up again.

During every frame, for all pixels that get a new value, all the distributions for the pixel are tried one for one until a match has been found. There is a match when the new pixel value is within 2.5 times the standard deviation. When there is no match at all, the least probable distribution is replaced by a new one with the value of the pixel as mean, a high standard deviation and a low weight.

The weight of every distribution is updated: the matched distribution gets a higher weight, and the rest a lower one. A parameter α , the learning rate, controls the amount with which the weights are updated.

The matched distribution does also get an updated mean and standard deviation. The mean moves in the direction of the pixel value. Again, the learning rate controls the amount of change.

After this is done, the decision must be taken: does the pixel belong to background or foreground? All the distributions are sorted by ω / σ . If the sum of the weights of all distributions with a higher ω / σ than the matched one (all the probable background distributions that did not have a match) is larger than a certain threshold T , the pixel will be classified as foreground. This classification is possible, because when there are a lot of background distributions that did not match, the odds are that the pixel is not part of the background at all.

The distribution that is the most probable background model, and that will because of that have a match often, will bubble to the top of the list, and stay there if nothing changes. When there is a moving or slightly changing background, there will be a set of most probable background models of which the top one will change once in a while.

Because there are only two parameters, the learning rate α , and the threshold for the number of background models T , this algorithm is very easy to use. Despite this simplicity the results are very reliable.

3.3 Condensation

The first real tracking algorithm used is Condensation [7]. The original paper focused on an application in tracking curves in visual clutter; for example tracking the contours of a hand over a complicated background. However, it can also be easily applied to tracking of persons (or other moving objects), and that is what this description focuses on.

3.3.1 Object representation

Tracked objects can be represented in different ways, but in our case, every tracked object is represented by an ellipse. These ellipses are compared by their color histogram, with one of the similarity measures described above. In the implementation used, the ellipse is split in an upper and a lower part, and the color histograms of these parts are compared separately. In this way people with different colors of clothing can be tracked more reliably.

3.3.2 The algorithm

Condensation makes use of factored sampling, applied iteratively to successive images (for example video frames). The basic idea of factored sampling is that a probability distribution (in this case, of the state of the object) is approximated by a set of weighted samples $S = \{(s^{(n)}, \pi^{(n)}) \mid n = 1 \dots N\}$. So, every sample of this set represents a hypothetical state s of the object together with the probability π that this sample will be chosen (the weight).

The state s is defined as:

$$s = x, y, v_x, v_y, a_x, a_y, b, v_b \quad (3.5)$$

where x and y are the position of the ellipse, v_x and v_y the velocity, a_x and a_y the acceleration, b the length of the main axis and v_b the the change of the axis length (growth).

The first step in every iteration is that a new set of N samples is chosen from S_{t-1} , the sample set from last iteration. Each sample has a probability π that it is chosen. Samples can be chosen more than one time, so those having a higher weight will appear more often in the resulting set. In the first iteration S consists of random samples with random weights.

All the samples are then subjected to a dynamic model, to give a prediction of the new position:

$$s_t = A s_{t-1} + w_t - 1 \quad (3.6)$$

where A is the deterministic dynamic model, and w_{t-1} a random Gaussian variable.

Then, for all the samples the color histogram for the resulting ellipse is computed. These are compared with the model histogram. The weights are updated as a result of this comparison: more similar samples get higher weights.

Now the new sample S , which is used in the next iteration, is derived. The mean state of this set is chosen as the new position of the target:

$$E[S_t] = \sum_{n=1}^N \pi_t^{(n)} s_t^{(n)} \quad (3.7)$$

3.3.3 Connection between background subtraction and Condensation

How does the Condensation algorithm know which objects to track? The background subtraction algorithm needs to give it some information as to which parts of the image are interesting to follow. A way in which to do this, is to run the background subtraction algorithm, and then to find connected components based on the foreground pixels. The components that have an area larger than a predefined threshold, will be used as tracked objects in Condensation. When blobs are used as object representation, nothing will have to be done, but with ellipses, as described above, the best fitting ellipses for the connected components will have to be found.

3.4 Mean Shift

The other tracking algorithm that was tested is described in [1] and is based on the idea of mean shift. The 'mean shift' is an estimated direction and distance in which the target moves, and this is computed without using a dynamic model, but only by comparing a candidate target with the model.

3.4.1 Object representation

As with the Condensation algorithm, in the Mean Shift approach every object is represented by an ellipse. When the algorithm is initialized, for every object that has to be tracked, the model color histogram of the ellipse is computed.

The histogram is computed in the Region of Interest (ROI), the pixels inside the ellipse (in fact, it's a box bordering the ellipse on the outside). To increase the robustness, to every histogram a convex and monotonic *kernel mask* is added: pixels in the center of the ellipse get a higher value than the ones on the border. The weight decreases with squared distance from center. After this, the histogram is normalized.

All the histograms used are in three dimensions (one dimension for each color). To compute the distances between the histograms, the Bhattacharyya distance is used.

3.4.2 The algorithm

During every frame processed, the target moves toward its most probable position in multiple iterations. To do this an algorithm is used that maximizes the Bhattacharyya coefficient (higher coefficient means a higher similarity, and thus a shorter distance).

1. First for candidate location y_0 , the current candidate histogram $p(y_0)$ will be computed, together with the kernel mask. After that the Bhattacharyya coefficient between this histogram and the model histogram q is computed.
2. For every pixel, compute a weight as defined by:

$$w_i = \sum_{u=1}^m \delta[b(x_i) - u] \sqrt{\frac{\hat{q}_u}{\hat{p}_u(\hat{y}_0)}} \quad (3.8)$$

where $b(x_i)$ is the bin for the color of pixel x_i , u is the current bin and δ is the Kronecker delta function, which is true only if both its arguments are true. So this means that every weight is the square root of the value of the model bin of the pixel color, divided by the value of the candidate bin of the pixel color.

3. Compute the *mean shift*, that is compute the new estimated location y_l :

$$\hat{y}_1 = \frac{\sum_{i=1}^n x_i w_i}{\sum_{i=1}^n w_i} \quad (3.9)$$

Then, again compute the Bhattacharyya coefficient, now between the new candidate histogram $p(y_l)$ and the model histogram q .

4. As long as the coefficient between $p(y_0)$ and q is larger than the one between $p(y_l)$ and q the target has not yet been reached, so the location of y_l must be updated: $y_l \leftarrow \frac{1}{2} (y_0 + y_l)$. Repeat this step until the target has been reached.
5. If $\|y_l - y_0\| < \varepsilon$, stop the iterations, and continue on to the next frame. Otherwise, start a new iteration at step 1 with the new candidate ellipse: $y_0 \leftarrow y_l$.

3.5 Occlusion Detection

There exist several methods for computing the occlusions between tracked people, a few of which are discussed here.

3.5.1 Occlusion relation as hidden Condensation process

In the paper by Hu, Hu and Tan [5], a method is described to do detection of occlusions, by introducing a 'hidden' object to the Condensation algorithm, that models the occlusion relationship between two objects. The occlusion relation for two objects A and B on time t is noted by $\varphi_t \in \{0, 1, 2\}$ where $\varphi_t = 0$ means no occlusion is happening, $\varphi_t = 1$ indicates that A occludes B and $\varphi_t = 2$ indicates that B occludes A .

The occlusion relation φ is sampled like normal Condensation objects. It is also subjected to a dynamic model, which can be expressed by a transition matrix, for example:

$$p(\varphi_{t+1}=j|\varphi_t=i) = \begin{bmatrix} 0.8 & 0.1 & 0.1 \\ 0.1 & 0.8 & 0.1 \\ 0.1 & 0.1 & 0.8 \end{bmatrix} i, j \in \{0, 1, 2\} \quad (3.10)$$

So there is a chance of 0.8 that the occlusion relation stays the same, and 0.1 that it will change.

When the mean sample $E[S_t]$ in the Condensation process is computed, there will also be a computation of the new estimated φ value:

$$E[\varphi_t] = \arg \max_{\varphi} \sum_{\varphi_t^{(n)} = \varphi} \pi_t^{(n)}, \quad \varphi \in \{0, 1, 2\} \quad (3.11)$$

The drawbacks of this method are that there are no special considerations for the case when more than two persons are occluding each other. Also, it is targeted for use with the Condensation method, so it is not easy to adapt it for other methods like Mean Shift.

3.5.2 A more general approach

McKenna et al. [9] describe a method to view groups as blobs that split and merge. Tracked objects are all represented as blobs. The color information of the object is represented by a histogram. This histogram can be used as the basis of a probability function giving the chance at a certain (color) value \mathbf{x} given a person i :

$$P(\mathbf{x}|i) = \frac{H_i(\mathbf{x})}{A_i} \quad (3.12)$$

where $H_i(\mathbf{x})$ is the histogram value and A_i is the area of the object. During the tracking, this histogram is updated adaptively every frame.

If nothing is known about the depth ordering of people in a group, the probability of a pixel corresponding to the i th person $P(i)$ is estimated as:

$$P(i) = \frac{A_i}{\sum_{j \in G} A_j} \quad (3.13)$$

where G is the group that the i th person is part of. When these things are known, the posterior density can be computed by combining them:

$$P(i|\mathbf{x}) = \frac{P(\mathbf{x}|i)P(i)}{\sum_{j \in G} P(\mathbf{x}|j)P(j)} \quad (3.14)$$

From this, also a visibility index (the part of the person that is not occluded) can easily be computed. So, in fact, this is also a sort of depth ordering algorithm. However it is not applicable to tracking systems where the objects tracked are discrete units, not changing blobs that can split and merge. So, unfortunately, it is also not applicable to Condensation or Mean Shift.

4 Design

In this chapter, the design of the system is given. First, a model is given of the system architecture. Next, for every module of the system architecture, the design of the algorithms is explained.

4.1 Global Architecture

An overview of the global architecture is given in the diagram below. The blue-colored parts were added or changed from the original architecture. A detailed description of all the parts of the diagram follows underneath.

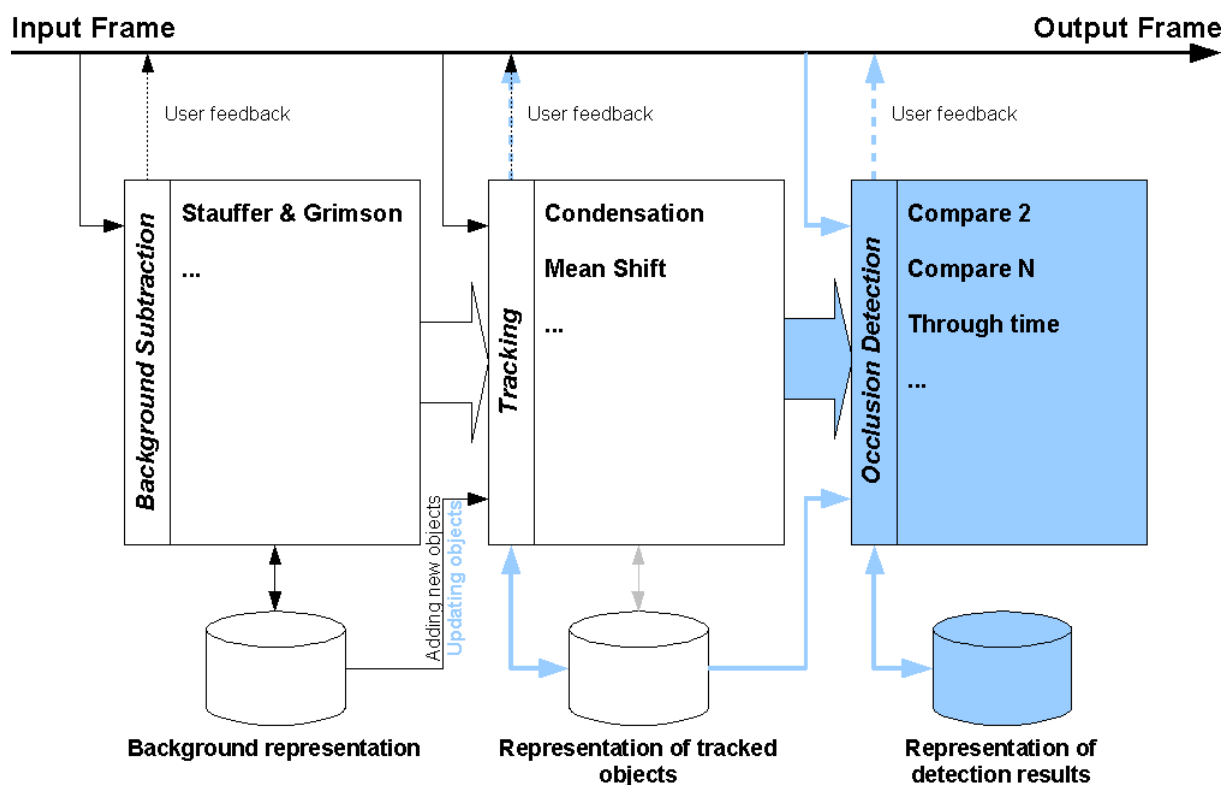


Illustration 4.1: Architecture Diagram

The diagram shows the control and data flow for one frame of an image sequence. The control flow is given by the large block arrows: so, first the background subtraction algorithm runs, then the tracking algorithm, and finally the occlusion detection.

These three phases can all have multiple implementations: in the case of the background subtraction, there is only the Stauffer and Grimson method; for the tracking, there are the Condensation and Mean Shift methods, and for the occlusion detection, there is the 'naive' comparing of two objects and the comparing of N objects. The third method reuses the occlusion detection results of the last frame, but it is not implemented.

On the bottom can be seen where the data is stored. The representation of the background subtraction results is dependent on the method that is chosen. With the tracking algorithm, this was also the case; this is changed so now the representation of the tracked objects is independent of the tracking method that is chosen. The tracking methods had to be changed rather drastically to accommodate this change, however the benefit is that there is now a more generic framework, to which, for example, the occlusion detection can be added. For the occlusion detection itself, the results are stored independently of the method chosen.

The background representation data is now also used for updating the tracked objects when the Condensation method is used. Before, the data was used only to identify new objects to track and add them to the tracked objects.

4.2 Background Subtraction Module

In the previous chapter, it was mentioned that the background subtraction plays a very important role in the tracking process, but that the influence on the real tracking algorithms is indirect, so there was no need for a very comprehensive description. The same could be said in this chapter; also because no additions were planned to the background subtraction algorithms themselves, a complete description of the design is not really necessary.

The thing that should be kept in mind is that the background subtraction module returns as its result a background representation, that consists of all the pixels that are considered background. So, the other pixels are considered to be part of the foreground.

4.3 Tracking Module

Both the Condensation and the Mean Shift algorithm are described here in pseudocode. Also the design for the modifications made to the Condensation algorithm is given. The pseudocode given is executed every frame when the algorithm is called.

4.3.1 Condensation

In pseudocode, the design of the Condensation algorithm can be given as follows:

```
choose set of n samples from old sample set
apply dynamic model to all the samples
update the weights of the samples based on comparison of
    histogram with model histogram
compute the mean sample
```

Illustration 4.2: Pseudocode of Condensation algorithm

4.3.2 Modifications to the Condensation tracker

There was an improvement made to the way in which the weights of the Condensation objects are updated, normally the weights get their value only because of the similarity of the sample histogram to the model histogram. But because all the background subtraction data is of course still available, this could also be incorporated in the computation of the weight.

So, in addition to the histogram comparison, now all the foreground pixels existing in the sample ellipse (as found by the background subtraction algorithm) are counted; and the ratio of this number of foreground pixels to the total number of pixels in the ellipse, is taken as another measure of the usefulness of the ellipse.

The ratio in which this new measure is responsible for the Condensation weight of the sample, is controlled by the parameter r . So in pseudocode the weight is now defined as:

```
foreground_confidence = (# of foreground pixels in ellipse) /
    (total # of pixels in ellipse)
weight = r * foreground_confidence
    + (r - 1) * confidence of the color histogram
```

Illustration 4.3: Pseudocode of Condensation improvement

4.3.3 Mean Shift

The design for the mean shift algorithm is given below in pseudocode. The steps 1-5 are the same as in the theoretical description in chapter 3.

```
step1:  compute  current  candidate  histogram,  and
        Bhattacharyya distance between model & cand. histogram
step2:  create a new weights histogram with each bin containing
        sqrt(value of the model / value of the candidate)
step3:  compute
        m00 = sum of all weights
        m01 = sum of (weight * x value of pixel with this color)
        m10 = sum of (weight * y value of pixel with this color)
        mean shift in direction x = m10 / m00 - width / 2
        mean shift in direction y = m01 / m00 - height / 2
        shift candidate ellipse in computed direction
        compute histogram of shifted ellipse, and Bhattacharyya
        distance between model and shifted histogram
step4:  while (distance from step 1 > just computed distance)
        shift candidate ellipse with half mean shift
        compute histogram of shifted ellipse, and
        compute Bhattacharyya distance between model
        and shifted histogram
step5:  if ((mean shift in direction x)^2 +
        (mean shift in direction y)^2) < epsilon
        stop iterations
    else
        new candidate = shifted ellipse
        continue to next iteration
```

Illustration 4.4: Pseudocode of Mean Shift algorithm

4.4 Occlusion Detection Module

For the occlusion detection, some new algorithms were implemented. Here their design is shown in pseudocode and clarifying diagrams.

4.4.1 Compare two tracked objects

This method just does the following: when there is an overlap between two objects, the most probable object, given this overlapping region and its histogram, is defined as being on top. So, all overlapping regions between two objects in an image are checked one by one, to determine which one of the two ellipses is on top.

In addition, some mechanism is added that keeps track of which comparisons have been done already, otherwise some combinations would be added twice.

In the example picture, three ellipses are overlapping each other. First the overlap of 1 and 2 (the green rectangle) is compared with ellipses 1 and 2 themselves. Because ellipse 2 is the one on top, the histogram of the overlap should be more similar to the histogram of ellipse 2 than that of 1.

Then the overlap of 1 and 3 (the red rectangle) is compared with 1 and 3 themselves. 3 will be found to lie on top.

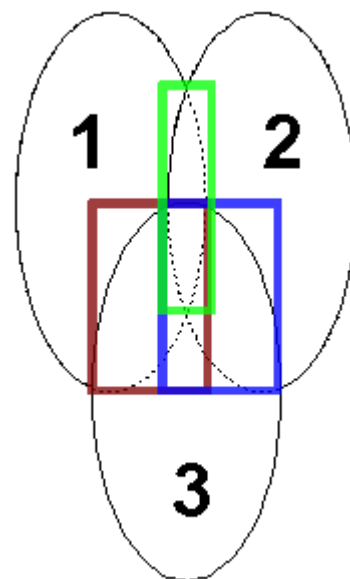


Illustration 4.5: Compare sets of two objects

After that the combination of 2 and 1 doesn't have to be checked again, but the overlap between 2 and 3 still has to be computed (the blue rectangle) and has to be compared. Again, 3 will be on top. For ellipse 3, no comparisons are left to be done.

In pseudocode the algorithm can be given as:

```

for all tracked objects {
    get overlapping region with every other object
    if overlap
        compute color histogram of overlap
        compare histogram with model histograms of both objects
        the best match is the object that lies on top
}

```

Illustration 4.6: Pseudocode for the compare two objects algorithm

4.4.2 Compare N tracked objects

Now, also multiple overlapping objects will be detected. For this a recursive algorithm is necessary. We start with the same example: three overlapping ellipses.

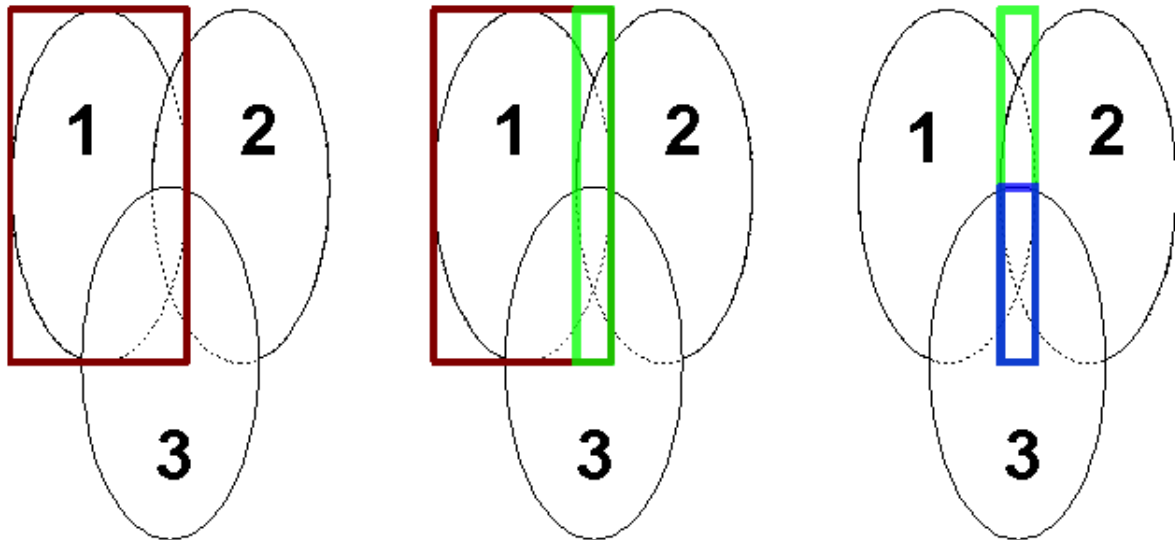


Illustration 4.7: Compare sets of N objects recursively

Ellipse 1 is taken as a starting point. The 'current overlap' will be set at the initiation to the complete area of ellipse 1 (the red rectangle). Ellipse number 1 itself will be removed from the list of objects that still have to be processed. The function will now be called recursively for all the other objects.

The next one is ellipse number 2. The overlap with the 'current overlap' is computed, and this can be seen in the middle of the image as the green rectangle. This new overlap will now be the new current overlap. Ellipse number 2 will be removed from the list and only ellipse 3 is left.

So, in the next (and last) round, ellipse number 3 is compared with the current overlap, and this can be seen on the right as the blue rectangle.

Now the results have to be returned: the distance between current histogram (number 3) and current overlap (the blue area) is computed, and returned to the calling function. Here, ellipse number 2 was the current ellipse, and the green area was the current overlap. So, this distance will be computed too, and the result will be multiplied with the earlier result given back. When we come back to the left of the picture this will be done again, the confidence is of course 1, because the current overlap was equal to the current ellipse (the red area).

What we now have as end result, is the confidence that the combination 1-2-3 really is the order in which the objects are 'stacked', with 3 being on top. Because all combinations are checked one by one recursively, all the confidences for the combinations 1-2-3, 1-3-2, 2-1-3, 2-3-1, 3-1-2 and 3-2-1 will be returned in the end. The combination which returned the highest confidence, will be chosen.

Because it is of course possible that situations as in illustration 4.8 occur (or much more complicated situations which come down to the same), multiple results may have to be returned by the recursive function. This is because ellipse 1 has to compare with 2 and with 3, but these are two different 'clusters' of overlaps. This complicates the algorithm, but the principle stays the same.

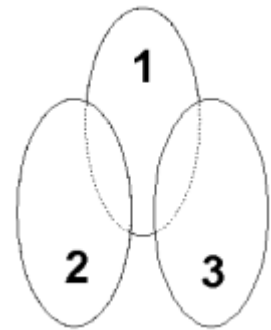


Illustration 4.8: Special situation

In pseudocode, the algorithm is expressed as follows:

```
function occl_recursive(objects, current_overlap) {
  for all objects {
    if there is overlap with current_overlap {
      compute new_overlap
      prob = probability that object is on top
    }
    else continue with next object
    new_objects = objects - current object
    results = occl_recursive(new_objects, new_overlap)
    if results for each result {
      return (result_probability * prob)
      and current object id
    }
    else
      return 1 and current object id
  }
}
```

Illustration 4.9: Pseudocode for compare N objects algorithm

5 Implementation

In this chapter, details about the implementation are given: which tools are used, and how the parts of the system defined in the last chapter were mapped to the classes that are implemented.

5.1 Tools and API's used

The application was written in C++, making use of Visual Studio 6.0. Besides that a number of tools and API's (Application Programming Interfaces) were used.

5.1.1 DirectShow

The tracking system needs a way to receive input frames from a video sequence. It also needs a way to show output frames to a screen, or to save it to a file. For that goal, DirectShow is used.

DirectShow is an API for developing multimedia applications under Windows. In modern Windows versions, it is an integral part of the operating system and used in a lot of often-used applications. It provides a framework to create *filters* that can do a specific job. Examples of DirectShow filters are:

- Reading an .avi file from your hard drive
- Decoding an MP3-compressed audio file
- Showing a video sequence on your screen

The filter can be connected to form a *graph*, which represents the complete control flow from input source to output. An example of a graph is given below in the description of GraphEdit.

DirectShow used to be a part of DirectX [10], a set of API's for all kinds of multimedia development (Direct3D, DirectSound, etc.). For development of DirectShow filters, it was necessary to download the DirectX SDK (Software Development Kit). In 2005, DirectShow was moved out of DirectX, to the Platform SDK, in which all basic things needed to develop for the Windows platform can be found.

However, as these newer versions are designed to work with Visual Studio.NET or newer, and not with Visual Studio 6.0 anymore, they were not used. To switch to a newer Visual Studio version would not be worth the effort. So, for DirectShow the DirectX SDK from summer 2004 was used.

5.1.2 GraphEdit

GraphEdit is a free tool (distributed with DirectShow) for editing and testing DirectShow graphs. In the picture below, a screenshot can be seen of an example graph in which the CMP Tracking Filter shows up. GraphEdit was used for all testing and for generating the output movies, from which the frames shown in the next chapter are taken.

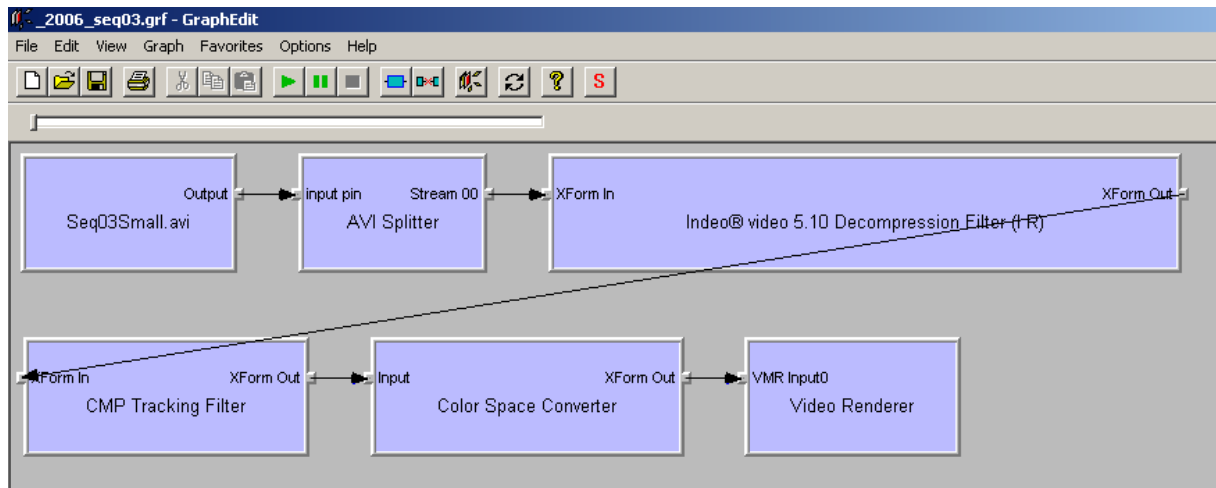


Illustration 5.1: Screenshot of GraphEdit

5.1.3 OpenCV

Intel Open Source Computer Vision Library (OpenCV) [6] is an multi-platform open source toolkit from Intel, that provides a lot of algorithms (in the form of C functions) from different fields of Computer Vision. For example, it provides implementations of both Condensation and Mean Shift, although support for the latter is very limited and was not used. Among the functions offered are edge detection, handling of color histograms, and a lot of basic functions for drawing all kinds of shapes and text to an image.

5.1.4 Flash

Macromedia Flash was used to create the animated test sequences. It is very easy to create moving vector graphics with this program, so it was chosen to make some tests quickly. The Flash animations were exported to normal .avi files before using them for testing.

5.2 Class diagram and descriptions

In this section a mapping of the architectural modules described in section 4.1 to the classes that were implemented is given. All these classes are collected in one project `CMPTackingFilter` that works completely independent of the `DirectShow` framework. The system was built modularized, so an algorithm can be replaced by the new one as long as the new one provides the same interface.

Another separate project, `CMPTackingFilter` (renamed from just 'TrackingFilter'), was made as a wrapper for the `CMPTackingFilter` to make it behave like a `DirectShow` filter. A template was used to generate the most of this code.

The class diagram below shows all the classes and their relationship. The three blue planes correspond to the modules defined in the last chapter: on the top, the background subtraction module; in the middle, the tracking module; and on the bottom right, the occlusion detection module.

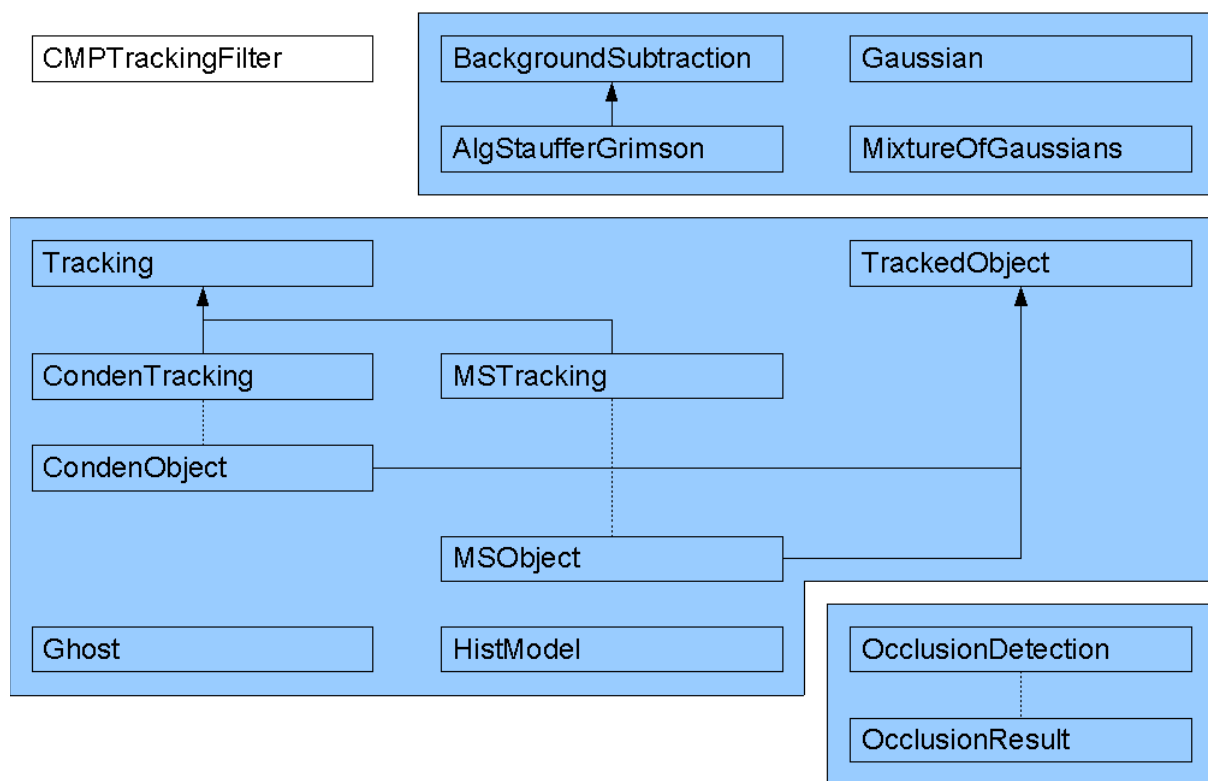


Illustration 5.2: Class Diagram

5.2.1 `CMPTackingFilter`

This is the main class. It contains the method `init()` which is called by the `DirectShow` wrapper when it is started and `processFrame()` which is called every frame. In here the preferred algorithms to be used for background subtraction and the tracking itself can be selected.

5.2.2 BackgroundSubtraction

Represents a generic background subtraction method that can be implemented by subclasses. The only algorithm implemented is in `AlgStaufferGrimson`. Helper classes are necessary for the Stauffer and Grimson algorithm: `Gaussian` and `MixtureOfGaussians`. Because the background subtraction algorithm was not modified (it is only used in more places now) it is not necessary to describe these classes in more detail.

5.2.3 Tracking

Represents a generic tracking method. The virtual method `process()` is called once every frame, after the background subtraction is finished. It must be implemented by one of more subclasses. It also has some method which are used in multiple algorithm implementations, such as `draw()`, which is used to draw user feedback like location of the tracked objects to the output frame.

5.2.4 TrackedObject

Represents a generic tracked object. It has subclasses for every tracking algorithm making use of it, `CondenObject` and `CondenTracking` are now implemented.

5.2.5 CondenTracking

Implements the Condensation algorithm. In the `process()`-method, the objects that have already be found are updated; and large blobs that are newly detected by the background subtraction algorithm are added. The class also takes care of the logging of data to a text file, and draws the ellipses to the output frame. Uses the `CondenObject` class as representation of the tracked objects. It also uses the `Ghost` and `HistModel` helper class.

5.2.6 MSTracking

Implements the Mean Shift algorithm in its `process()`-method. Uses the `MSObject` class as representation of the tracked objects. It also uses the `Ghost` helper class.

5.2.7 CondenObject

Implements an object tracked by the Condensation algorithm as implemented by the `CondenTracking` class.

5.2.8 *MObject*

Implements an object tracked by the Mean Shift algorithm as implemented by the `MSTracking` class.

5.2.9 *Ghost*

A helper class that represents the 'hole' that a target leaves after moving away. It protects the location from a false initialization.

5.2.10 *HistModel*

A helper class that represents the color histogram of a certain region. It contains methods to compare these histograms using different algorithms.

5.2.11 *OcclusionDetection*

Implements the occlusion detection algorithms. It could be split in subclasses for every algorithm used, but for now, that is not really necessary. Uses `OcclusionResult` for storing the results.

5.2.12 *OcclusionResult*

Contains the results obtained by the occlusion detection. Stores the order of the occluding objects (referenced by their ID) and the confidence belonging to that order.

6 Testing

In this chapter, first the test material used is described. After that all the tests are described one by one, together with the results of the tests and illustrated with screenshots showing the behavior.

6.1 Test material used

The test material used can be divided in two groups: material obtained from a camera, that was available at CMP from previous projects; and animated material, that was self-made using Macromedia Flash. In the tables underneath titles and descriptions of these sequences can be found.



<i>Camera</i>	<i>Title</i>	<i>Description</i>
	Seq03Small.avi	Several people moving in front of a camera and sometimes moving out of view and back in. Length 1650 frames.
	vid3.avi	A lot of people moving in a room, with abrupt changes in the movement and a lot of occlusions. Length 1800 frames.

Table 6.1: Camera test material used

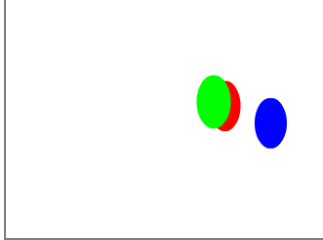
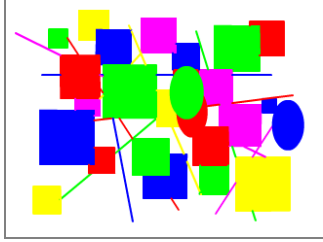
<i>Animated</i>	<i>Title</i>	<i>Description</i>
	triple_overlap.avi	Three ellipses with primary colors moving and occluding each other. Length 100 frames.
	triple_overlap_noisy.avi	Three ellipses with primary colors moving and occluding each other, over a noisy background. Length 100 frames.

Table 6.2: Animated test material used

6.2 Condensation improvement with animated sequence

This is the first test for improvement in Condensation tracking that has resulted from the incorporation of the number of foreground pixels in the sample weight computation, as described in paragraph 4.3.2. The animated sequence 'triple_overlap_noisy.avi' is used.

6.2.1 Goal

The influence of parameter r (see 4.3.2) on tracking in general, and especially on tracking of multiple objects simultaneously is tested. For different values of r (0, 0.25 and 0.5) the Condensation tracker will run, and the tracking performance will be seen by looking at the feedback given (the ellipses drawn on the screen).

When r is 0, the additions to the algorithm are not used at all, so the original tracking algorithm is tested. In this way, results can be compared.

Because in the test sequence 'triple_overlap_noisy' all objects that have to be tracked move according to a dynamic model, it is expected that the effect of changing r will be seen here clearly. Because of the background noise, it is not easy for the tracker, this is better for testing because without background noise, it is expected that all objects will be tracked flawlessly anyhow.

6.2.2 Results

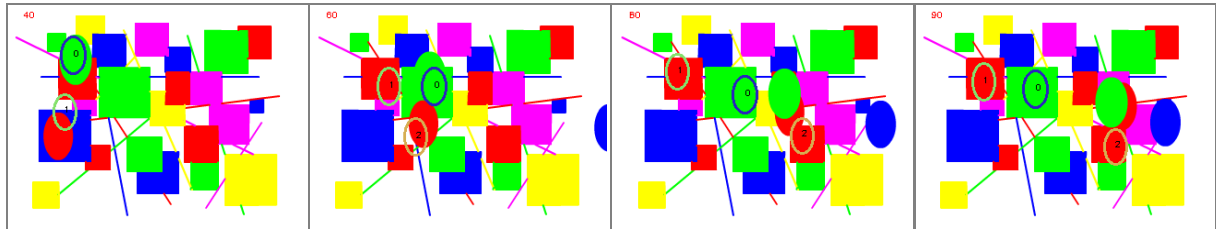


Illustration 6.1: Frames 40, 60, 80 and 90 with $r = 0$

This are the results with $r = 0$. Already in frame 40, the tracker for the red ellipse (#1) is moving towards another red object. In frame 60, another tracker has started for the same red object! In frame 90, all ellipses are lost.

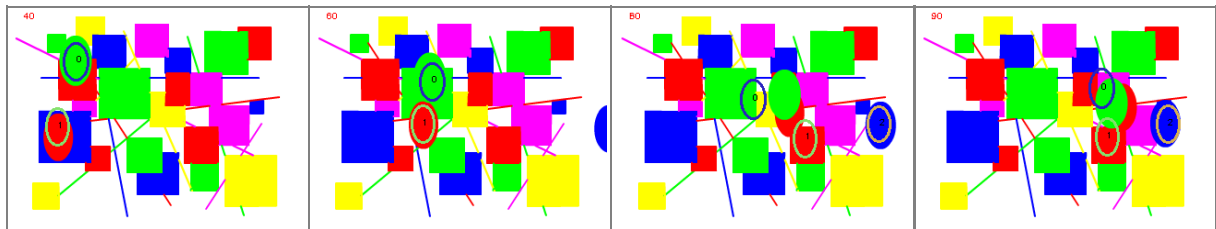


Illustration 6.2: Frames 40, 60, 80 and 90 with $r = 0.25$

With $r = 0.25$, performance is slightly better. In frame 40 and 60 there is no error. Frame 80 and 90 show that the trackers for ellipses #0 and #1 are more or less lost, but the one for #2 works perfectly.

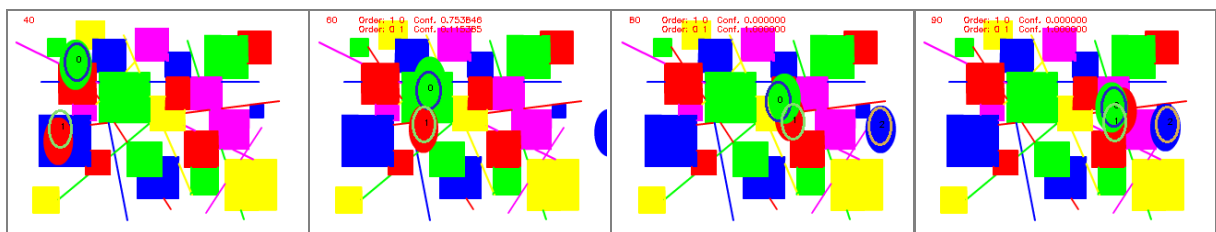


Illustration 6.3: Frames 40, 60, 80 and 90 with $r = 0.5$

With $r = 0.5$, we see that all three ellipses are almost perfectly tracked during the whole sequence. Where things did go wrong before, like in frame 80, now the right object is tracked (with the same color as part of the background) because it is moving all the time.

6.2.3 Conclusions

When r is 0, the tracker only gives completely right results for the first 35 frames. Because there are 100 frames in total, this gives a score of 35%. When r is 0.25, the first 67 frames are tracked perfectly, a score of 67%. Finally, when r is 0.5, all ellipses are tracked correctly during the complete sequence, so a score of 100% is reached!

We can conclude that the introduction of parameter r has helped at least in one circumstance. Most probably, this is because the movements of the ellipses are not random, but constant speed and in one direction. More experiments have to follow before we can conclude that the using of background model data in Condensation has helped in every case.

6.3 Condensation improvement with camera sequence

This is the second test for improvement in Condensation tracking that has resulted from the incorporation of the number of foreground pixels in the sample weight computation. Now, the camera sequence 'vid3.avi' is used.

6.3.1 Goal

Again, the influence of parameter r on tracking in general, and especially on tracking of multiple objects simultaneously is tested. For different values of r (0, 0.25 and 0.5) the Condensation tracker will run, and the tracking performance will be seen by looking at the feedback given (the ellipses drawn on the screen).

Like before, when r is 0, the additions are not used at all, and the original tracking algorithm is tested.

6.3.2 Results



Illustration 6.4: Frames 351, 532 and 562 with $r = 0$

With $r = 0$, so when the tracking solely depends on the Condensation samples and not on the background detection too, the three persons seen in these screenshots are tracked quite good. Only in frame 532 there is some confusion, the blue ellipse for person #0 is too far to the left, and the green ellipse for person #1 is a little bit too much to the right. However, in frame 562 everybody is found again in the right place.



Illustration 6.5: Frames 351, 532 and 562 with $r = 0.25$

When $r = 0.25$, the performance is slightly better in frame 532, but the increase is not really very noticeable. The only thing is that now, there no triple overlap anymore, only two times an overlap of two ellipses.



Illustration 6.6: Frames 351, 532 and 562 with $r = 0.5$

Lastly, when $r = 0.5$, the performance noticeably drops. In frame 351, everything is still alright, but after that things go wrong. Especially the green ellipse has difficulty in finding the right person, and in frame 562, it is tracking a complete different person.

6.3.3 Conclusions

When we take only the first 600 frames into account (after that, a fourth person joins the others), the following scores are reached: if r is 0, only 40 frames give a completely wrong result: a score of 93%. When r is 0.25, just 10 frames are wrong, giving a score of 98%. In the last case, when r is 0.5, the results are a lot worse, 80 frames having a bad result. The score is just 87%.

We see that in camera sequences, the introduction of parameter r has not helped in every circumstance. Because the movements of the people were random, there is not much use in using the background detection information, only the comparing of the color histograms can give a reliable estimation whether the right person is tracked.

From now on, r will be set at 0.25. Hopefully this will represent the best of both worlds in that the background detection information is not thrown away completely, but also good performance in the case of random behavior is preserved.

6.4 Occlusion detection by comparing two tracked objects using animated sequence

In this test, the first algorithm for occlusion detection, as described in paragraph 4.4.1, will be tested. So, if there is an overlap between two ellipses the one with the highest confidence (that is, the one with the highest mean weight according to the Condensation algorithm) is on top. The animated sequence 'triple_overlap.avi' is used.

6.4.1 Goal

The goal of this test is to see whether the algorithm of 4.4.1 works, by looking at the returned results of the occlusion detection, and seeing if they are right or wrong. Because of the uniformly colored ellipses it is very easy to see which one is occluding the other, also in cases where more than two are overlapping.

6.4.2 Results

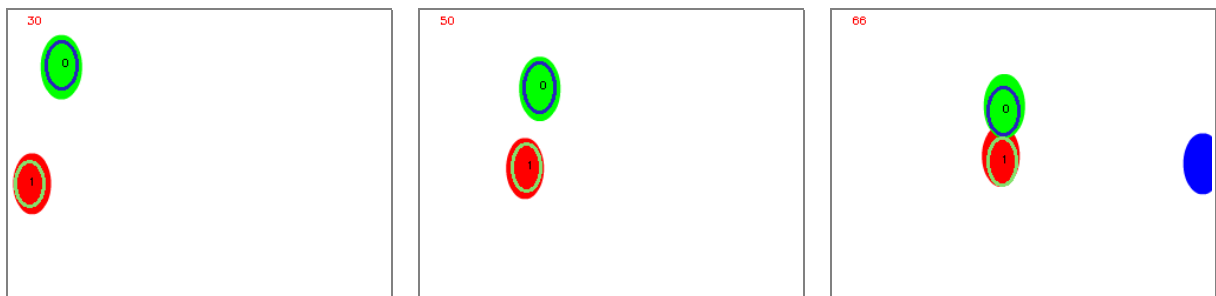


Illustration 6.7: Frames # 30, 50 and 66

The red and green ellipses are identified correctly as objects #0 and #1. The blue ellipse coming from the right has not been identified yet, because the moving area is still too small. Overlap has started between 0 and 1, but it has not been detected yet.

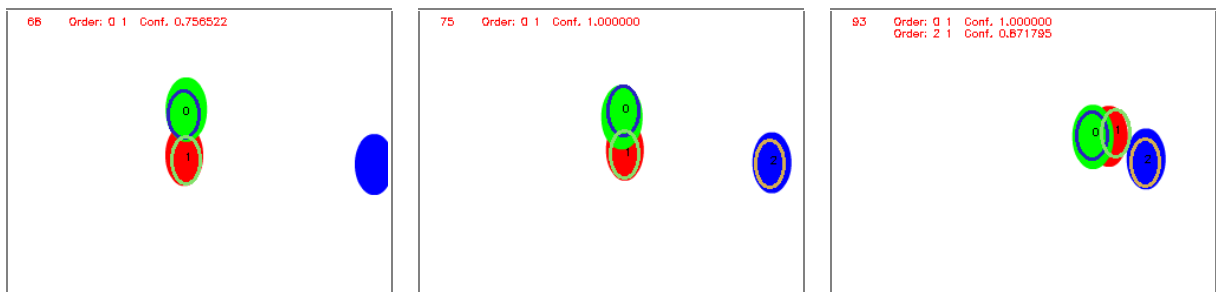


Illustration 6.8: Frames # 68, 75 and 93

The red and green ellipses start overlapping. The green one is correctly seen as being on top during the complete period they overlap. The blue ellipse is identified as object #2, and is on

its way to overlap with the other two ellipses. In frame 93, the overlap of object #2 with object #1 has started, and object #2 is seen as lying on top. Because the two ellipses are still too far apart, this is not a reliable result: there is no possibility yet that we can see which one is on top.

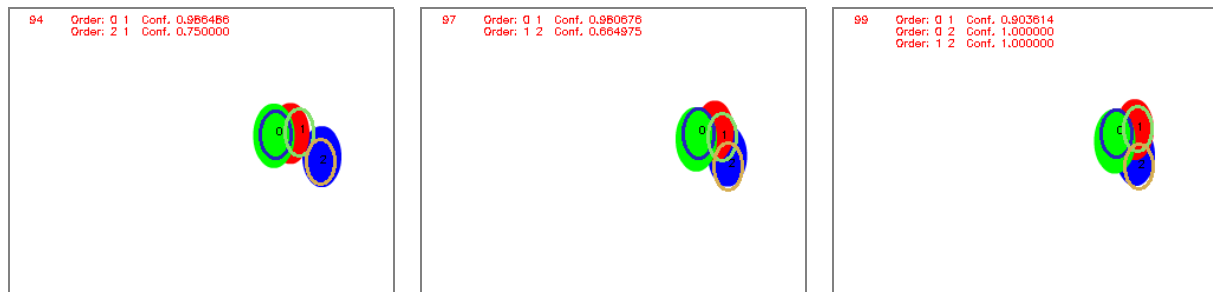


Illustration 6.9: Frames # 94, 97 and 99

Frame 94 shows the same problem as frame 93. We can see now that the red ellipse is on top, but the system still selects the blue ellipse, because the distance between the two is too large. In frame 97 the red ellipse is finally seen correctly as lying on top of the blue one. The overlap between #0 and #2 is not detected yet. In the last frame, number 99, all three ellipses overlap, and all combinations are identified correctly.

6.4.3 Conclusions

With this algorithm, 33 overlap situations are detected, and in 4 cases the wrong result is given. This gives a score of 88% (29 out of 33).

Errors in the detection of the top ellipses are only made, when the ellipses in question are too far apart to give a reliable result. Of course, because all ellipses have one uniform color, which is substantially different for every ellipse, detection is quite easy. When an overlapping area completely has the color of one the ellipses, the confidence that that ellipse is on top is 1.

6.5 Occlusion detection by comparing two tracked objects using camera sequence

In this test, the first algorithm for occlusion detection, as described in paragraph 4.4.1, will be tested with a camera sequence. The camera sequence 'vid3.avi' is used.

6.5.1 Goal

The goal of this test is to see whether the algorithm of 4.4.1 works, by looking at the returned results of the occlusion detection, and seeing if they are right or wrong. This time, it is harder to see whether the result is alright in every case, also because sometimes, there are more people in the frame than there are tracked objects.

6.5.2 Results



Illustration 6.10: Frames # 300, 390 and 455

In frame 300, no overlap has been detected yet. Frame 390 shows that there is a large overlap now. Person #1 (with the white shirt) is correctly identified as being on top (in the front). Frame 455, where there is again a large overlap, but now with the person in blue shirt on the front, shows again a correct detection.



Illustration 6.11: Frames # 520, 530 and 655

Frame 520 shows an occlusion of person #0 by person #2, this is correct. No other overlap is detected. In frame 530, overlap between #0 and #2, and between #1 and #2 is detected, and #2 should be in front. These occlusions do not really exist, but are found because the tracker for persons #0 and #1 has a deviation in the direction of #2. The last frame tested, 655, shows an overlap of persons #1 and #2, and #2 is detected as being on the front, again correctly.

6.5.3 Conclusions

In the first 600 frames, before the fourth person arrives, there are around 150 frames in which an overlap occurs, of which only 15 are really wrong. This gives a score of 90%.

In all real occlusions, the person in front has been detected. Only in the case of frame 530 occlusions are seen that do not exist, and this is a result of deviations of the tracker. Unfortunately, in this sequence, occlusions of three persons at the same time could be found only when a lot of people were in the frame. In these occasions, the tracker was completely lost.

6.6 Occlusion detection by comparing N tracked objects using animated sequence

In this test, the second algorithm for occlusion detection, as described in paragraph 4.4.2, will be tested with an animated sequence. The animated sequence 'triple_overlap.avi' is used.

6.6.1 Goal

Now, the second algorithm for occlusion detection will be tested with the same sequence as in 6.4. Again, the goal of the test is to see whether the algorithm (of 4.4.2) works, by looking at the returned results of the occlusion detection, and seeing if they are right or wrong. Because of the uniformly colored ellipses it is very easy to see which one is occluding the other, also in cases where more than two are overlapping.

6.6.2 Results

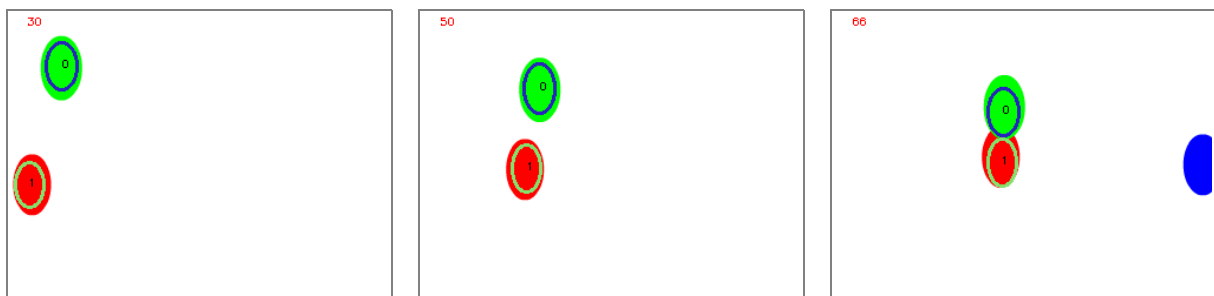


Illustration 6.12: Frames #30, 50 and 66

As with the first occlusion detection algorithm, in this case too, no occlusion is detected in the first three frames tested.

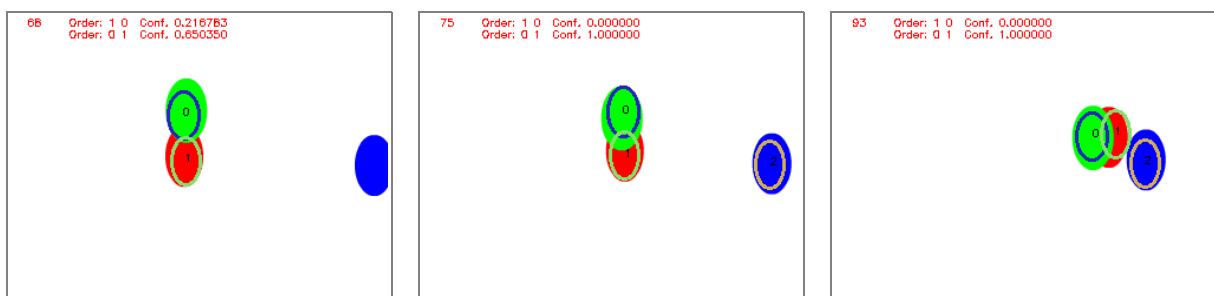


Illustration 6.13: Frames # 68, 75 and 93

In these three frames there is only overlap between ellipses #1 and #2, and in all occasions the top ellipse (#0) is detected correctly, in frame 75 and 93 with a confidence of 1. This 100% confidence can be explained, because the overlapping area is completely within one of the two ellipses, and so also of one uniform color.

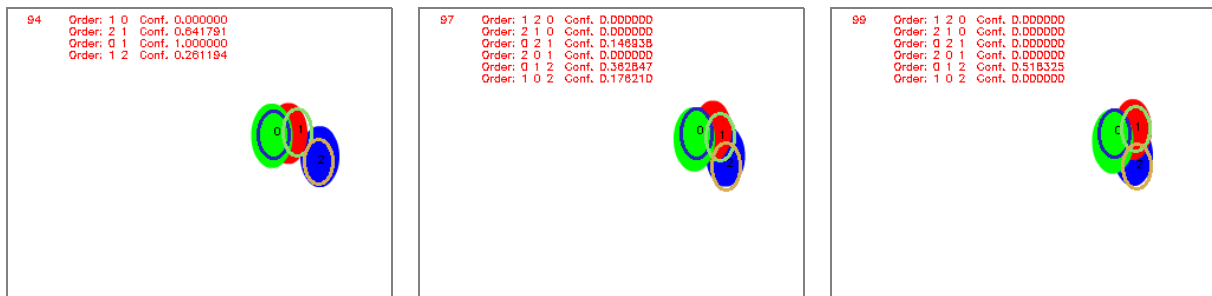


Illustration 6.14: Frames # 94, 97 and 99

Frame 94 shows an overlap of ellipses #0 and #1, and of #1 and #2. Ellipse #0 is again seen correctly as the top one, with a confidence of 1. #2 is also seen as the top ellipse, which is not true, but the classification is understandable, given that the overlap is still very small. In frame 97 and 99 all three ellipses overlap and in both occasions the order 0 1 2 (ellipse #0 on top, #1 underneath, and #2 at the bottom) has the highest confidence.

6.6.3 Conclusions

When the second algorithm is tried on an animated sequence, 34 overlap situations are found, and in 3 cases the wrong result is given. This gives a score of 91% (31 out of 34).

In all cases, the occlusions are detected correctly, only when the overlapping area is very small, the wrong ellipse is chosen. Also when all three ellipses were overlapping, there were no problems at all finding the right order in which they are 'stacked'.

6.7 Occlusion detection by comparing N tracked objects using camera sequence

In the last test, the second algorithm for occlusion detection, as described in paragraph 4.4.2, will be tested with a camera sequence. The camera sequence 'vid3.avi' is used.

6.7.1 Goal

The goal of this test is to see whether the second algorithm, of paragraph 4.4.2, works, by looking at the returned results of the occlusion detection, and seeing if they are right or wrong. The same sequence as in 6.5 is used. Again, it is harder now to see whether the result is alright in every case, also because sometimes, there are more people in the frame than there are tracked objects.

6.7.2 Results



Illustration 6.15: Frames # 300, 390 and 455

Frame 390 shows that overlap of person #0 by person #1 is correctly detected. In frame 455, the occlusion of person #1 by person #0 is also seen perfectly. #2 has no overlap yet, as in the previous algorithm.



Illustration 6.16: Frames # 520, 530 and 655

No overlap is detected in frame 520 for person #1. The occlusion of person #0 by #2 is correctly identified, however with a small difference in confidence. In frame 530, overlap between #0 and #2, and between #1 and #2, is seen. In both occlusions, person #2 is seen on the front. This is correct, if the deviation of ellipses #0 and #1 in the direction of #2 is taken into account. The last frame tested, 655, shows an overlap of persons #1 and #2, and #2 is detected as being on the front, again correctly.

6.7.3 Conclusions

The second algorithm gives more or less the same results as the first: in the first 600 frames, before the fourth person arrives, there are around 150 frames in which an overlap occurs, of which only 15 are really wrong. This gives a score of 90%.

So, almost all occlusions are detected correctly, only because of some tracker mistakes occlusions are shown that do not really exist. Still, also in those cases, the most probable top object chosen seems to be the best choice when looking at the color.

7 Conclusions and Recommendations

First, this chapter revisits the problem definition and draw conclusions based on the test results. After that, suggestions are given for further research.

7.1 Conclusions

In the problem definition stated at the beginning of this document, in the introduction, two problems were identified. First, how to improve existing trackers so that they are more useful for tracking multiple people simultaneously? And secondly, how to tell from tracked people that are occluding each other who is in front and who is in the back?

7.1.1 Improvement in tracking multiple people

The original tracking algorithm on which this work is based, was not very good at tracking multiple people at the same time. A modification to this original work was designed and implemented.

As seen in the test results, the changes made to the tracking algorithm, have led to improvements in some cases. It was hypothesized that these improvements occurred mostly in the case where movement was happening according to a fixed dynamic model.

Because in real life situations it is not often the case, that persons are moving constantly according to a simple model, there will not be much instances in which a significant increase in performance can be seen right now. Unfortunately, this means that algorithms built on top of the tracking algorithm, like the occlusion detection, cannot benefit from any performance improvement either.

7.1.2 Occlusion Detection

The second problem was that there was no way to detect if people are occluding each other, and also, to see who is occluding who. Two different methods to solve this problem were designed, implemented, and tested.

The results have shown that the difference between these methods is not very large. The problem was, that the testing of multiple overlapping objects in a camera sequence was not really possible. The difference between the two algorithms should really show in situations where more than two objects are overlapping. The tracker performance should be still better before these situations can be tested reliably.

However, given that the performance of the multiple person tracker was below expectations, the results of the occlusion detection algorithms are still by all means very promising. In almost all cases the choices made by the algorithm seem to be the most logical in the circumstances.

7.2 Recommendations

Of course the two questions defined had to do a lot with each other. One possible other way of improving the tracking of multiple people, would be using the result of the occlusion detection result as feedback in the tracking itself. This way, the occlusion information is not lost, but used for updating the tracker. However, the question how this should happen precisely still remains. One possibility may seem to track the occlusions themselves, as described in 3.5.1. But in fact, this should not make any difference, as there is no feedback defined from the occlusion tracker to the person tracker.

Another promising direction of further research would be the use of auditory information. When people are talking or making noise, things that could happen very well in groups, the could be tracked much more precisely when you use this information. Also the detection of who is in front and who in the back would be easier.

If there would be need for improving just the occlusion detection algorithm, one option that may give better results, is to split all the histograms that have to be compared horizontally in two or more parts. The same was done with the Condensation tracker, because of the way people dress. The only hard thing about this is that the location of overlapping people with respect to each other has to be taken into account, because the top part of one person can overlap the bottom part of another.

There is also still room for improvement in the real-time performance. The using of the algorithms in real-time was not a demand for this research, so a lot may be gained when effort is spent at optimizing the algorithms.

The problem of tracking multiple people still remains a very hard problem, for which no easy solutions exist. With more processing power, of course more expensive computations can be done. But it seems that there always will remain some exceptions, that we humans can detect easily, but with which computers have severe problems.

Bibliography

- [1] Comaniciu, D., Ramesh, V. and Meer, P. “Real-Time Tracking of Non-Rigid Objects using Mean Shift.” *IEEE Conf. on Computer Vision and Pattern Rec.*, vol. 2, pp. 142-149, 2000
- [2] Fexa, A. “Separation of individual persons in a crowd from a videosequence.”, 2004
- [3] Gavrilă, D.M. “The Visual Analysis of Human Movement: A Survey.” *Computer Vision and Image Understanding*, 73(1):82-98, 1999
- [4] Haritaoglu, I., Harwood, D. and Davis, L.S. “W4: Real-Time Surveillance of People and Their Activities.” *IEEE Trans. Pattern Analysis and Machine Intell.*, 22(8):809-830, 2000
- [5] Hu, M., Hu, W. and Tan, T. “Tracking People through Occlusions.” *IEEE 17th Int. Conference on Pattern Recognition*, vol. 2, pp. 724-727, 2004
- [6] Intel Corporation. “Open Source Computer Vision Library.” <http://www.intel.com/technology/computing/opencv/>
- [7] Isard, M. and Blake, A. “Condensation - Conditional Density Propagation for Visual Tracking.” *International Journal of Computer Vision*, 29(1):5-28, 1998
- [8] Khan, Z., Balch, T. and Dellaert, F. “An MCMC-Based Particle Filter for Tracking Multiple Interacting Targets.” *8th European Conference on Computer Vision*, vol. 4, pp. 279-290, 2004
- [9] McKenna, S.J. et al. “Tracking Groups of People.” *Computer Vision and Image Understanding*, 80(1):42-56, 2000
- [10] Microsoft Corporation. “DirectX Developer Center.” <http://msdn.microsoft.com/directx/>
- [11] Pentland, A. “Looking at People: Sensing for Ubiquitous and Wearable Computing.” *IEEE Trans. Pattern Analysis and Machine Intell.*, 22(1):107-119, 2000
- [12] Stauffer, C. and Grimson, W.E.L. “Adaptive background mixture models for real-time tracking.” *IEEE Conf. on Computer Vision and Pattern Rec.*, vol. 2, pp. 246-252, 1999
- [13] Vermaak, J., Doucet, A. and Pérez, P. “Maintaining Multi-Modality through Mixture Tracking.” *IEEE 9th Int. Conference on Computer Vision*, vol. 2, pp. 1110-1116, 2003

- [14] Wren, C., Azarbayejani, A., Darrell, T. and Pentland, A. "Pfinder: Real-Time Tracking of the Human Body." *IEEE Trans. Pattern Analysis and Machine Intell.*, 19(7):780-785, 1997