# Intelligent Methods for Automated Video Surveillance

Delft University of Technology

Faculty of Electrical Engineering,
Mathematics & Computer Science

Alexander Keur (1055380)
26th March 2009

**Author**
Alexander Keur
**Title**
Intelligent Methods for Automated Video Surveillance
**MSc presentation**
February 2009

## Abstract

At the Man Machine Interaction research group at the Delft University of Technology research is being done on the subject of aggression detection in trains. The goal of this project is to research different aspects of train surveillance, including video surveillance, but also audio surveillance storyboard based modeling.
This thesis discusses the current state of the art methods and techniques that or being applied, or could be applied to the task of automated video surveillance.
This work discusses the application to the video surveillance problem of several methods, most notably motion detection, face tracking, face recognition and facial expression analysis.

# Acknowledgements

I would like to thank my supervisor, Leon Rothkrantz, for his much needed support, advice and guidance during my thesis work. I would like to thank Zhenke Yang for his cooperation and involvement in this project.

Alexander Keur

Delft, The Netherlands
26th March 2009

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Surveillance in public places is currently widely used to monitor locations and the behavior of the people those areas. Closed Circuit Television (CCTV) systems around the world are used to monitor the safety of people in public spaces 24 hours a day. Since events like the terrorist attack in Madrid and London there has been a further increasing need for video network systems to guarantee the safety of people in public areas. But also events like football games, music concerts and large venues like shopping malls where large amounts of people gather, have a need for video surveillance systems. Another field of application is to protect people as well as property against aggression, physical attacks and crimes like robbery and vandalism. However, the greater the number of cameras, the greater the number of operators and supervisors needed to monitor the video streams.

In public transport surveillance has proven to be a useful tool in detecting a preventing potentially violent situations. This research focusses on the application of intelligent video surveillance methods in public transport, most notably inside the train compartment. Aggression in trains can lead to great discomfort of other passengers and working personnel, and even go as far as disrupting the service schedule, causing physical harm to people and the damaging of train furniture and material. Aggression against the train conductor and other passengers leads to great distress on the bystanders, and may eventually lead to physical aggression.

According to a 2004 report on social safety by the Dutch Ministry of Transport, Public Works and Water Management (Ministerie van Verkeer en Waterstaat) [9], 20% of all train travelers in 2004 have been the victim of an incident. The highest level in four years. The report also shows an increase in serious incidents such as abuse and theft. Reducing the number of incidents will increase the (perceived) safety of the passengers. A passenger not feeling safe is more likely to choose other transportation methods such as private cars.

An efficient surveillance system could lead to improved incident handling and prevention, resulting in a safer public transport. Video surveillance allows a single supervisor to monitor several video feeds of multiple areas at the same time, as opposed to a local surveillant on the scene who can only oversee his immediate surroundings. This allows one single supervisor to cover a much larger area. In the wake of this development we are seeing efforts to further automate the task of surveillance. Computer systems are already being used to record and preprocess video and audio data, and we are now seeing dedicated video surveillance software entering the field. The use of computer software can be simple as in merely detecting change in a scene, but more complex applications and methods are currently being researched.

Currently, video surveillance systems are mostly passive. They require a human operator to monitor the video feeds on a screen, and to alert security crews when their assistance is required in case of emergency. Automated video surveillance as suggested in this work is currently not being used at the Nederlandse Spoorwegen (NS). The number of available cameras for monitoring is currently very limited. Few trains, and only the larger train stations in the Netherlands are equipped with video cameras. As more cameras are installed in the future, a system like the one proposed will become necessary.

A fully automated surveillance system is currently not commercially available. Some software packages do exist, but they mostly record video streams and provide little further functionality. Behavior detection and motion detection, as well as human tracking methods are a widely researched topic. A combination of these methods could be used to provide an automated system capable of classifying human behavior, and in case of this project specifically, detecting aggression. A multimodal surveillance system would use audio and video data from the feed, and analyze this to determine the behavior that is present in the currently observed scene. To analyze a situation, the scene must first be interpreted. Separate systems are proposed to analyze both the video and the audio stream. This work will focus on methods of analyzing and processing the video data. The goal is to ultimately extract as much as data as possible from the imagery, and to find a way to interpret this data in a meaningful manner given the context.

Eventually an automated surveillance system should aid the train surveillant in detecting certain types of unwanted behavior, and make it possible to intervene in aggressive situations more quickly. The goal is to eventually have a system that can quickly and accurately monitor a large and very complex area for human behaviors, and when needed report observed activities to a surveillant, or even deploy assistance if required. Furthermore, the presence of a functioning surveillance system has been shown to have preventive effects against aggression and vandalism.

The goal of this work is to research the field of video surveillance, and investigate what the common problems are and how they can be solved. For this we research the current literature to find the most common problems and how they are currently being solved. We are particulary interested in the integration of computer vision techniques into one system geared towards crowd surveillance. Using available methods we aim to develop a model for an automated surveillance system. This work focusses on video surveillance, but the eventually the goal is to design a system that integrates different modalities, especially audio. We can thus define the goal of this project as follows:

**Project Goal**

*To design a system composed of a network of video cameras, intelligent computer* *vision techniques, for automated video surveillance and crowd monitoring.*

We use video recorded earlier using actors in a train compartment provided for this research by the NS, as well as some data recorded at the TU Delft and available face detection training data on the internet. This data will then be used to test different methods and our own implementations.

Although the main focus of this work is researching the theory and designing a system, several implementations will be offered to demonstrate the possibilities of current techniques and their application to the automation of video surveillance. Parts of this research were used for the publishing of a paper for the Euromedia 2008 conference [21].

## 1.1 Organizational Description

The outline of this work is as follows: The first chapter gives an introduction to the work that will be presented in this thesis, as well as the general social and academic relevance of this research. The goal of this work is also stated in this chapter. In **Chapter 2** we present an overview of the entire problem in which we describe the setting for the proposed system, and various key elements for our work, such as the features we wish to detect, the environment in which the system would operate, and how these factors can influence each other. In **Chapter 3** we discuss the state of the art by means of a discussion of related works. Many computer vision as well as mathematical methods used later on in this work are based on existing techniques, some directly related to video surveillance and face detection, others more general, such as point tracking and image processing. **Chapter 4** presents a framework for an automated surveillance system, and the subsequent design for each of the modules. This work is to be part of a greater framework in which aggression detection is to be achieved by acquiring and combining gathered environment data from different modalities, including video. The methods used include motion detection and classification, face detection, and tracking. This chapter also describes the design of different methods to detect and classify behavior, based on the latter mentioned methods. We also describe the implementation provided for some of the modules described in this chapter. In **Chapter 5** we describe the experiments performed for this work including the recording of the data, and the performance of the modules we implemented. Afterwards we evaluate these results. **Chapter 6** presents the conclusions of this work, as well as recommendations for future work in this area. Finally, in the **Appendix** we include a paper partly based on the research in this thesis that was presented at the 2008 Euromedia conference in Porto.

# Chapter 2

# Problem Overview

The main goal of this project is to develop a video surveillance system that can detect and analyze types of human behavior in trains. This thesis will focus on aggression and violence detection in the video data, specifically in train compartments.

## 2.1 Environment

The environment where the system is to be used are the compartments in the trains of the Nederlandse Spoorwegen (NS). These compartments consist of a 3 meter wide cabin, with a length of approximately 15 meters. These compartments form an enclosed space, that can be entered through doors at each end of the compartment. An example is shown in figure 2.1.

Our system is designed to be used in a real world train compartments. The system will therefore be subjected to situations different from the common lab environment. This means that a lot of environment variables such as lighting and people behavior will be out of our control.

The compartments contain rows of four seats, with a corridor in the middle. People enter the train through the entrances on both ends, and walk through the corridor. People can therefore be walking through the corridor in opposite directions. People will either find a place to sit, or walk along through the corridor and exit the

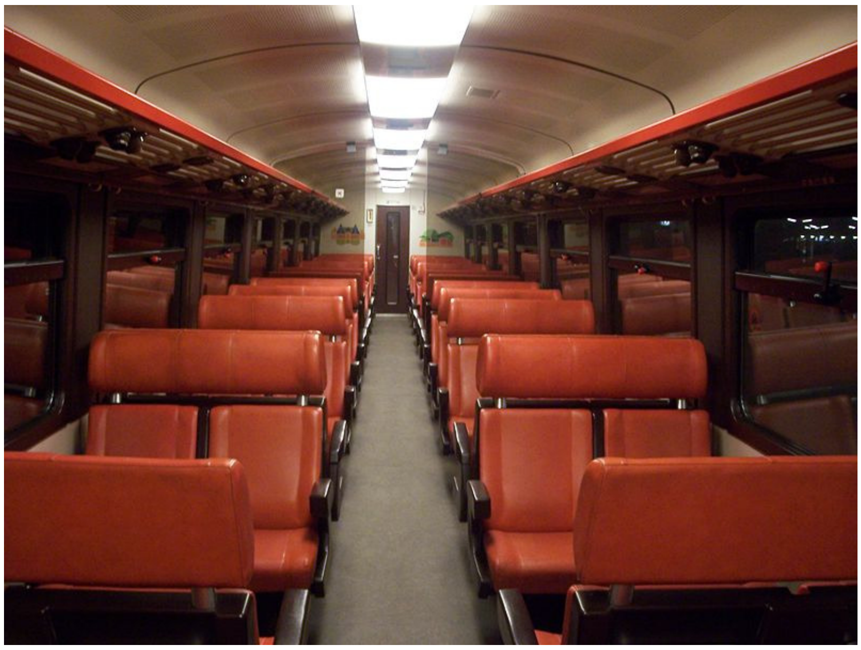


Figure 2.1: A typical train compartment

compartment on the other end. If the train is very crowded, people can use the corridors for standing. During busy hours, the corridor can easily get congested during stops when many people are entering and exiting the train. This leads to passengers commonly occluding other passengers. During travel, there is less movement in the compartment, usually only the train conductor checking for tickets, and passengers moving to and from the toilet. Passengers will have unobstructed access to the corridor, and will be able to walk through the compartment at higher speed compared to busy times like rush hour. At night it is not uncommon to observe complete unoccupied compartments since usage at that times is low.

When the train is moving, most passengers will be expected to be sitting in their seats, either talking to eachother, looking out the windows, reading, or sleeping. In case we still observe movement in a moving train, this is sometimes caused by people moving from one seat to another, people running in the corridor, or people pushing eachother to either get in or out just after boarding. All of these cases can be seen as a sign of possible upcoming aggression.

Both sides of the train are fitted with large windows The windows are divided into a small upper part and a larger lower part. Only the top part of the window can be opened or closed.These windows let in daylight, which leads to varying lighting conditions in the train, due to outside weather, or when the train enters a tunnel. Buildings casting shadows on the train also can cause very rapidly changing lighting conditions in the train. During nighttime, fluorescent lamps illuminate the cabins. Opening windows also causes wind noise when the train is moving, but this is not directly relevant to our research. The ceiling is approximately 2.10 meters high in Dutch trains. The width of the cabin is 3 meters.

The train compartments are equipped with digital cameras at the locations shown as in figure 2.3, which will be providing the video streams used in this project. The cameras are located on the ceiling of the train compartments, four cameras in each. Furthermore, there are two cameras located at both entrances of the train, facing outwards. These cameras capture the entering passengers. The cameras are tilted slightly downwards at an angle of approximately 30 degrees, focusing on eye-height of most passengers in the direct vicinity of the camera, as well as the seats near by. The cameras are currently static, that is, they are fixed at their position and in their orientation. Panning cameras however could some day be installed, providing each individual camera with a larger field of view. Similarly, other functions could be added, like the ability to zoom in to individual faces of objects.

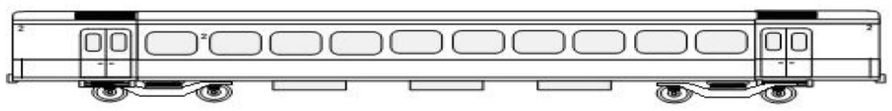


Figure 2.2: Side view of a train

The cameras used have a resolution of 640x256 pixels, or about 1.6 megapixels. The video is provided in an unusual $2.5 : 1$ aspect ratio for PAL video compatibility, meaning that some scaling and resizing has to be performed in order to make the video suitable for digital processing. The resolution is adequate for a human observer to make out individuals in the video feed, but recognizing faces can be

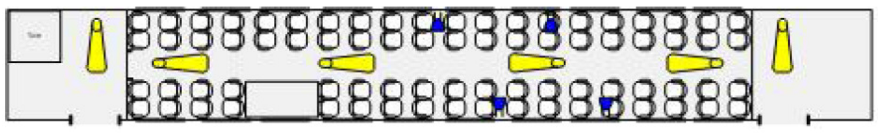


Figure 2.3: Seating plan for trains including camera positions

difficult when viewed on a small screen. Furthermore, the resolution of these cameras is too low for observer to recognize faces that are not in the close vicinity of the camera.

Apart from these cameras, we installed several webcams, filming at a slightly flatter angle, to capture about one third of the train in a single scene. These webcams record video at a resolution of 352x288 pixels.

## 2.2 Behavior

To detect and classify human behavior in a scene we must first determine what types of behavior we wish to distinguish. Different types of behavior will have different characteristics, which may or may not be suitable for automated detection and analysis using video data. Our goal is to determine which specific features of different behavior exist, and how these could be relevant to our research topic.

The behavior we wish to detect for this research is mostly aggressive behavior. Aggression is any kind of behavior that is intended to cause harm or pain. For our case we consider two basic types of aggression, verbal and physical. Verbal aggression can for example be aimed at train personnel checking for tickets, but loud conversation is also a common nuisance in the train. Physical aggression is anything that involves fighting with other passengers, but also things like destructive behavior and damaging of property. For both types of aggression, there are several scenarios that are most likely to occur in the train environment.

Because we want to focus on visual behavior aggression detection, we are mostly interested in physical aggression. This also seems the most suited for detection by video surveillance. There are for more visual characteristics to detect than in other types of aggression like verbal abuse, frustration, or other more passive forms. We will concentrate on features like wild sudden movements by passengers, erratic behavior, and other unexpected behavior such as running or falling.

### 2.2.1 Aggression in the Train

The Dutch Railways (NS) spend a large amount of money each year for prevention of aggression in their trains. The main goal is to keep their own personnel and passenger safe, but also to prevent damage to the train equipment and furniture. Furthermore, high aggression will undoubtedly scare many passengers away, thus reducing profit from ticket sales. The Dutch Railways define different categories of incidents, which include aggression. This list is shown in table 2.1.

Table 2.1: Incident categories used by the Dutch Railways (NS)

| Category | Description |
|---|---|
| A | Suspicious behavior |
| B | Robbery and theft |
| C | Violence |
| D | Serious public inconveniences |
| E | Small public inconveniences |
| F | Vandalism |
| G | Accident |
| H | Fire |

The following are examples of passenger behavior and some of its features, and the aggressive behavior that we could expect to observe.

• Normal behavior - Passenger

A passenger is expected to enter a train compartment at a normal walking speed, traveling in a straight line towards an empty seat, and perhaps standing still for a few seconds before moving towards the seats and sitting down. When not sitting down, a passenger is expected to move through the corridor in one movement, without stopping. When the train is very full however, passengers can be expected to stand still in the corridor due to lack of sufficient seats, otherwise this behavior is not expected.

• Normal behavior - Train Conductor

A train conductor when checking tickets will show a slightly different behavior, namely walking through the corridor, and stopping briefly at each row of seats to check tickets. This delay should not often take more than a few seconds, when taking longer it can be expected that a passenger can not immediately provide his ticket, and/or the conductor is issuing a fine to said passenger. Since this can sometimes lead to aggressive behavior, being able to detect when a) a passenger is identified as the conductor and b) the conductor is experiencing such a delay is desirable.

• Speed of movement

When a person is moving faster than normally expected in a train compartment, this can often be linked to aggressive behavior. For example, running towards another passenger (charging), or running away from the train personnel when they are coming to check for tickets. Smaller fast movements can also be linked to aggressive behavior. These can include actions like swinging an arm (punching) or a person being pushed. Fast movements not directly linked to a person may also constitute aggression, such as the throwing of objects within the cabin, and the slamming of doors. Also, since we do not expect many sudden movements in normal behavior, any such sudden fast movement could be of interest.

• Non-standard/non-scenario behavior

In the train compartment there are areas designated for sitting (the seats), and for walking (the middle corridor). When we observed people not moving in the corridor, or observe a lot of movement in the seats area, this could be a situation that would require attention. The only times when we would expect movement in the seating area, is when the train has just stopped at a station, and people are taking or leaving their seats. At this point we do however expect a lot of movement in the corridor. Contrary, when the train is moving, we expect the opposite behavior. Only few people should be walking through the compartment at this point. At this time we expect to see the occasional passenger visiting the restroom, and the conductor coming into the compartment to check the tickets. Opposed to these forms of non-standard movement, we might also encounter non-standard non-movement. An example of such behavior could be passengers resting their feet on other seats, lying down in the seats or in the corridor.

• Erratic behavior

When a passenger is taking an unusually long time to find a seat, switching seats often, or moving at non-constant speed in a non-constant direction this could have several reasons we are interested in. For instance a passenger could be drunk or under the influence of other substances, which could potentially lead to a dangerous situation. Any such non-standard behavior could be detected by defining standard behavior and simply determining when an observed action differs to much from it.

• Poses

People who are physically aggressive, will often assume different poses. They may use their fists to punch as in a fighting stance, or for example be kicking train furniture or even other passengers (Figure 2.4). A person being the victim of aggression will often be using their arms and hands to shield their heads, or use their knees when on the ground to protect their bodies. More subtle features of aggressive poses include bending the forehead slightly forward, sometimes towards the subject of the aggression, or the raising of the arms.

• Personal area invasion

According to social psychological studies, human beings have a personal area surrounding them (the size of which is culturally dependent) and do not like intrusion by other people in this area, especially strangers. In case one person is moving close to another person, within the other person's personal area, and this other person then moves away, we might have a case of invasion here. This is often the case when people are trying to intimidate others, or more explicitly, when people are fighting.

• Facial expressions

Facial expression convey a great deal of information about the emotions being experienced by a person. In dangerous situation we expect to see people express

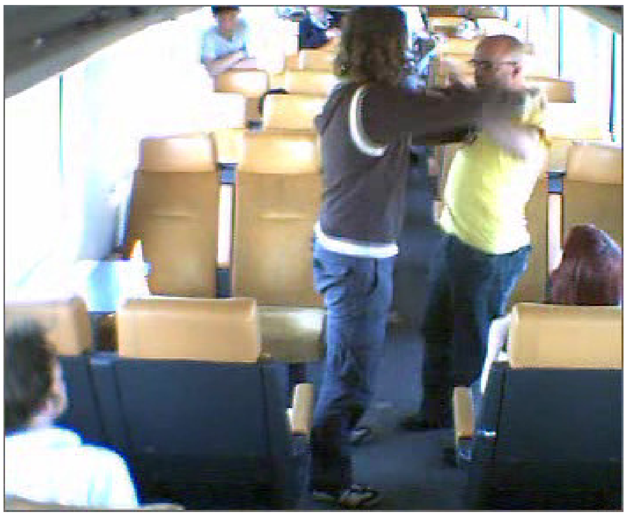


Figure 2.4: Fighting passengers in the train

emotions like fear and anger through their faces. Examples of different expressions are shown in figure 2.5.

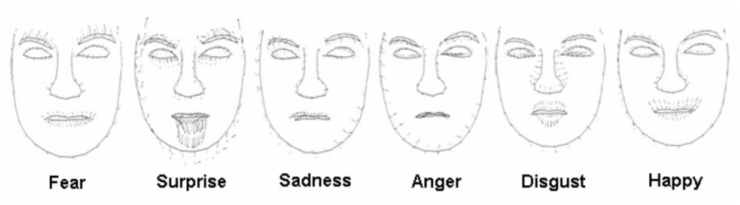


Figure 2.5: Facial expressions templates used by Datcu and Rothkrantz in [7]

Tables 2.2 and 2.3 summarize several types of reprectively static and dynamic features, related to aggression or other unwanted behavior, that might be detected using currently available methods, and could therefore apply to our system. Static features are those features that can be detected from single frames. Dynamic features are detected in a series of frames, and involve behavior exhibited and detected over a period of time.

Table 2.2: Static features related to aggression

| Behavior | Description |
|---|---|
| Unusual locations for faces or body parts | Faces very close to the floor, or in the lockers. Shoes on the furniture. |
| Unexpected objects | Knives, sticks, cigarettes & smoke, bottles |
| Invasion of personal space | Persons detected very close to eachother, passengers touching or pushing one another. |
| Body poses | Offensive or aggressive postures, raised arms |
| Smoking | Persons smoking cigarettes when not allowed |
| Facial expression | Angry or aggressive expressions |
| Abandoned luggage | Suitcases, backpacks |

Table 2.3: Dynamic features related to aggression

| Behavior | Description |
|---|---|
| General motion of persons | The amount of motion energy in a scene is different from the expected value given the current status of the train |
| Running | Persons running through the train |
| Fighting | Fights among passengers, pushing, kicking, jumping |
| Vandalism | Damaging of train seats or windows, graffiti |
| Throwing objects | People throwing with objects in the train compartment |

# Chapter 3

# Related Work

## 3.1 Overview

To detect the situations previously described, there are currently a number of computer vision techniques available, especially tracking, face detection and motion interpretation. Different methods have different applications to our project. We will discuss the possibilities of each, and the feasibility of their application to our project.

## 3.2 Motion Detection

Being able to detect motion in a scene is the first and most basic step towards understanding behavior, since most of the behavior we wish to detect, especially aggression, is associated with some kind of body movements. Motion detection aims at segmenting regions of an image corresponding to moving objects over frames to the rest of the image. Most techniques try to find a so-called 'blob' in one frame, and associate it to a similar looking blob in the next frames. The easiest blobs to detect are uniformly colored regions. If the camera is fixed, blobs in the frame will be changing position, and these blobs can the be assumed to be moving objects. Most of the other operations we wish to perform, such as person tracking and behavior interpretation are highly dependent on motion detection. Motion detection algorithms usually involve environment modeling and motion segmentation.

### 3.2.1 Environment Modeling

Environment modeling is very important for motion detection since it provides a description of the scene which can help in interpreting the observed motion data. Especially background modeling is important because it can greatly reduce the cost of computation. One of the challenges is being able to model the background pixels under varying lighting conditions. Many methods exist, some including Gaussian pixel models, others using Kalman Filters to reduce the variance in illumination [10]. A more simplistic, but static approach is using a previously acquired image of the scene without any objects or persons in the scene, possibly under varying lighting conditions, so that it can later be used for background subtraction. This method can be applied to static cameras, but is generally unsuited for moving cameras.

2D environment models are usually preferred over 3D models for their simplicity, and the fact that most modeling can be done using only the data visible in the image plane from the camera. Usage of 3D environment models is being researched but currently mostly limited to indoor scenes due to the high complexity of outdoor scenes [22].

### 3.2.2 Motion Segmentation

Motion segmentation aims to segment the moving parts of an image from the background of the image, thus to detect moving objects such as persons or vehicles. Some currently conventional motion segmentation methods used today are outlined below.

1. **Background Subtraction**
   Background subtraction is a very simple, and popular method for motion segmentation. It simply subtracts a previously acquired reference background image from the currently observed image, the reasoning being that any pixels that have significantly changed in value must belong to the foreground.

It is useful for scenes with a relatively static background. Due to its simple nature it is extremely sensitive to changes in the environment, such as changing lighting conditions. To reduce the influence of these changes, a good background model is extremely important. An example of background subtraction is shown in figure 3.1. In this case we could take advantage of our access to an empty scene, and having a fixed camera. Although the lighting conditions are slightly different between the background frame and the video frame, it is still possible to perform the algorithm with satisfactory results in this case. In the example we also see that the bag in the lower right seat is not present in the empty background image, and thus we also detect the bag in this example.

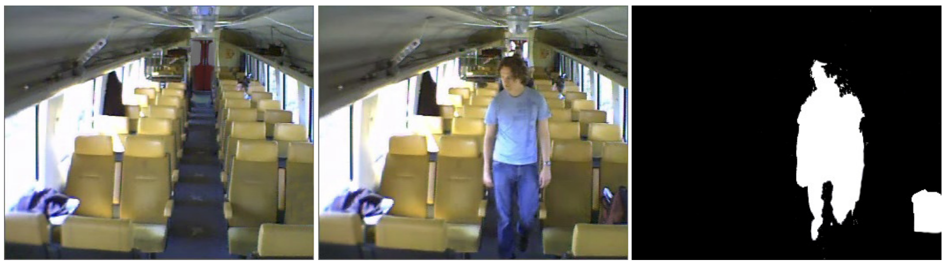


Figure 3.1: Background subtraction applied using the first image as the static background, and the second image as the input. The resulting mask is shown in the third frame.

2. **Temporal Differencing**

   Temporal differencing uses the difference in pixel values over several frames of an image sequence to determine moving regions. It is therefore more adaptive to dynamic environments than background subtraction. A downside of this method is that it can ignore pixels that are part of a moving object, but remain at the same color value for several frames, effectively producing 'holes' in the detected object. For example, a large, plain-colored object will produce the same color of pixels at a certain coordinate for a certain number of frames. If this period is longer than the time the algorithm looks back in the image sequence, these pixels will appear unchanged, and therefore be considered background instead of foreground. This characteristic makes it less suitable for detection large, uniformly colored objects, that exhibit fairly linear motion. Another drawback is that the difference between two frames will include the pixels from the posterior frame, that is, the area in the image where the movement originated from. This is shown in figure 3.2 where we can clearly see the pixels being masked in the top area of the image where the moving person was in the first frame. When compared to background subtraction as shown in figure 3.1, we notice that the masked area found in the third frame in this case indeed is slightly larger, because it also includes the actor in the original frame having moved, and thus changing the pixel values in that area. Especially when the time difference in frames is larger, this overlapping will also be larger. Also, we notice that since the bag in the lower right corner is present in both frames, there is no difference between these pixels, and therefore the bag is not detected, while it is detected using background subtraction in figure 3.1. This is a drawback for the temporal differencing method, it cannot detect changes in a scene if they occurred before the last instance we use to compute the frame differences.

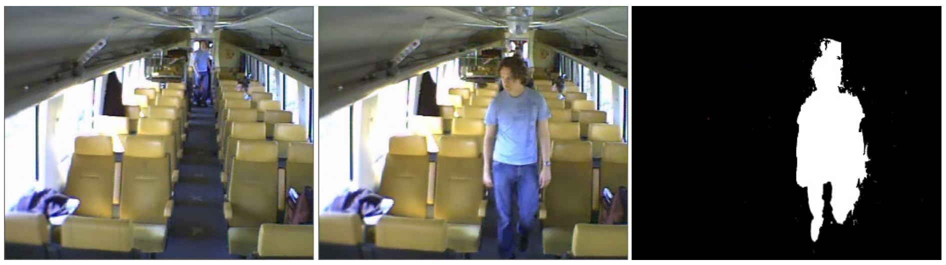


Figure 3.2: Temporal differencing applied to two video frames, with the resulting selected pixels shown as a mask in the third image

## 3.3 Face Detection

Face detection can be regarded as a special case of object-class detection. Object-class detection aims to detect all objects in a scene belonging to a certain class, such as vehicles, cars, but also upper bodies or pedestrians. Most face detection methods are also face localization methods, because they also determine the location, and size of a face in an image. While most face detection methods try to detect frontal views of faces, newer algorithms attempt to detect faces from multiple angles, or multi-view face detection [23]. A wide variety of techniques exist, ranging from simple edge-based algorithms, to complex high-level approaches using pattern recognition methods. We will discuss the two most common approaches to face detection: image based, and feature based.

### 3.3.1 Skin Pixel Based Face Detection

Detecting skin in an image can effectively enable us to also detect humans. Skin detection is mainly a tasks of color segmentation, which divides an image into sections which can be used for face detection applications. An example of this method is presented by Albiol et. al. in [3, 2]. This method detects faces by first detecting skin pixels, and then applying a segmentation algorithm to find skin regions. Region grouping is then applied to find region that most likely represent a face.

Skin pixel have very specific correlation between color values which can be exploited by a classification algorithm. This correlation is present for all skin colors. The relation between color values in different color spaces for skin tones is highly correlated [16], and falls within a narrow range. It is therefore quite trivial to simply segment the image into skin and non-skin pixels. Most methods simply use a color map to classify pixels.

In the next stage, the skin pixels are clustered to obtain regions. In the case of [3], a watershed segmentation algorithm is used to find clusters, after which the most face-like blobs are selected as the faces.

For this work we designed and implemented a pixel based human detector based on the works of [3, 2, 16], and tested its performance in our live train setting.

### 3.3.2 Viola & Jones

Paul Viola and Michael Jones introduced a new approach for visual object detection using the principle of a boosted cascade of classifiers [17]. It can be trained for face detection, and is capable of processing images extremely rapidly while achieving high detection rates. They used the Adaboost machine learning algorithm and claim to have achieved a 15 times speedup over the original Rowley [1] implementation for their detector. To understand the Viola & Jones detector, the concept of *boosting* needs to be explained first.

Randomly answering a yes or no question with an evenly distributed answer space will yield the correct answer 50% of the time in the long run. If a method can improve this score by a very small amount, it is called a *weak classifier*. It is possible to generate weak classifiers for a great number of tasks in an automated manner by enumerating a large set of generated data on a basis of very simple rules, and then evaluating their performance on a set of samples. A heuristic that can improve the detection rate by a larger amount is called a *strong classifier*. By 'boosting', we aim to combine several of these simple, weak classifiers, and create a strong classifier. Adaboost is a well known method to combine weak classifiers and create strong classifiers [17].

The weak classifiers in Viola & Jones are based on three different kinds of features. The two-rectangle feature is the difference between the sum of the values in two adjacent rectangular windows. The three-rectangle feature takes three adjacent rectangles, and computes the difference between the sum of the pixels in the extreme rectangles, and the sum of the pixels in the middle rectangle. A four-rectangle feature considers a 2 by 2 set of rectangles, and computes the difference between the sum of the pixels of the diagonally opposed rectangles. An example of these features is shown in figure 3.3. It is shown how for example, a three-rectangle classifier can return recognizable results for face sections like the nose, where the differences in pixel sums over the square are very characteristic. The minimum size of a feature is roughly comparable to the size of a face in an image, so a 16x16 section of an image can already contain hundreds of thousands of features, since a 12x12 pixel or even smaller detector is swept over each pixel. This is where the Adaboost algorithm comes in, and selects those weak classifiers to limit the selection to a few hundred weak classifiers, that will still yield good enough results.

This obviously greatly increases the speed of the algorithm.

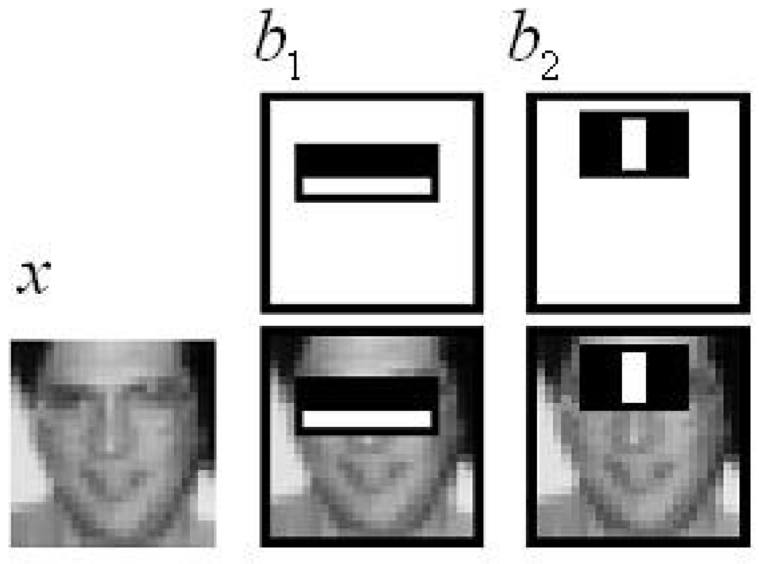


Figure 3.3: Example of rectangle features in Viola & Jones

Computing the rectangular features is a straightforward operation. The algorithm then introduces the *integral image*. The integral image at a location $(x, y)$ is defined as the sum of the pixel values above and to the left of $(x, y)$. It is therefore like an integral function over the entire image. The integral image $ii$ is defined as a function of the original image $i$ in equation 3.1. By using this representation, we can greatly increase the efficiency in calculating rectangular sums, since we only need to compute the difference in the total sum for the two corners of the rectangle. This is a contribution by Viola & Jones to the original algorithm, which greatly reduces the computational complexity, since it effectively removes the needs to calculate large sums of pixels for each and every classifier, every single time. To determine whether a sample contains a face, the sum of weighted classifier scorer is taken, and compared to a previously determined threshold. An example of the results is shown in figure 3.4.

$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y'), \tag{3.1}$$

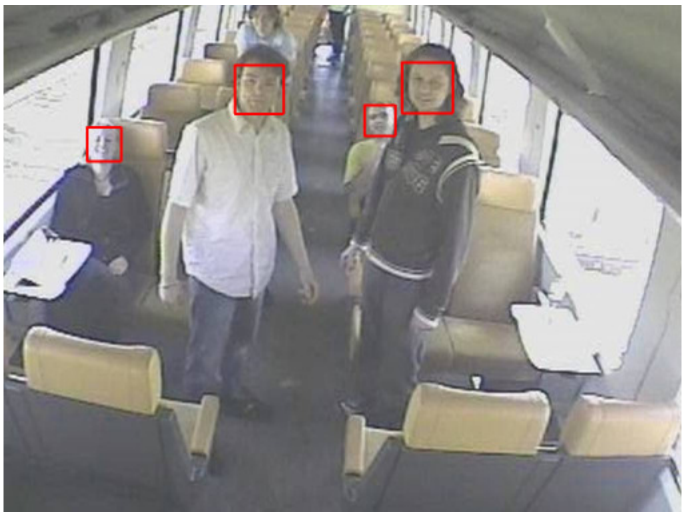


Figure 3.4: Example of face detection using Viola & Jones

The original work by Viola & Jones is proprietary, but since then their algorithm has been reimplemented. The most widely used version is the one included in the OpenCV toolkit. It comes with several so-called *cascades* of classifiers, which are each trained for a certain class of objects. The cascades provided in OpenCV detect frontal faces, profile faces, upper bodies, lower bodies, and full bodies. One characteristic of face detection cascades, is that they are trained for a single view of the face, hence separate frontal and profile cascades included. To perform true multi-view face detection, it would therefore be necessary to either use both provided cascades in parallel, or to train additional cascades for multiple view angles and orientations.
The face detection in this work will be done using the Viola & Jones implementation provided with the OpenCV toolkit. We will use the resulting face detections from this method to perform human tracking in the video samples.

### 3.3.3 Huang & Haizhou

The Huang & Haizhou face detector [20] is a relatively new method introduced in 2004 aimed at multi-view face detection. It introduces a Width-First Search (WFS) tree structure (figure 3.6) that achieves higher performance both on speed and accuracy when compared to other current methods for multi-view face detection (MVFD). It uses a similar approach to the Viola & Jones multi-view face detection method, in that it applies the Adaboost algorithm on weak classifiers for several orientations of faces. It divides human faces into several categories based on the varying appearance from different angles. For each of these categories, weak classifiers are used to detect Haar features. These weak classifiers are then boosted using Adaboost, and construct a face detector. Apart from detecting faces it therefore also detects the orientation of the face. This is illustrated in 3.5.

There is currently no implementation of this method publicly available for comparison with other methods, since it has already been commercially licensed. The results presented so far are very promising since this method not only provides face localization, but in combination with the facial orientation also provides information on where a person is focusing his sight on, and possible in which direction a person is traveling. Knowing where other passengers in a scene are looking at is a very desirable feature in our project, so this detector could be of great future benefit for this project.

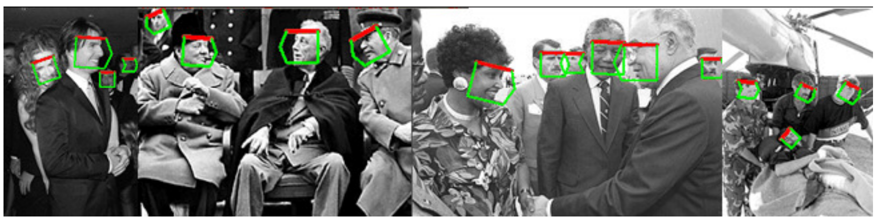


Figure 3.5: Example of face detection by Huang & Haizhou

## 3.4 Person Tracking

There are several possible method for tracking in video frames. To track motion, we must first be able to detect objects. Color value based methods like the ones discussed above can be used for blob tracking, but there are other methods as well such as contour based tracking, or feature based tracking. The Viola & Jones method will return the image coordinates each time it detects a face. Given these coordi-

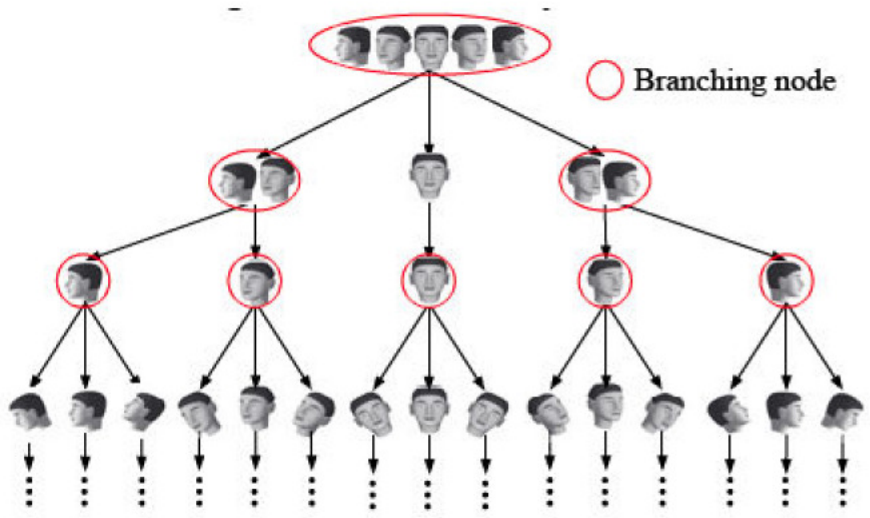


Figure 3.6: WFS tree structure used by Huang and Haizhou

nates for each frame, we can perform tracking using a filtering algorithm on these coordinates to perform the task of data association and motion prediction. Methods that could be applied are for example the Kalman filter, or simpler forms of data interpolation and nearest-neighbor matching.

## 3.5 Behavior Analysis

Human behavior analysis concerns the detection and tracking of people, and more in general, the understanding of human behaviors. A successful human behavior analysis system consists of three major components; human detection, tracking, and finally, behavior interpretation. Human behavior analysis has attracted great interest from the scientific community due to its wide range of applications such as automated video surveillance, video conferencing, virtual reality and perceptual user interfaces [19]. Niu et al. [12] present a framework for recognizing human activities, aimed mainly at outdoor activities, which recognizes behavior based on the observed paths by humans.

The application we are most interested in is naturally the field of video surveillance. In a lot of areas like banks, large department stores, and in our case, train stations and compartments, video surveillance cameras are already in place. Security-sensitive areas like these can greatly benefit from smart surveillance methods. Being able to interpret human behavior can help detect suspicious or otherwise unwanted behavior automatically.

Having motion data available, the task of behavior analysis can be considered a classification problem of time varying feature data. The features in this case could be the locations, or the tracks of each individual being observed over a time period. A set of reference data can then be used to compared the measurements, and construct a best fit. This requires prior knowledge of the behavior to be detected.

### 3.5.1 Template Matching

The template matching approach takes an image sequence and converts it into static movement data, which can then be used to match with a set of templates. It requires pre-stored prototypes for comparison during recognition. In early approaches, optical flow fields were used to model the movement in a scene. The optical flow fields of successive frames are stored, and split into both the horizontal en vertical motion components. A set of these flow fields is then accumulated to form a set of feature vector for a time period. This set of feature vectors is then compared to a template using a nearest neighbor algorithm.

More recent work by Bobick and Davis [4] uses a view-based approach. They use the motion history image (MHI) and motion energy images (MEI) to interpret human behavior. This effectively produces a dual component representation of an action based on the observed motion. This method is based on the property of different actions have different motion history patterns, which can be used to detect and classify human actions. The use of these images makes the algorithm useful for detection of wide ranges of actions, however also it makes it very susceptible to noise. An action must be observed without any occlusion, since any occlusion will greatly influence both the MHI and MEI signatures of the performed action.

In both of the above cases it is assumed that similar actions will have similar measurable features. For this particular work we developed a template matcher loosely based on above methods, but which instead uses the tracked paths of passenger locations to interpret their most likely behavior by performing pattern matching against previously tracked and annotated paths.

# Chapter 4

# Design and Implementation

We want to design a system than can detect a few basic types of behavior, which can then be used to analyze the overall situation in a train compartment. This is a process that requires many steps, from the raw video input, to an automated interpretation of the observed actions. We propose a modular approach that divides this tasks into sub-problems, and describe the workings of each module in this system.

## 4.1 Proposed Surveillance System Framework

Automated video surveillance is a task that includes many subtasks. For this reason we suggest a modular approach for such a system. A diagram of the proposed system is shown in figure 4.1. The video input stage takes the raw input from the video cameras in the train. In the next step this data is processed so that it is fit for the next two stages where the image sequences will be analyzed. This involves tasks like resizing the video, and applying corrections for orientation.

The system then splits into two parallel tasks, motion based, and human detection based analysis. The motion based analysis will perform motion detection and some motion recognition. The human detection side is aimed to detect the humans in a scene and their locations, and track them accordingly. The results of these two modules are then analyzed in the behavior analysis module, which will try to detect behavior, and discern between aggressive and non-aggressive behavior. The final stage of the system will take this behavior analysis, and if necessary produce an alert, in this case when aggression is detected. This could then be used to alert the operator to evaluate the situation on the video stream, or even automatically dispatch the security personnel required for handling the situation.

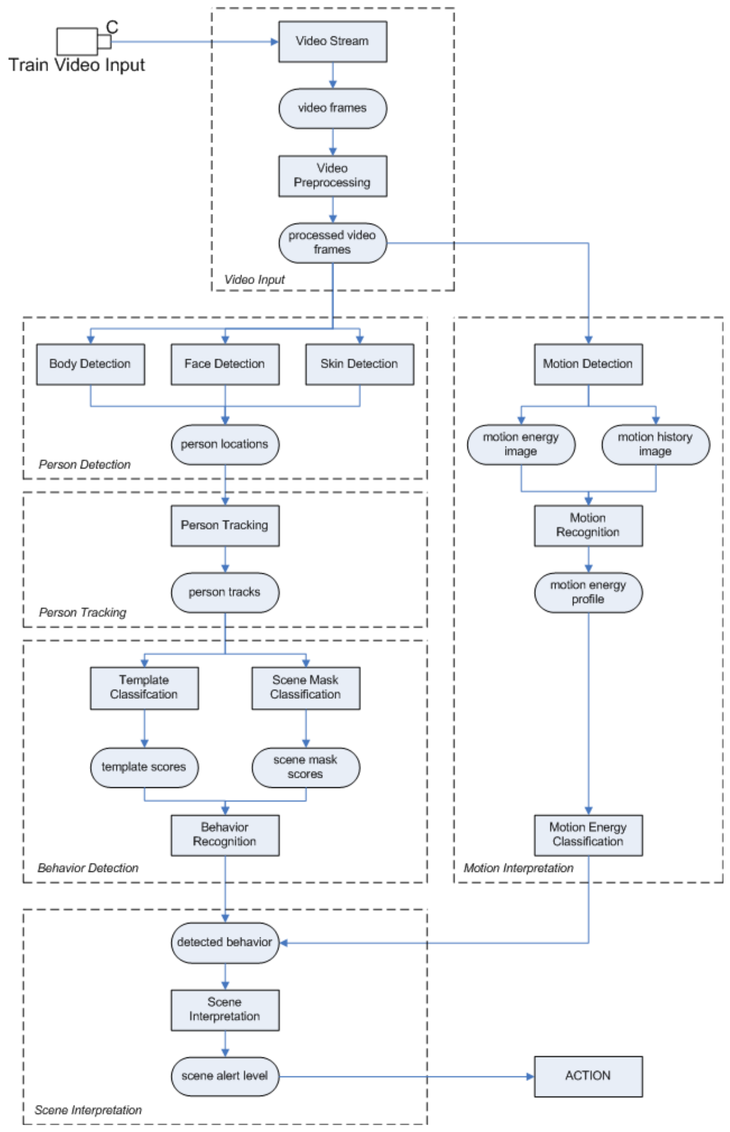


Figure 4.1: Proposed framework for an automated surveillance system

## 4.2 Video Input

The video is received from the cameras directly in raw video format. In our case, the video is coming from a CCTV system already implemented in the trains, and has a unusual 640x256 resolution, and a framerate of 8 fps. The color depth is 24-bit, but reduced greatly when the video is compressed. The average size of a compressed frame will be reduced to around 15 kilobytes per image, resulting in data stream of around 120Kb/s for the video. The video can be received real-time as uncompressed RGB data, or as streaming AVI. For our system we stored the video in MPEG-4 format for reduced size, and loaded them into our system for processing afterwards. The eventual goal is to provide the system with real-time data however.

## 4.3 Image Processing

This work is different from previous works in that it is designed to work under real-life conditions, and therefore faces problems not found in the lab environment. Some of the challenging circumstances we have to cope with in the train compartment include the varying (and unpredictable) lighting conditions. The preprocessing step consists of reducing noise in the video stream. Some smoothing is applied to allow for easier segmentation by the algorithm, since more noise in an image will lead to more small segments being created. This can be done by lowering the color depth. Another common approach in computer vision methods is simply using a grayscale version of an image. A blurring algorithm can be used for even further smoothing. This further reduces the number of regions that will be found in an image, simplifying segmentation.

### 4.3.1 Image Scaling

The video received from the cameras in the trains is in an uncommon 640x256 resolution. The video image is scaled down vertically, as to achieve a interlacing-like effect. This is useful for compatibility with television systems, but the images need to be scaled up to normal resolution for processing. After scaling the images up to 640x512 resolution, we also have a more usable 5:4 aspect ratio. In addition, it is possible to downscale frames to reducing computational complexity. The Viola & Jones algorithm performs quicker when using smaller images, but detection rates suffer, since the method performs better on higher resolution images, and smaller images contain less detectable features. Apart from the cameras available in the train, we also used common webcams for video recording. These cameras record video at a 352x288 resolution.

### 4.3.2 Image Adjustment

The images recorded by the camera in the train compartment during our experiments need to be adjusted before being usable by our methods. The cameras in the train are facing downward at an angle, and have slight variance in their orientation between them. To accurately determine locations of objects, we need to take into account necessary adjustments in the image caused by both camera orientation, and perspective distortion. Objects near the camera will appear larger, and distances between objects will change as a result of this. The images recorded will be projection of a 3-D scene onto a 2-D image. We would like to be able to map the position in our 2-D frame to coordinates in the actual 3-D space. This would be very useful for example to be able to determine exact locations and speeds of objects in a scene. The method for image adjustment used in this work is based on a camera model called Direct Linear Transformation (DLT). The DLT model describes a model for camera calibration using a linear transformation that takes into account zoom, pan, and tilt of a camera. The DLT method is computationally cheap due to being a simple linear transformation. It can however not correct non-linear effects like radial distortion. The imaging process produced by a projective camera can be interpreted as a sequence of three projective transformations. Given a point $p = (x_w, y_w, z_w, 1)$ in homogeneous world coordinates and a point $q = (f \cdot x_i, f \cdot y_i, f)$ in image coordinates corresponding to the projection of $p$ onto the image, then the mapping of $p$ to $q$ can be expressed as:

$$q = K \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot M \cdot p \tag{4.1}$$

Where, $K$ represents the intrinsic parameters of the camera and is given by:

$$K = \begin{bmatrix} \sigma_x & \sigma_\theta & u_0 \\ 0 & \sigma_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.2}$$

With $(u_0, v_0)$ the coordinates of the principal point, and $\sigma_x$ and $\sigma_y$ the scale factors in image $u$ with axes $v$. The parameter $\sigma_\theta$ describes how much the image axes are skewed, and therefore account for non-rectangular pixels. However, in most modern cameras pixels are almost perfectly rectangular, therefore $\sigma_\theta$ will be very close to zero. $M$ represents the extrinsic parameters of the camera and is given by:

$$M = \begin{bmatrix} \ddots & \vdots & \iddots & \vdots \\ \cdots & R & \cdots & T \\ \iddots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.3}$$

Where $R$ is the rotation and $T$ the translation relating the world coordinates to our camera coordinates. By performing the inverse rotation and scaling on the image, the u,v axes of the image is aligned with the x,y axis in the world coordinate system. Figure 4.2 shows the images before and after resizing and adjusting. The slanted lines in the original image have become horizontal lines in the adjusted image. This shows an important consequence of the inverse rotation: horizontal movement in the train compartment should now indeed register as horizontal movement in the adjusted image and vertical movement in the compartment will register as vertical movement in the adjusted image.

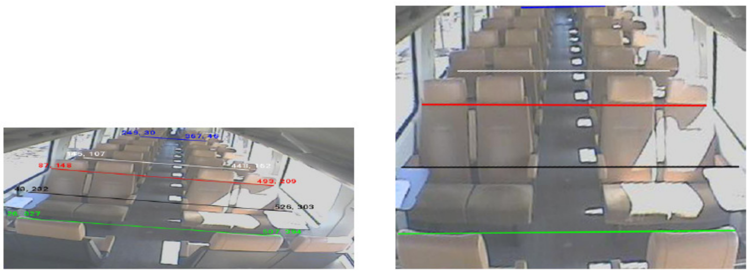


Figure 4.2: Comparison of an original image (left) from the train camera, and the same image after scaling and adjusting (right)

## 4.4 Motion Detection

A first and low level approach to scene understanding is to detect motion. Even without any further investigations, we find ways in which basic motion detection could still be of use to our system. Most motion detection algorithms, including the ones discussed in the previous chapter, will perform a comparison with a previous frame, or a background frame. The result of the motion detection then yield a so-called foreground image, which contains the moving pixels. With this information we can determine several things that might be of our interest:

• Location in the image

The location where we detect motion can be helpful to determine what action is being observed. Motion in the seating area will be expected to be people sitting down, motion in the corridor is expected to be from moving passengers. We can make a scene model in which we can simply designate seating and walking areas as a start.

• Direction

The direction of the movement tells us whether a person is entering or exiting the train, and where a moving object is headed towards. Lateral motion is usually only expected when passengers take their seats.

• Speed

We expect passengers to travel at a certain walking speed. Objects moving at a higher or lower speed can be a sign of aggression or obstruction respectively. To accurately determine speed, we must take into account the perspective distortion of our images by converting the 2D distances to approximations or the real world distances.

Random noise in the video stream can cause apparent motion in an image where in reality no motion is occurring. Changing lighting conditions need to be taken into account. In the case of a moving train we also have the added problem of rapidly changing lighting conditions when for example entering a tunnel, or the changing scenes observed through the windows. These circumstances add a lot of noise to the scene, and make it difficult to perform accurate motion detection. A partial solution is to filter out the most abrupt changes in a scene, comparing results with previous measurements.

## 4.5 Motion Recognition

Motion detection algorithms can give use data on the direction and speed of moving objects in a scene, as well as the total amount of motion observed. We can compare this data to previously recorded scenarios to see if they are similar, and recognize behavior in this manner. Niu and Long [12] described methods to recognize individual human gestures given the localized motion detection data, as well as more general applications. One way of recognizing motion would be to classify scenes according to the amount of movement.

We expect that different scenarios will have different amounts of motion. We would not expect a lot of motion when the train is in transit, but do expect a lot of energy in the scenes where the train stops at station at busy hours, and many people are exiting and entering the train. If we know the average motion energy for common scenarios, we can effectively build a simply scene classifier based on the amount of recorded motion.

Since we have a certain expectation for passenger behavior in the train, we can tune the motion thresholds for different circumstances. For example when observing a lot of motion while the train is in transit, we would suspect abnormal behavior. While the train is stopped however, we will observe many people boarding as well as entering the train, so we would expect high amounts of motion during these periods. We aim to find a threshold for both of these cases later in this work.

Similarly with motion detection, we can take into account the location where the motion was detected. Large amounts of motion energy near the corridor are most likely to be from people exiting and entering the train, while a large amount of energy detected in the sitting areas is most likely a cause for concern. We can also take into account whether the train is in transit or not, since for example we would not expect much movement in the corridor when the train is in transit. A possibility is to assign different weights to different areas of the train, during the different phases of travel. We can then judge each motion energy profile against the current set of rules. This obviously requires some knowledge about the current operation of the train however.

### 4.5.1 Implementation

Applying the motion detection algorithms from OpenCV, we use the acquired Motion Energy Image (MEI) as an indicator of motion of the last viewed frames (opposed to the Motion History Image (MHI) which describes the *recency* of motion). The MEI is a two-color image in which only those pixels are colored that have changed in the last two frames, or in some cases the last several frames. Therefore, we can use this image and the number of colored pixels in it as a measure of the amount of change occurring in the image, that is, the amount of energy in the current scene. We can express the amount of motion as the total number of pixels that have changed, which is the number of pixels in the MEI. A slightly more subtle approach is to use the MHI instead. This image assigns grayscale values to a pixel according to the amount of motion observed in the last few frames. We can use a similar approach, converting the pixel values to an intensity between 0 and 1, or in the case of grayscale color values, between 0 and 255, and simply add these up instead. The resulting data is much more smooth, as the MHI calculation by default filters a lot of noise from the data due to being performed over more than 2 frames. In both cases we used the implementations provided by the OpenCV toolkit.

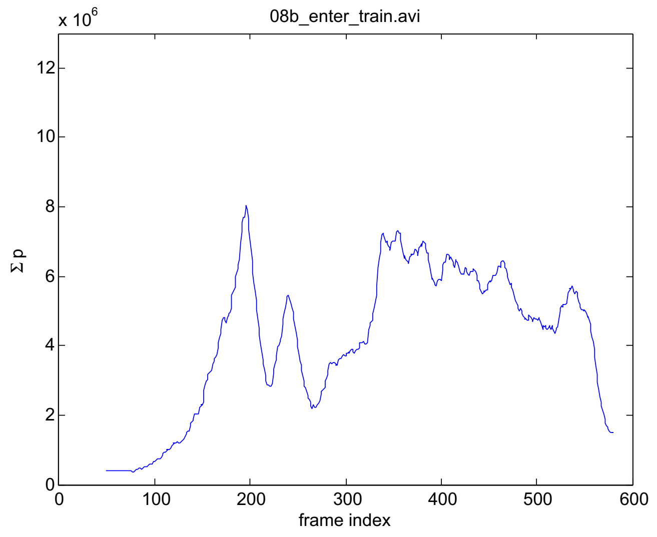


Figure 4.3: Energy graph for scene 8b; people entering the train and walking through the corridor

**Algorithm 1** function motionEnergy(Image i)

```
image $i$
**for** pixel $x$ in image $i$ **do**
    $totalEnergy = totalEnergy + pixel.grayvalue$
**end for**
return $totalEnergy$
```

Adding up motion energy values from images is fairly simple and is implemented as illustrated in algorithm 1. This algorithm is run for each time index, thus supplying an energy value for each recorded frame. An example of motion energy plotted for a scenario is shown in figure 4.3.

## 4.6 Human Detection

### 4.6.1 Viola & Jones

The main method we used for the purpose of human detection is face detection. When a face is detected in an image, this obviously means we have detected a human as well. We used the implementation provided with the OpenCV library, which is completely based on the method proposed by Viola & Jones in [17]. When applied to a single frame, the algorithm will return the regions in the image in which a face is believed to be. This gives us the location and the size of face for each frame, as well as the total number of faces present. This data can be stored for a video stream. We plan to use this data to construct paths of known locations of faces, so we can track movement of passengers in the train. Having detection data over a longer period of time will allow us to use filtering methods to smooth out the paths of passengers, as well as fill in the gaps in detection should we have any false negatives. Since this algorithm is fairly computationally expensive, we also consider the option of not running it every single frame. Using the same filtering techniques, we expect to be able to perform person tracking with only several true positive measurements per seconds, so given high enough detection rates we could opt to run the face detection algorithm on a smaller percentage of the total frames. The Viola & Jones classifier is provided with several cascades, or cascading classifiers. These are each individually trained to detect frontal faces, profile faces, upper bodies and full bodies. We can use a single one of these for example only to detect frontal faces, but we can also combine several of these cascades to boost detection rates if necessary. The advantage would be more accurate detection, due to for example the frontal and profile cascades providing us some basic version of multi-view face detection. The downside however is that we need to run the algorithm several times, increasing the running time of our method, as well as having to combine the detection results into one final location of a person. When using both frontal and profile faces these locations might overlap very well. However when in a single frame both a full body and a face is detected, we need to find a means of determining whether these two (presumably) different coordinates belong to the same person or not. This would require some sort of passenger modeling.

### 4.6.2 Skin Pixel Detection

An alternative to feature based face detection such as Viola & Jones, is color value based skin detection. This method was compared to Viola Jones since it could provide a computationally cheap alternative, which can also run in realtime at higher resolutions. Similarly to face detection, when skin is detected, we have also detected a human. Albiol et al. have implemented a method [3] that is based on the characteristic relation between color values in skin tones of any color, that can effectively be used to determine whether a pixels is part of the skin color range or not.

Detecting skin tones in an image will effectively allow us to localize persons in a scene as well. Skin detection can be done in several ways, most of which are computationally inexpensive. The most basic method is simply determining for each pixel whether it belongs to an certain empirically determined color range, in this case that of skin tones. In the skin model used for this implementation, we use several methods as described in [16].

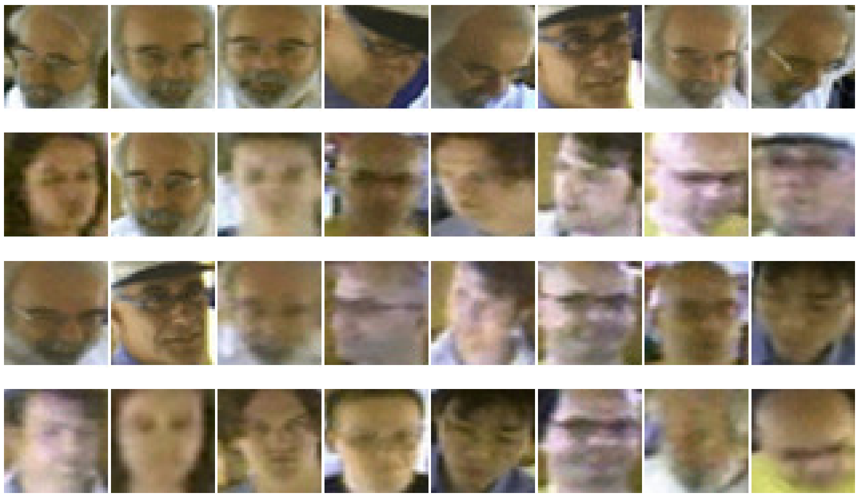


Figure 4.4: Positive data samples all containing skin pixels

Figure 4.4 shows some of the manually selected positive face samples. We visually observe that most skin tones are similar in color value. A feature that we can exploit for automatically detecting skin pixel is the fact that the values of individual color components in skin tones are highly correlated. The most commonly used color models for skin detection are the RGB, YCbCr, and HSV color models. For our first simple algorithms we observe the common color values for skin pixels on our image set, and create a simple pixel classifier based on the recorded values in the positive samples.

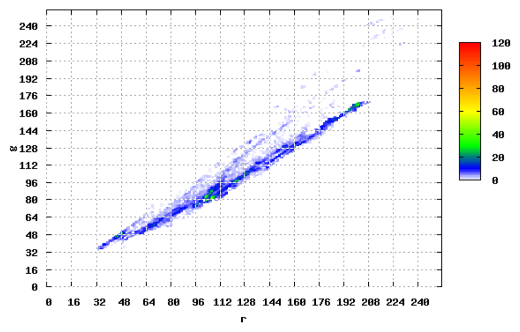


Figure 4.5: *Red* and *green* values of RGB model

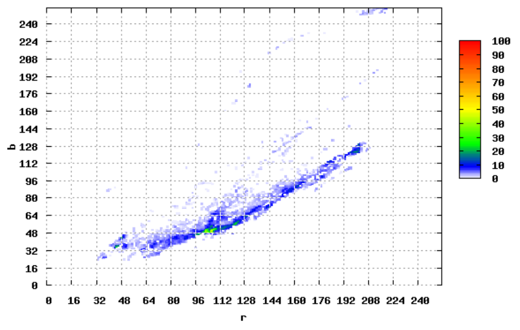


Figure 4.6: *Red* and *blue* values of RGB model

Figures 4.5, 4.6, and 4.7 show the correlation of the $(R, G, B)$ values in the pixel of

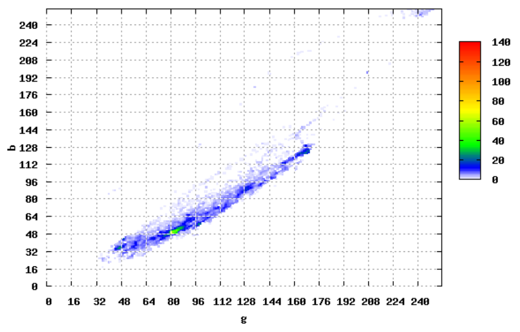


Figure 4.7: *Green* and *blue* values of RGB model

our positive samples (figure 4.4. We can observe several things from these figures. Figure 4.5 indicated that the *red* and *green* color components in skin pixel always seem to appear in a fixed ration, since all the data points are roughly on the same line. The value of the red component is always close to 1.3 times the value of the green component. We observe similar correlation in the values for the *red* and *blue* components (figure 4.6), and the *green* and *blue* components (figure 4.7). Also, for each component there appears to be both a minimum, and a maximum threshold. This knowledge can be used to construct a simple, rule based pixel classifier. The observed data is presented in table 4.1. The table shows the average quotients between the color components. For example, an $(R, G, B)$ value of a skin pixel can be constructed using the values $(R, R \cdot 1.33, R \cdot 1.78)$. The MIN and MAX values are the respective minimum and maximum values observed for each particular component.

Downside of the RGB color representation is that it is very much influenced by lighting conditions, and the channels are highly correlated with eachother. This partially explains the found relations between the color channels, although the correlation can still contain enough information to do some useful analysis on. Another downside is the mixing of chrominance and luminance data. For this reason, most pixel based skin detectors use a different color model [16].

A partial solution to this problem is using *normalized RGB*. The normalized RGB values are easily obtained from the original data:

$$r = \frac{R}{R+G+B}, g = \frac{G}{R+G+B}, b = \frac{B}{R+G+B} \tag{4.4}$$

Since the sum of the values $r+b+g=1$, any single value can be omitted since it represents no significant information, effectively reducing the space dimensionality. The benefit of normalized RGB is that the values of the red and green components in particular, are less dependant on the brightness of the source RGB color. Therefore, this representation will yield similar $(r,g,b)$ values for a skin pixel in varying lighting conditions. This remarkable feature makes normalized RGB a popular choice among researchers [16].

Table 4.1: RGB skin pixel range

| | R | G | B | MIN | MAX |
|---|---|---|---|---|---|
| R | 1 | 1.33 | 1.78 | 48 | 208 |
| G | 0.75 | 1 | 1.55 | 36 | 160 |
| B | 0.56 | 0.65 | 1 | 24 | 128 |

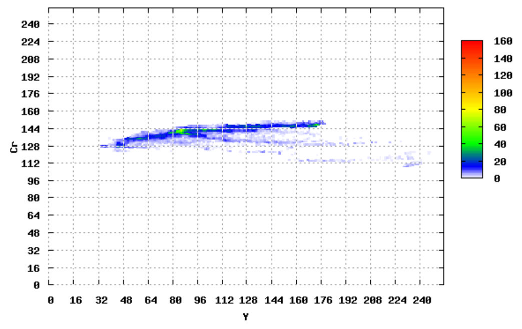


Figure 4.8: *Luminance* and *red chrominance* values of YCbCr model

### 4.6.3 Implementation

Our tested adaptation of the discussed skin detection algorithm uses a skin-classifier to determine whether *foreground* pixels belong to the skin pixel range. The goal is

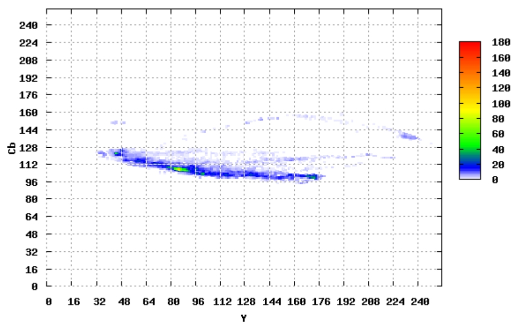


Figure 4.9: *Luminance* and *blue chrominance* values of YCbCr model

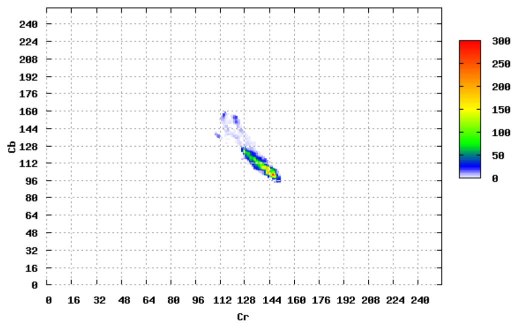


Figure 4.10: *Blue* and *red* chrominance values of YCbCr model

to be able to achieve higher detection range solely for moving faces, thus ignoring any stationary persons and objects, which reduces the number of false positives. We use the already available Motion History Image as a mask to determine which pixels classify and which ones to ignore.

The first algorithm (algorithm 2) calls the isSkin() function for those pixels that are thought not to belong to the background. This is done to further reduce the false positive rate of this algorithm, which is originally quite high at 30 percent [11]. By masking the foreground in this way we also reduce the computational complexity. The pixels which are determined not to belong to the background, and thus being moving objects in the image, as classified using the isSkin() function.

**Algorithm 2** Skin detection using background subtraction

```
Image $foreground$, $background$
for $i$=0 to $foreground.width()$ do
  for $j$=0 to $foreground.height()$ do
    if $background(i, j)$ == 255 then
      if isSkin($foreground(i, j)$) then
        setColor($foreground$, 'white')
      end if
    else if $background(i, j)$ == 0 then
      continue
    end if
  end for
end for
```

**Algorithm 3** function isSkin(Pixel px)

```
$R$ = $px.Red()$
$G$ = $px.Green()$
$B$ = $px.Blue()$
if $R < 95$ and $G > 40$ and $B > 20$ and $max(R, G, B)$ - $min(R, G, B) > 15$
and $|R - G| > 15$ and $R > G$ and $R > B$ then
  return $true$
else
  return $false$
end if
```

When a pixel is in a certain color range (algorithm 3), it is classified as skin. In the resulting images, these pixels are colored in white for illustration.

## 4.7 Human Tracking

The task of tracking humans in video data can be performed in several ways. Some methods rely on first detecting any kind of motion, and then tracking the moving 'blob' [12]. It can then be determined whether this blob is a human or not by analyzing shape, size and movement. This method works by first detecting and tracking the motion, and then classifying the found motion pattern. Another approach is to first detect the humans in each frame, building a face map for each instance. These face maps can then be used to link detected faces in different frames together, for example by using a filtering method like the Kalman filter [8], or some other point tracking algorithm [15]. Both methods can exploit the common characteristics of human motion, such as expected routes, walking speed and directions.

### 4.7.1 Fast & Simple Point Tracking

Veenman et al. [15] present an algorithm that tracks a predefined set of points in a time sequence of images. The method aims to iteratively optimize the correspondences between points, thus aiming to find the most likely track of an object. Missing points are interpolated, making this method a possible candidate for datasets with high false-negative rates. Outliers are either left out, or removed afterwards, thus also taking into account the possibility of filtering out isolated false positives. At the same time, spurious measurements are left out, allowing for filtering of false positives. The algorithm presented is an extension on the Greedy Exchange optimization algorithm by Sethi and Jain [13].

Veenman et al. define the tracking problem as a sequence of $n$ time instances. At each time instance $t_k$ we have $m_k$ measurements $x_{ik}$, with $l \leq i \leq m_k$ and $l \leq k \leq n$. At $t_1$, **m** points ($\mathbf{m} \leq m_1$) are identified among the $m_1$ measurements.

The task is to track these **m** measurements over the whole sequence, or to return a set of trajectories that represent the motion of the $m$ points from $t_1, t_2, ..., t_n$. A trajectory is a fully tracked path of a single point: $X_{i_1 1}, X_{i_2 2}, ..., X_{i_n n}$, with $l \leq i_k \leq m_k$.

A difference from the original Greedy Exchange is the way in which missing data points are handled. Veenman et al. interpolate the missing measurement locations by using preceding and succeeding measurement to generate a new point that maximizes the smoothness of a trajectory. This retains the most motion information as possible and produces more plausible correspondences.

To fill in missing data points, the last two known measurements are used to estimate the distance between the last known measurement and the missing measurement $|x_{ik} - x_{ik-1}|$. If $x_{jk+1}$ is present and $x_{ik}$ is missing, the distance $|x_{jk+1} - x_{ik}|$ is estimated likewise. Besides the distance, the smoothness criterion is also used in some cases. The data is interpolated in similar manner as described above, which return vector estimates $v_1 = \overrightarrow{x_{ik-1} x_{ik}}$ and $v_2 = \overrightarrow{x_{ik} x_{ik+1}}$. An adaptation of this method is used in this work to track persons.

Our approach is very similar to this method. To determine the next point in a track of coordinates, we calculate the direction and speed of the last known points. We then consider the nearest available coordinate measurements in the current frame to pick the best candidate. The criteria used for determining this candidate are the distance to the last known point, and the resulting 'smoothness' of the new path. We determine smoothness using the change in direction and speed of a path, aiming to find the lowest possible change for both values. The reasoning behind this approach is that human motion will be mostly linear and predictable given enough sampling, and at the sampling interval we use we also do not expect many sudden movements. Therefore a new coordinate will likely be an almost linear extension of the last known points. A new candidate point that differs the least in distance and direction will be picked as the next likely candidate.

If no suitable points within the thresholds are found, we search in a limited number of subsequent frames using slightly larger thresholds to account for the extra missing time. This however, reduces the accuracy of tracking results.

To compute the smoothness of a line segment we first compute the angle between de last two known points, and the line between the last known point and the next candidate. We can calculate the angle by first normalizing the vectors such that $|v_1| = |v_2| = 1$. We can then define the angle $\theta$ between these two vectors as $\theta = \arccos(v_1 \cdot v_2)$. Similarly we find the recorded speed between frames by first calculating the last known speed. In the case where we just look at the last 3 frames we simply calculate the euclidean distance traveled in the first sequence, and compare it to the distance in the possible next sequences. The euclidean distance $D$ between points $P = (p_x, p_y)$ and $Q = (q_x, q_y)$ is shown in equation 4.15.

$$D = \sqrt{(p_x - q_x)^2 + (p_y - q_y)^2} \tag{4.5}$$

**Algorithm 4** function tracksinglePerson(Point p, Point q, Measurement r)

```
for coordinate x in r do
  newDistance = $|distance(q, x) - distance|$
  if $newDistance \leq 1.5 \cdot distance$ && $newDistance \geq 0.5 \cdot distance$ then
    currentAngle = $\arccos((p, q), (q, x))$
    if $currentAngle \leq smallestAngle \,\|\, smallestAngle == null$ then
      smallestAngle = angle
      nextPoint = x
    end if
  else
    if !nextpoint then
      nextPoint = x
    end if
  end if
end for
```

The resulting algorithm (4) is then initiated with a simple greedy search to find the first two points that form the first line segment. We use these points to compute the current speed and direction of movement. To find a point that is reached at the current speed, we look for the next data point that is at a distance from our last known point as the other end of that line segment. Thus, we search for a new line segment with a length closest to the original line segment. Once this is established we continue as described in algorithm 4. We consider each detected point in measurement array $r$ found by the face classifier in the current frame, and find the best candidate from those points based on the conditions discussed above. Since we expect the motion to be linear, we prefer measurements with a small angle on the original line segment, but will prefer a point at a higher angle if the distance is much shorter. For points within 50% of our projected distance we choose the point with the smallest angle, beyond we simply take the nearest point.

### 4.7.2 Implementation

Human tracking is done by tracking the detected faces in the video stream. To track the detected faces an algorithm was implemented based loosely on the method described in [15]. This algorithm considers the last $n$ measurements, where in our case $n = 10$, of detected face coordinates. Since there are many gaps in this data due to a low detection rate, considering only a single previous frame for data association yields too coarse results. A empirically found threshold of frames is determined in which the algorithm will look for the best matching posterior coordinate. This is done up to a certain number of frames back, after which if no suitable candidate is found, the point is classified as a possible start of a path, or a false positive. The youngest frames always have preference over the distance between coordinates. A false positive can also be treated as a single true positive, but without any usable parent point to compute a path, and thus not of any value for tracking.

We start looking for parents recursively, up to a predetermined maximum depth. We prefer to find parents in the most recent frames. The most likely parent is the closest point in the previous frames. Certain limitation are taken into account when looking for the best match, such as the maximum distance that can reasonably have been traveled by a person in the elapsed time. Since we a working with a 2d perspective of a 3d coordinate system, we must compensate for the spatial distortion of the acquired images. When no suitable points are found in the previous frame, the next frames are processed recursively. Recent frames are always preferred above possibly better matching points in older frames, hence the algorithm runs linearly through the measurements array. The algorithm is stopped either at a predetermined path length, or when no further candidate parents are found. The resulting data structure is a list of paths, with each patch containing tuples of the coordinates, and the frame index, with a link to the previous point in that path. These paths can then be analyzed to determine what behavior they most likely portray.

Algorithm 5 is the main method which calls the other functions. It has two arguments, a vector $currentMeasurement$, and an integer $depth$. The vector $currentMeasurement$ contains the coordinates of the measured positives of the current frame. The $depth$ value specifies how deep we search for our greedy nearest neighbor search. For each face detected, the most likely parent faces are linked to it up to the specified depth.

**Algorithm 5** function buildPath(**vector** $currentMeasurement$, **int** $depth$)

```
for $face$ in $currentMeasurement$ do
  $current = face$
  for $i < depth$ do
    $parent$ = findClosestParent($face$, $currentframeindex$, $Measurements$)
    if $parent$ then
      $current$.setParent($parent$)
      $current$ = $parent$
    end if
    $i - -$
  end for
end for
```

Algorithm 6 finds these most likely parents. This is a simple greedy search algorithm that takes three arguments. The current face, integer $t$ denoting the frame number, and the measurements vector. It then searches for the best match, that is, the closest available coordinate of a measurement in frame indexes $[t - threshold : t]$, where $threshold$ is usually less than a second, or less than 8 frames.

Algorithm 7 then computes a score for a found path by comparing it to a previously recorded template. The score is the lowest average distance between nodes in the two paths. Since we use larger templates than the usual measurement, we will have

**Algorithm 6** function findClosestParent(**face** $face$, **int** $t$, **vector** $Measurements$)

```
for $index = t - threshold$; $index < t$; $index + +$ do
  if $parent$ = $face$.findNeighbor($Measurements[index]$) then
    return $parent$
  end if
end for
```

to vectors of different lengths. We find the best match for the measurement by comparing it with the template at each possible offset, keeping the best score.

**Algorithm 7** function patternSimilarity(**vector** $foundPath$, **vector** $templatePath$)

```
$offSet = templatePath$.length $-$ $foundPath$.length
while offSet $< 0$ do
  for $nodes$ in $foundPath$ do
    $distance + =$ getDistance($node$, $templatePath$)
  end for
  if $distance < bestDistance$ then
    $bestDistance = distance$
  end if
  $offSet - -$
end while
return $\frac{bestDistance}{foundPath.length}$
```

### 4.7.3 Kalman Filter

The Kalman filter is a state estimation filter implemented using a model and an estimator. A model contains the data structure with the relevant information from our visual scene, in this case the measured location of the presumed persons. An estimator is then used that manipulates this data to compute beliefs about the world. Many computer vision applications involve repeated estimating, as is the case with tracking, of system quantities that change over time. These dynamic quantities are called the system state. The system in question can be anything that happens to be of interest to a particular vision task.

To estimate the state of a system, reasonably accurate knowledge of the system *model* and parameters may be *assumed*. Parameters are the quantities that describe the model configuration but change at a rate much slower than the state. Parameters are often assumed known and static. In this system a state is represented with a vector. In addition to this output of the state estimation routines, another vector introduced is a vector of *measurements* that are input to the routines from the sensor data, given by previous phases of the process. To represent this model we specify the following:

- Estimated dynamics of the state change from one time instance to the next.
- Method of obtaining a measurement vector $z_t$ from the state.

An estimator should be preferably unbiased (when the probability density of estimate errors has an expected value of 0). There exists an optimal propagation and update formulation that is the best, linear, unbiased estimator for any given model of the form. This formulation is known as the discrete Kalman estimator.

The Kalman filter addresses the general problem of trying to estimate the state $x$ of a discrete-time process that is governed by the linear stochastic difference equation

$$x_{k+1} = Ax_k + w_k \tag{4.6}$$

with a measurement $z$, that is

$$z_k = Hx_k + v_k. \tag{4.7}$$

What the filter eventually tries to do is estimating the state $x \in \mathbb{N}^n$ with a measurement $z \in \mathbb{N}^m$. The measurement in this case is the estimated location that is retrieved from the steps that were taken earlier. The random variables $w_k$ and $v_k$ represent respectively the process and the measurement noise. They are assumed to be independent (of eachother), white, and with normal probability distributions.

$$p(w) \approx N(0, Q) \tag{4.8}$$

$$p(v) \approx N(0, R). \tag{4.9}$$

Both process and measurement noise covariance $Q$ and $R$ are constant in this model.

The NxN matrix $A$ in the difference equation (4.6) relates the time step $k$ to the state at step $k+1$, in the absence of process noise. The MxN matrix $H$ in the measurement equation (4.7) relates the state to the measurement $z_k$.

If $\hat{x}_k^-$ denotes a priori estimate at step $k$ provided the process prior to step $k$ is known, and $\hat{x}_k$ denotes a posteriori estimate at step $k$ provided measurement $z_k$ is known, then a priori and a posteriori estimate errors can be defined as

$$e_k^- = x_k - \hat{x}_k^- \tag{4.10}$$

$$e_k = x_k - \hat{x}_k. \tag{4.11}$$

The a priori estimate error covariance is then $P_k^- = E[e_k^- e_k^{-T}]$ and the a posteriori estimate error covariance $P_k = E[e_{\bar{k}} e_k^T]$

The Kalman filter estimates the process by using a form of feedback control: the filter estimates the process state at some time and then obtains feedback in the form of noisy measurements. As such, the equations for the Kalman filter fall into two groups: time update equations and measurement update equations. The time update equations are responsible for projecting forward in time the current state and error covariance estimates to obtain the a priori estimates for the next time step. The measurement update equations are responsible for the feedback, that is, for incorporating a new measurement into the a priori estimate to obtain an improved a posteriori estimate. The time update equations can also be viewed as predictor equations, while the measurement update equations can be thought of as corrector equations. Indeed, the final estimation algorithm resembles that of a predictor-corrector algorithm for solving numerical problems.

The output of the system is the input of a new process cycle. We will use the Kalman filter to predict the future position of a person to aid the next process cycle in the classification phase. This predicted position can be compared to the results in the face detection phase retrieved from the camera footage. The Kalman filter smoothes out the input measurements, in this case the individual location estimates and cancels noise of the measurements. It will produce a location estimate for every person in set $\Omega$ for the next time step. In this system the input measurement can be processed in two ways.

- By combining all location maps to one array and feeding it to a single Kalman filter.
- Assigning each person to an individual Kalman filter. Every location map is assigned to a separate filter.

For this system instead of combining all location estimates and then feed them to a single Kalman filter, every person is assigned to an individual Kalman filter. The reason for this approach is the fact that the set of current people in the room is dynamic. This means that at any time the set can contain more or less, or even no persons when people enter or leave the scene.

In this system the matrix $A$ will be a 4 by 4 matrix and describes the transition model of the system. The columns from left to right representing respectively $x$, $y$, $\Delta x$, $\Delta y$.

$$A = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.12}$$

Multiplication with $x$, the new $x$ and $y$ coordinates are computed by adding their $\Delta x$ and $\Delta y$ components respectively. The input measurement represented by an estimated location of a person will be a vector with two elements, an $x$ and $y$ value. This means the state vector will have the following form:

$$\begin{bmatrix} x \\ y \end{bmatrix} \tag{4.13}$$

The filter is supplied with the current location vector of a person, which is the measurement that is currently taken. This measurement together with a set of mathematical equations and a model of the process can be used to estimate the actual state of the process. This makes sure sudden changes in movement by errors are smoothed out.

## 4.8 Behavior Recognition

The two most basic types of behavior we must recognize are *sitting*, and *walking*. Sitting is specified as a person sitting in a seat, that is, in the areas of the train where we expect a person to sit, and not moving from that position for at least a significant period. Some movement within the boundaries of a single seat can be accepted. Walking is specified as a person walking either through the central corridor in the train, or walking towards a seat. A subset of this behavior is *running*, which can simply be specified as a person moving at a higher speed. Running inside trains is not considered normal behavior, and is often related to aggression, or other volatile situations in a train. We wish therefore to make a special distinction between running and walking. To detect these kinds of behavior we need to know the exact positions of the persons at at least several intervals over a period of time long enough to be able to do some meaningful work with the data. We suggest using face detection for the tasks of detection persons in the train, and to use the data from the face detector to build tracks from these locations. These tracks can then be analyzed for recognizable behaviors.

### 4.8.1 Preprocessing

*Data processing* The Viola & Jones face detector returns values representing the coordinates in the frame where a positive is supposedly detected. A list of these positives is generated for each frame. A matching algorithm will then match each of the found positives to the most likely know predecessor. The technique used for is a simply greedy algorithm that performs a nearest neighbor search. A more sophisticated and possibly more accurate method would be by using a linear filtering method, such as the Kalman filter. The result is a vector with several locations at different time intervals, representing the observed track of a person. Consecutive points in the track do not necessarily mean they were detected at single frame intervals; it is possible not to detect a face for several frames and then detect it again at a location corresponding with the predictions from our filtering method. This will help accounting for false negative measurements, making the method more robust. Algorithms that can be applied for this are effectively occlusion detection algorithms.

After this step we have a list of frame indexes containing the coordinates of faces detected at each frame index, and a collection of lists containing the found tracks to which these positives are believed to belong. The list of tracks is used for further processing by the behavior classifiers to determine the action they represent.

### 4.8.2 Data Classification

#### Template Based Classifier

The template based classifier used for the observed tracks of the actors compares the observed paths to previously observed benchmarks paths. This benchmark comes in two forms. The first is a specific observation of a complete action, that is, a path of observed locations of the actor over time, for which we know the action associated with it (i.e. walking, sitting).

We use prerecorded templates of certain types of behavior, and match the found paths of an actor to these templates. These templates consist of the coordinates of passenger performing actions in the train, such as walking or running. We manually determine the locations for all the frames, and stores several examples as templates. The reasoning is that later on the best match will occur with the template most closely resembling the action taken by the actor. Templates are basically lists of coordinates that the actor occupies at different points in time during this particular action. For example, during the action 'walking' we observe the actor moving from one point in the frame to another, at a reasonably constant speed. For the action 'running' we would observe a similar path, only at a higher speed. Since the paths an actor can follow are limited due to the design of the train compartment (actors can only walk through the pathway in the middle), the observed coordinates will always be more or less in the same coordinate field. To apply this method to an open space would require a different technique, since our templates are dependent on the actual locations where actions are being performed. A simple workaround however, would be a block-matching-like application of the matching, applying a transition to the coordinates at each step. We use a similar method to find the best fitting time interval at which our data matches the template, since measurement tracks and template are not always the same size.

When we have observed a path using our surveillance system, the matching algorithm produces 'tracks' for the template matcher to identify. These tracks tend to be smaller in length than our templates, about 8-32 frames. This is because of the false negatives occurring in the detection data, creating many gaps in the measurements. We are hence looking to match a large-enough subset of this presumed behavior to a template. We arbitrarily determined a minimum size for these measurement. For each measurement we compare the coordinate to the coordinate in a template, and compute the distance between these two points. We do this for every other point in the observed track. The cumulative difference in distance between compared points is expected to be smaller when two paths are more alike. We look for the best possible score by shifting our observed path along the template path. The point at which the paths produce the best match is used to compute the average distance between compared points in the path and the template. If this value is below a previously determined limit, that is, the paths are sufficiently alike, the template is considered a positive match. The size of the compared (subset) measurement must be sufficiently large to produce an accurate result. In our tests we used a minimum length of 10 frames containing 3 or more positive data points, but more data is desired for accurate results.

We define the behavior template as a series of coordinates, in this case the coordinates as they would be observed by the camera. The typical behavior template contains about 15 to 20 data points, at successive time intervals, but larger templates can be constructed to detect more complex behaviors. This set of coordinates is then used to compute the similarity with an observed path, expecting similar behaviors to have similar paths. We can use a method like the Mean Square Error (MSE) (equation 4.14) as a measure for similarity. In our case the error is the distance between the observed point and the corresponding point in the template. We can then compute the MSE for all the $x$ ($\hat{\theta}_x - \theta_x$) and $y$ ($\hat{\theta}_y - \theta_y$) coordinates separately.

$$MSE(\hat{\theta}) = E[(\hat{\theta} - \theta)^2] \tag{4.14}$$

Instead however, we can also use the euclidean distances between point directly. Since values will always be positive, we do not necessarily need to normalize our data anymore. The euclidean distance $D$ between points $P = (p_x, p_y)$ and $Q = (q_x, q_y)$ is shown again in equation 4.15.

$$D = \sqrt{(p_x - q_x)^2 + (p_y - q_y)^2} \tag{4.15}$$

To score an entire measurement against a template we compute the sum of these distances, and divide it by the number of points compared. Equation 4.16 shows the formula to compute the score for measurement $M$ against template $T$ with lag $j$ for the template. We can increase the lag $j$ up to the size difference between our template and measurement, given that the template is the larger of the two, and as such find the best score.

$$score = \sum_{1}^{i} distance(T_{i+j}, M_i) \tag{4.16}$$

Even then a measurement may not be complete however, and therefore not cover the entire template. We might observe a passenger walking in the front of the train, while our template for that action covers a path from the front up to the back of the train. We must therefore try to match our measurements to the corresponding part of the template. To find the best correspondence we simply shift our measurement along the template, and use the lowest average error value because it is the best 'fit' for our measurement.

The average distance of two paths between corresponding points can be used as a measure for similarity. We must note that with larger paths, this method will no longer respond well to variance, because the larger number of measurements means that cumulative differences between slightly different paths will grow too large, and small but significant differences between paths might be evened out. It is therefore important to find an optimal size for general behavior template so that this will not occur, or either produce many template to account for these variations.

### Scene Masking Based Classifier

Certain specific locations in a train will usually show passengers exhibiting very specific behaviors. We can exploit this by using the location of a detected person as an indication for the most likely behavior. To do this we can model a scene according to the expected behaviors for each location. For example, the probability that a passenger will be walking is very low in the areas containing the seats in the train. In the corridor we would expect passengers to be walking instead of sitting however. To designate an area we must indicate the pixels in the image that belong to this area. We can define an area as a polygonal shape (see figure 4.13 (b)). However, certain areas at the edge of these segments can represent several types of behavior. We therefore do not want to have such a strict division between these segments. Instead we can assign a probability to each pixel for each behavior we

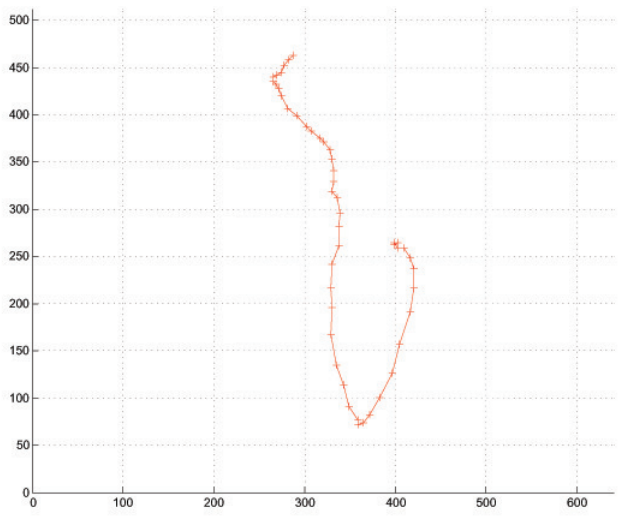


Figure 4.11: Route plot for fragment of scenario 08b, a person entering the train and sitting down

wish to detect, with higher probabilities at the centers, and lower probabilities as well as some overlapping at the edges of these segments. In our model we have created several such 'masks' that cover certain areas that usually have the passengers exhibiting certain types of behavior, i.e. mostly sitting or mostly walking. These 'masks' can be represented as bitmap overlays of the original frames, containing a grayscale value representative of the probability for a behavior. The darker the value for, for example, walking, the higher the likelihood that a passenger observed in that area will be walking. Since we are observing a train compartment, every area has a very clear designation, making it suited for this method of classification.

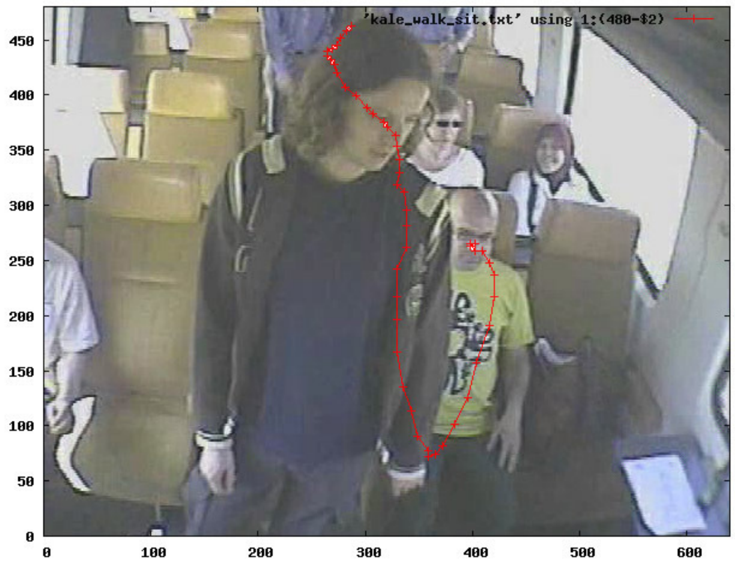
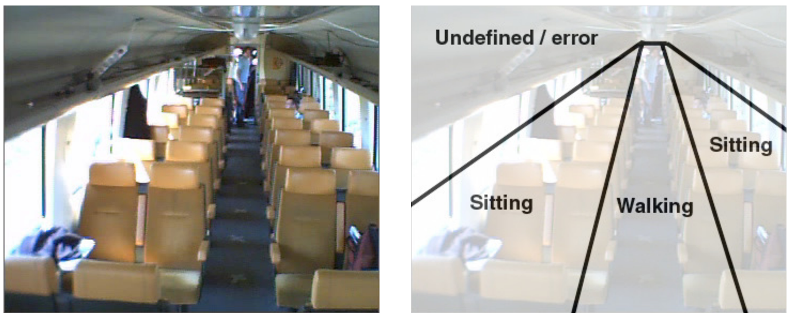


Figure 4.12: Route plot for fragment of scenario 08b, overlayed on last frame of the scene

Due to perspective distortion, sometimes we observe a face in the front of a scene that is overlapping the seats in the rear of the frame. This causes the potential problem for miss-classification. For this reason the transitions from one action to the other produces smooth overlapping values for each action, seen in the mask as a black to transparent gradient. Remaining areas, or areas that are in a location where one would not expect a face to be detected, can be classified with another mask, in this case the 'error' mask. For example the ceiling and window areas of a train compartment are unlikely to ever contain faces. Due to the nature of the Viola & Jones face detector we do however sometimes have false-positives in these areas. The masking technique can be used to quickly dispose these positives and ignore them. The values for each pixel in the frame can be looked up when classifying a path. We expect a passenger walking through the middle of the train corridor to be occupying mostly points where the mask values will be similar to $(M_{sit} = 0, M_{walk} = 1, M_{error} = 0)$. Taking the average of these values over an entire observed path would return the highest average value for the 'walk' mask. This value is then used to determine the action observed in a path. This method can also be used to split a path in two. When a passenger is first walking through the middle of the train and then sits down, there will be a clear difference in the observed mask value somewhere along the path. Since our template matching method only uses general templates containing one action, we can use the mask values to identify the possibility of two or more actions in this track, and therefore split the track into pieces which are more likely to match a part of a template. This is important because although the template matcher will search for a local best match in a template, this does not account for the possibility of several actions within a track. By using the masks to split a path we are more likely to be looking at only a single scenario per track, thus greatly enhancing the accuracy and success-rate of classification.



(a) (b)

Figure 4.13: Empty train (a) and designation of areas for masking (b)

We can classify areas of the train in this way by assigning a value $0 \leq M_{action} \leq 1$ to each pixel for each action. By using values between 0 and 1 we can use the probabilities for each action at each location. The total value of all the mask values for all possible actions must therefore not exceed 1. In noisy environments we can have total mask values $< 1$ for pixels, leaving room for some uncertainty in the measurements, for example in the areas above the chair where we would not expect to detect any faces. In figure 4.13 we define three areas to detect, those designated for walking, sitting and an undefined area in which we would expect only false positives or unlikely scenarios. We can use the error mask to quickly discard data we are not interested in, or which we simply are not able to accurately classify. Oppositely, it can be used to simply detect people being in a certain (prohibited) location, regardless of their actions. In this example the pixels in the corridor could have a mask value $M_{walk} = 1$ to indicate the high probability of observed walking in this areas, whilst the other actions are considered not possible, giving mask values $Msit = 0$ and $M_{error} = 0$.

We can illustrate these areas by assigning a mask to the image. The color of the mask is visualized as grayscale color values, therefore having RGB values $(r, g, b) = (255 - (255 \cdot M), 255 - (255 \cdot M), 255 - (255 \cdot M))$. This produces a higher intensity (in this case black) color at pixels where the likelihood for that particular action is higher. Figures 4.14, 4.15 and 4.16 show the scene masks for respectively walking, sitting, and undefined behaviors.

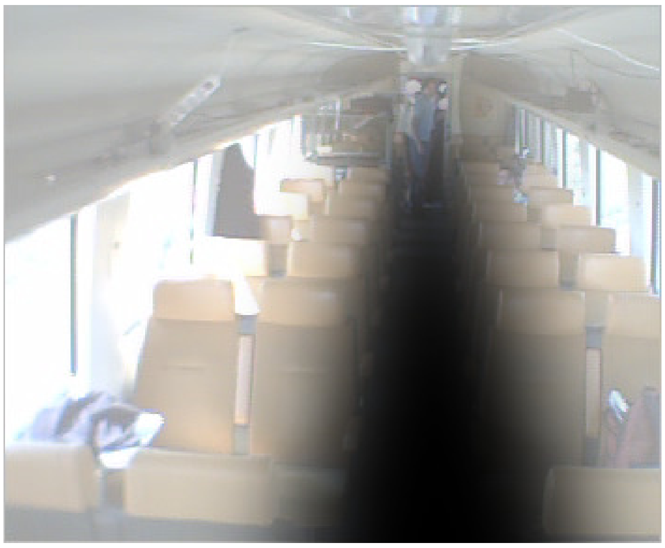


Figure 4.14: Scene mask for walking

We use lower values at the edges of these masks for several reasons. First, at some of the edges we do not have very reliable measurements, such as in figure 4.14 where the faces to be detected at the top of the image would be very small. Second, in the areas where masks overlap we do not know with certainty if for example a passenger is still walking, or in the process of taking a seat. We choose to model this uncertainty by lowering the probabilities in both masks.

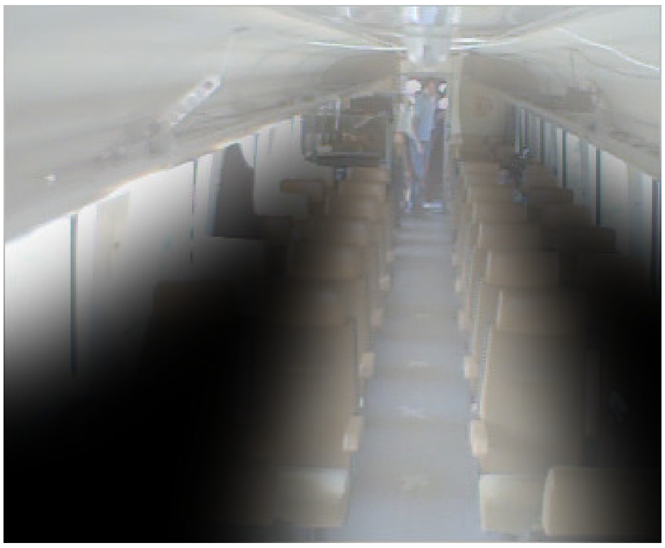


Figure 4.15: Scene mask for sitting

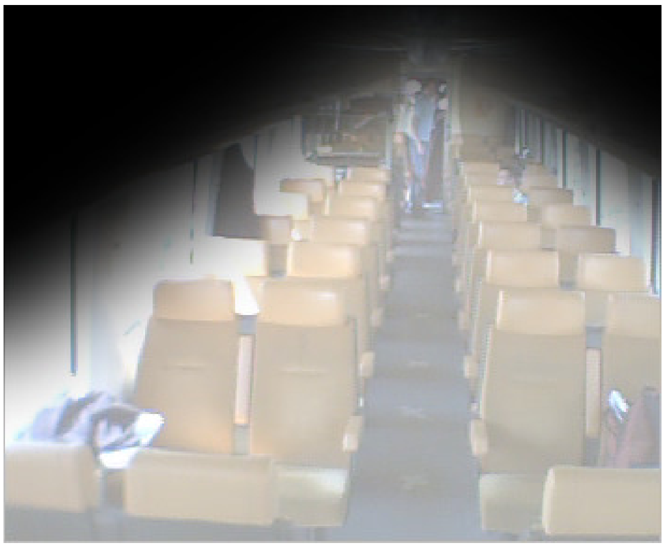


Figure 4.16: Scene mask for error and undefined behavior

We can also designate areas in different ways for different scenarios, for example by using different scene masks during stops and when the train is moving. An example is given in figure 4.17 where the scene is divided in segments according to seat numbers. It is most usable when the train is moving, as we would expect most passengers to be sitting in their seats. We can then alternatively use the walking

masks (figure 4.14) to monitor the amount of movement during this phase, as we would expect it to be much less compared to when the train is stopped.
This method can quickly classify the location of a person, which can be used for static interpretation of single frames. When used in dynamic interpretation, the masking data can be used to detect transitions from on area to another, and be used to again use the corresponding feature based classifier.

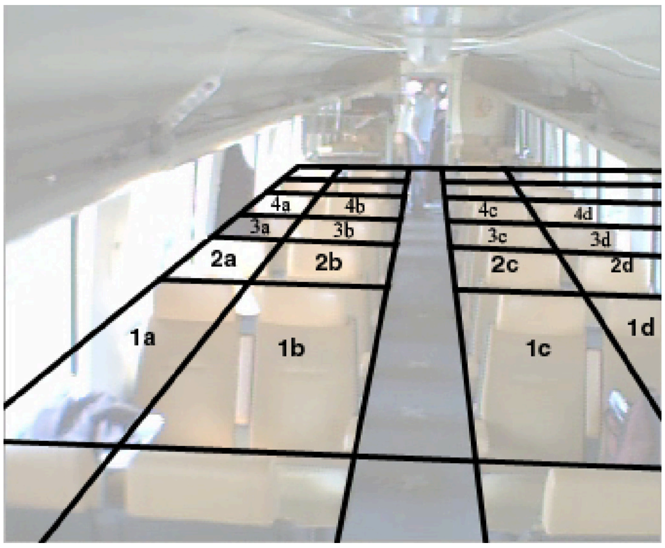


Figure 4.17: Scene segmented by seat numbers

# Chapter 5

# Experiments and Results

## 5.1 Experiments and Setup

We used the train provided to us by the NS to record both video and audio. A number of scenarios was prepared to be performed by actors, with the assistance of a train conductor from the NS. The goal was to produce a database of both video and audio data that we could use for analysis and testing of methods. The actors were asked to depict a number of common scenarios, most involving some kind of aggression. A total of around 90 minutes of video was recorded during the experiments. All the data was stored in separate streams. The data from the cameras provided by the NS into four audio and video streams, the data recorded with webcams in the train in single video streams. Most scenarios were performed in the middle of the train, with most action recorded by two of the fixed cameras, as well as our own webcams.

## 5.2 Motion Recognition

We plotted the motion energy graphs for our scenarios to determine if we can use the amount of motion energy in a scene for interpretation of scenes.

If our assumption that different scenarios will have different amounts of motion proves to be true, then we should be able use motion energy to effectively discern different situations in a crowd. For example, we would not expect a lot of motion when the train is in transit, but do expect a lot of energy in the scenes where the train stops at station at busy hours, and many people are exiting and entering the train. We have plotted the motion energy for several scenarios for illustration. Figure 5.1 shows some frames from scenario 8b, in which several people enter an empty train compartment. Figure 5.2 shows the motion energy values for this scenario. We can see as the people approach the camera, more pixels change, increasing the amount of energy in the picture. Figure 5.3 and 5.4 shows the corresponding images for for a train in transit, with one person briefly getting up from a seat halfway into the scene. Figures 5.5 and 5.6 show the same data for a train with sitting passengers, and a single passenger entering the train, causing a slight spike in the motion energy.

By comparing different scenarios, we can determine thresholds for the amount of motion for train compartments during different phases of travel.

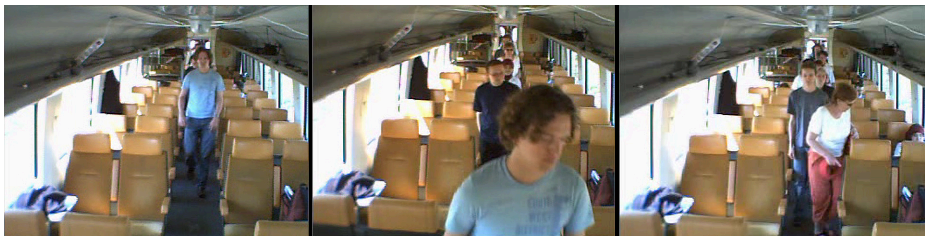


Figure 5.1: Scenes from scenario 8b

We found that the movement of a scene with nothing happening has an upper limit around $1.9 \cdot 10^6 \Sigma p$ (figure 5.4), with $\Sigma p$ being the cumulative amount of grayscale value change of all the pixels between frames, as discussed in the algorithm design. We also found that when the rest of the train passengers are sitting, for example during travel, it is possible to detect the movement of single persons walking through the train compartment. The threshold value for this occurrence was observed to be around $5 \cdot 10^6 \Sigma p$ (figure 5.6). Higher values than this were only found in scenes with several people moving at the same time, or very sudden highly energetic events (figure 5.2). We can conclude from these experiments that the motion

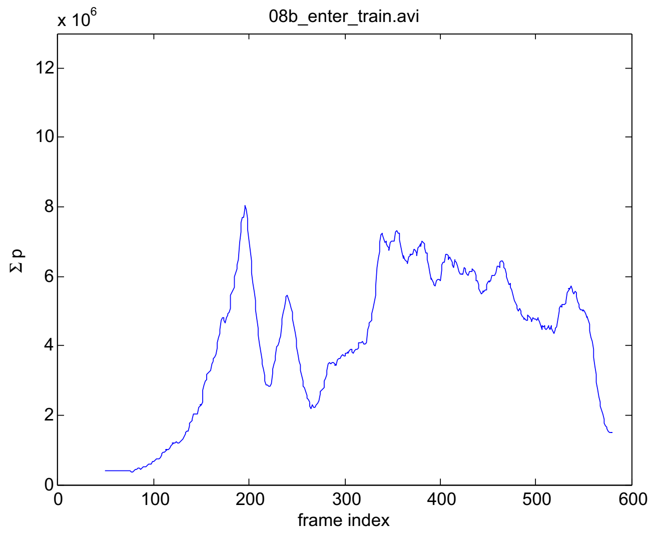


Figure 5.2: Energy graph for scene 8b; people entering the train and walking through the corridor

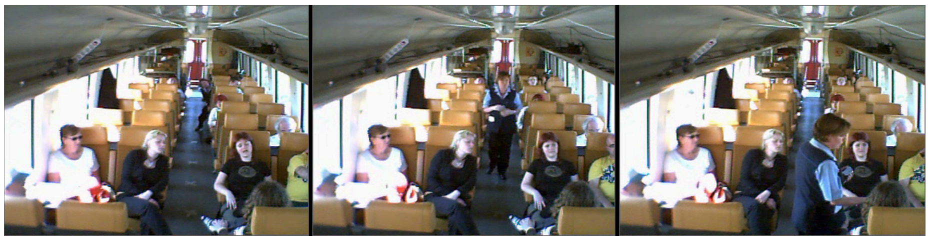


Figure 5.3: Scenes from scenario 12

energy in a scene is a usable measurement in determining the current status of a crowd, given some basic knowledge of the current situation, such as a moving, boarding, or departing train.

## 5.3 Human Detection

### 5.3.1 Face detection using Viola & Jones

The Viola-Jones face detector in OpenCV contains several classifier cascades, each trained to recognize a class of objects, such as frontal faces, profile faces, upper- and full-bodies. We used the included frontal face cascade for our system. While

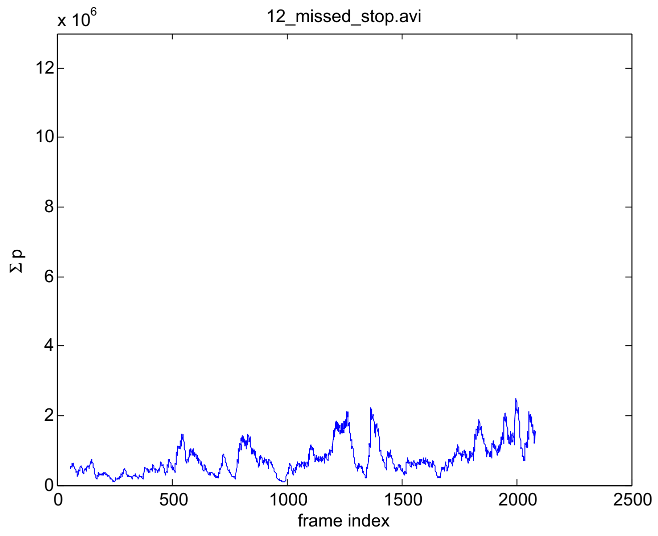


Figure 5.4: Energy graph for scene 12; people sitting in the train, a single person enters the compartment between frames 1100 and 1300

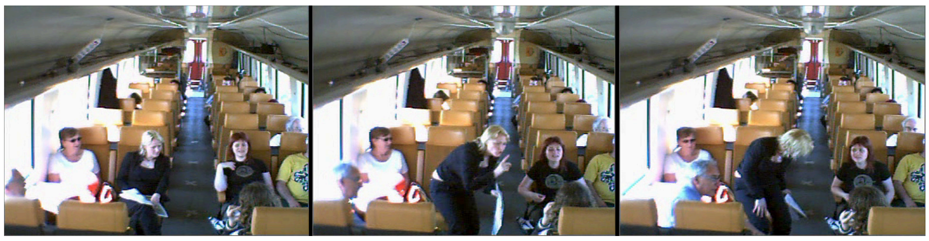


Figure 5.5: Scenes from scenario 13

the literature reports face detection rate of over 90% on the MIT+CMU test set [18], real world performance on our low-resolution dataset was found to be drastically lower.

The Viola & Jones method is capable of accurately detecting faces for which the classifier is trained, in reasonable time. It is however not very robust under noisy circumstances, and the frontal face classifier used is very susceptible to changes in orientation. In large frontal faces we achieved a detection rate close to the rates reported in the literature [16, 17]. However, if the orientation or angle of a face changes beyond the threshold for which the classifier is trained, detection rates fall dramatically, to a point where they hardly contribute to detection at all due to the

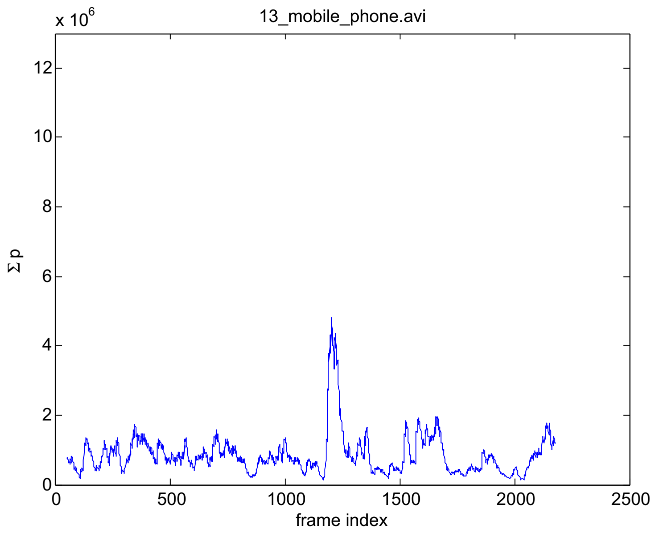


Figure 5.6: Energy graph for scene 13; people sitting in the train, one person leaving seat around frame 1200

relatively higher rate of false negatives. Since we plan to analyze the path of faces however, we need not locate the face in each and every consecutive frame. Using filtering techniques, we estimate that being able to locate a frame about once every 4 to 5 frames should yield enough accuracy to do a basic nearest neighbor match over the past frames to link individual positives together, with larger gaps data association becomes a problem given the movement of passengers. This problem of data association has been widely discussed in [15]. The algorithm we will use is similar in that it aims to find the best matching posterior quickly using a greedy search algorithm.

The data we used to measure the performance of the Viola and Jones algorithm is a video of all the actors entering the train, walking past the camera. All faces can therefore be observed at different distances. The frontal faces in the video were manually annotated for comparison with the returned results. On this dataset containing 574 video frames, and 1379 frontal faces, the Viola-Jones face detector was able to correctly identify a face 84 times, while returning a false negative 81 times. This corresponds to a successful detection rate of 6.1% and a false positive rate of 5.8%. The reason for this low figure is likely to be due to the low quality of the video, and the low resolution of the images. Although the Viola-Jones method can be used for lower resolutions, detecting faces below a 16x16 pixel resolution proved very problematic. As a result, most usable results were obtained at the front of the scene, where faces appear larger due to proximity.

The smallest faces we would like to detect at the 16x16 resolution proved troublesome. Reducing the searching boundaries of the Viola-Jones algorithm to search up to this small resolution increased detection rates somewhat. A very slight further increase was seen when reducing the scaling factor this algorithm uses between subsequent scans of individual images. Equivalent results were achieved by upscaling the video resolution. The limiting factor still seems to be the smaller amount of features that can be found in images of this low resolution.

Table 5.1: Face detection results for 574 video frames of 10 persons entering a train

| | |
|---|---|
| Total faces | 1379 |
| Total detections | 165 |
| True positives | 84 |
| False positives | 81 |

**Face size & detection thresholds**

The Viola & Jones face detector needs features to detect a face. Thus, the smaller a face, the lower the detection rate will be. The faces we wish to detect in our video are small in size, due to the limited resolution of our cameras. We determined that a person standing in front of a camera in our train setup, will produce an image in which the size of the face is roughly 32x32 pixels, or about 1024 pixels. We used a subset as shown in figure 5.7 taken from the Caltech face database by Markus Weber. We varied the sizes of the face images during the tests to find the threshold at which we could still perform reliable face detection.

As shown in figure 5.8, we found a cutoff threshold for successful detection at about 1200 pixels, which is slightly above our desired size. Below this size we notice a sharp decline in detection rates. This means that most faces recorded

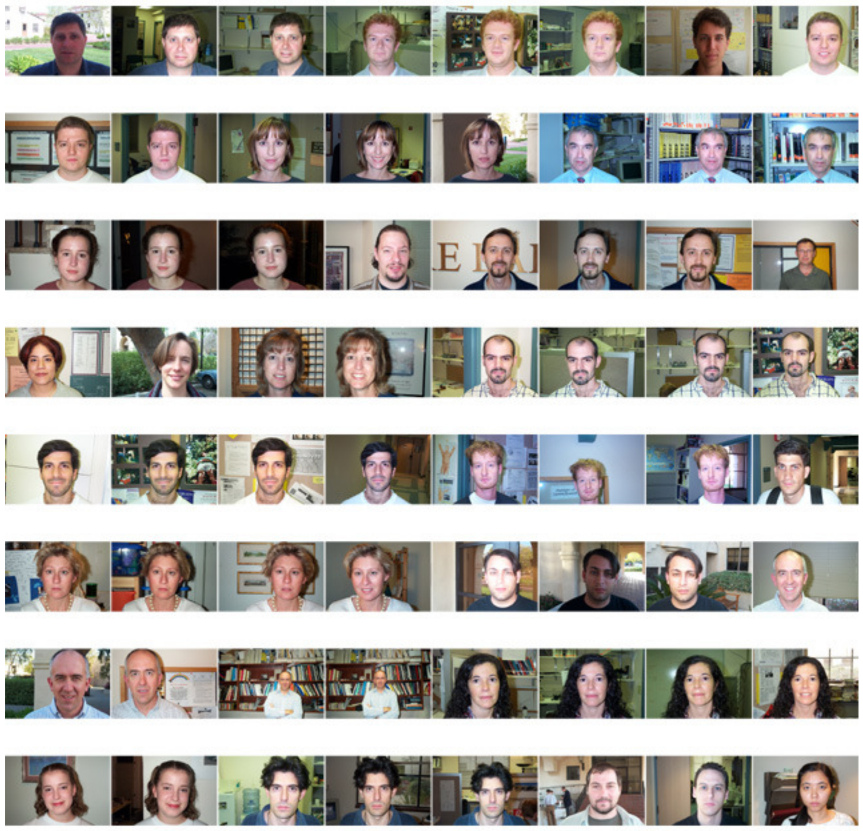


Figure 5.7: Faces used for testing detection at varying sizes

using our setup will be below the optimal size for detection, and we must anticipate a lower detection rate than reported in the literature for laboratory conditions.

To confirm these results we ran another series of experiments involving the same equipment as used in the train videos, only in a controlled lab environment with better lighting. Figure 5.9 shows some of the video we captured in a environment with controlled lighting. Since the resolution is slightly higher than the data we captured in the train at 640x480 pixels, we did find slightly better detection rates, however this was only a small improvement and in line with previous results. We mainly found that the stable lighting conditions eliminate the need for the camera to adjust to changing lighting conditions, thus provide a more stable source of imagery. The data from this experiment furthermore confirms the data gathered in figure 5.8, indicating that the main limiting factor for successful face detection is

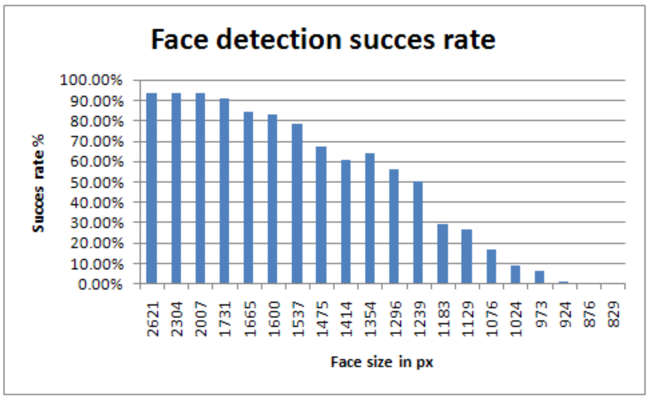


Figure 5.8: Face detection success rate for various face sizes

the image resolution.

Figure 5.9: Scenes taken in lab conditions

The results from this experiment therefore suggest that we need to have images in which the faces are at least 40x40 pixels, and preferably larger. The results obtained correspond well with the data in the literature, in which face size of at least 100x100, or 10000 pixels are suggested [17].

### Viola & Jones running times

Face detection is a computationally intensive procedure. The used method by Viola & Jones is currently one of the fastest methods available. When we wish to use face detection in a real time method however, we need to be able to detect faces at a rate in which they appear in our video stream. For a framerate of 10 frames per second, this leaves about 100 ms for each frame to be processed. We do not need to process every single received frame however, a lower framerate can still provide enough face detection data for human tracking. Also, most surveillance hardware in use records video at framerates of 8 frames per second or even lower.

The experiment was performed on a Pentium 4 3.2GHz processor, with 1024 megabytes of RAM, using the 1.0 version of the OpenCV library.

We have tested the Viola & Jones implementation provided with OpenCV using several resized versions of the same image shown in figure 5.7. The image contains 64 faces. We used an original high-resolution version, that we scaled down to 1024x1024 up to 576x576 pixels. Judging from the results shown in figure 5.10 we can conclude that this method scales linearly for the total number of pixels in the image. We must note that when doubling the resolution from for example 32x32 to 64x64, the total number of pixels becomes four times as big. For comparison, the resolution of the camera is 640x512 pixels, or 327680 pixels. We found that these running times correspond with the data we obtained in our live experiments. We also found that the running times are apparently unrelated to the actual contents of the image; there was no difference in running times between images with faces, blank images, or images containing random noise. This behavior was expected since the Viola & Jones algorithm simply applies its rectangle features over all the pixels of the image.

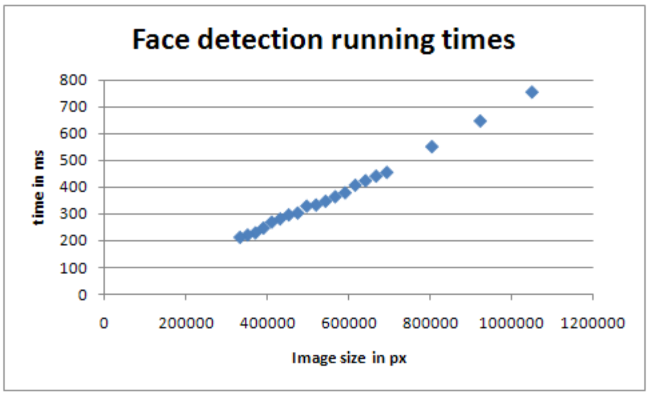


Figure 5.10: Face detection running times for Viola & Jones

### 5.3.2 Face detection using Skin pixel classification

Skin pixel classification is very sensitive to the thresholds selected for the skin color ranges. An empirical approach was used to determine an ideal color range. The example below shows an original frame with several actors appearing in the scene. We wish to detect skin pixels.

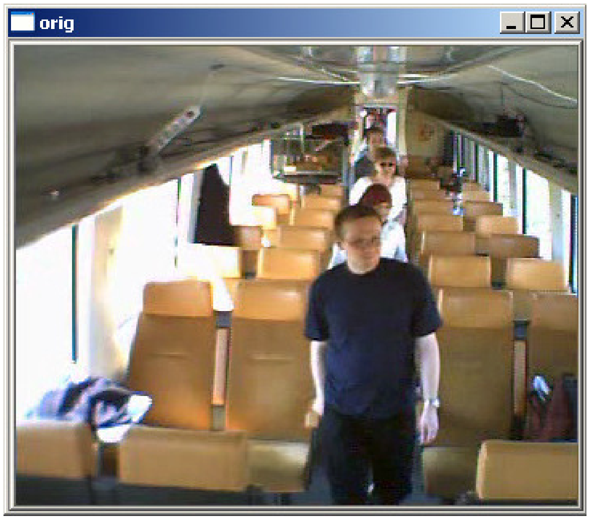


Figure 5.11: Original frame from camera feed

In figure 5.12, we applied the skin detection algorithm to the video frame from figure 5.11. Too many pixels are selected in this example, the pixels selected which are not skin are all considered false positives. The color range apparently needs to be adjusted to detect only skin and in this case, not the seats in the train. Narrowing the color range yields slightly better results, however also leads to more false negatives. The example in figure 5.12 shows a result representative for the optimally achievable results using this technique on our train data.
It appears that using this technique in this environment is not usable without modification, since it still detects many similarly colored pixels. As suggested in [3], this technique can be applied in settings where faces appear against a solid, high contrasting background, but will suffer from noise in other settings. To counter this problem, we limited the search to only the foreground pixels.
When applying the pixel classifier to all pixels in the image, false positives are very

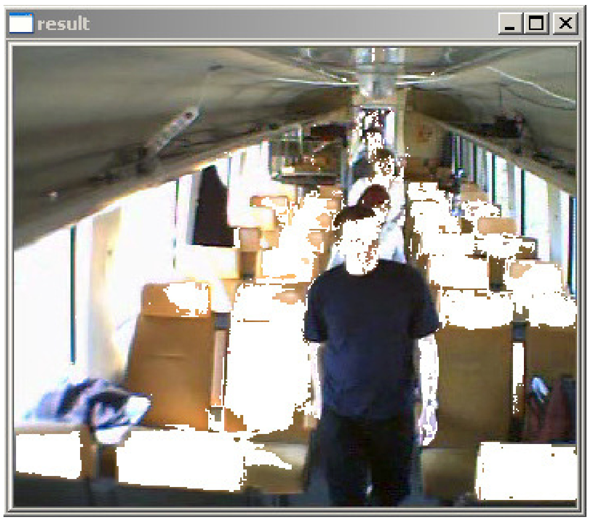


Figure 5.12: The isSkin() algorithm applied to all pixels in the frame

likely to occur. The authors report a false positive rate of 30 percent. In this case however, our input video feed contains a lot of yellow in the color of the seats, and therefore have a much higher false positive rate. To counter this we compare every observed frame to a static background frame. Pixels that differ more than a set amount, will be considered as foreground. Non-changing pixels, such as the seats in this image, will therefore be masked from the isSkin() algorithm.

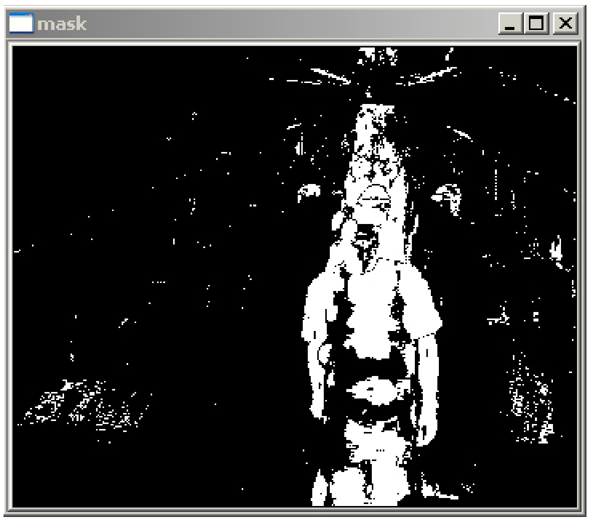


Figure 5.13: Background subtraction generated mask

Figure 5.13 shows a black and white image mask with the white pixels indicating detected movement. When running the skin classifier on just those pixels indicated by the mask, a much lower false positive rate was achieved.

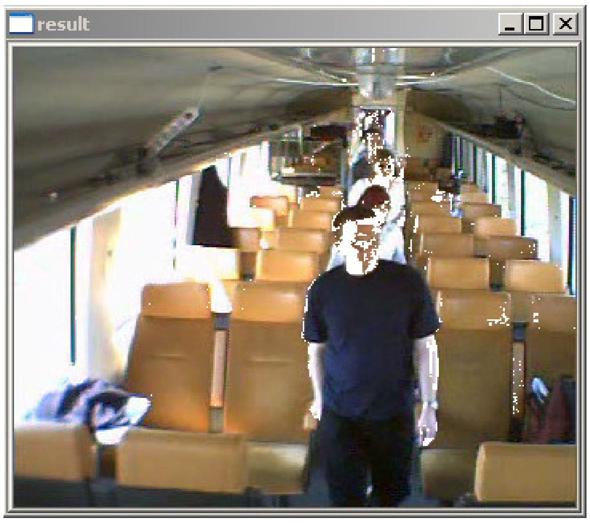


Figure 5.14: The result of the isSkin() algorithm on the foreground pixels

As shown in figure 5.14, the method now much more accurately matches skin pixels since it simply ignores the static pixels, in this case a large part of the seats. An obvious advantage of this method for person detection is its simplicity. Adding a foreground mask like this greatly increases the accuracy of this method, as well as reducing some of the complexity since we need to run our algorithm on less pixels. A downside however is that this method requires finding good threshold values for motion detection, and still has a significant rate of false positives. The biggest problem however is that this method needs high contrast between the faces and the environment. In our particular situation we found that the color of the seats in the train had nearly the same color values as human skin tones, and thus kept producing many false positive readings. We determine that this method is less suitable than feature based face detection for usage in this particular setting.

## 5.4 Tracking and Human Behavior recognition

### 5.4.1 Template matching

We defined several templates for testing against our obtained data from the train experiments. To test the performance of the template matcher we used both positive and negative samples to see if our method can successfully distinguish between them. Figures 5.15 and 5.16 show templates of a passenger respectively walking through the train compartment, and a passenger walking towards the seats and sitting down. The template in figure 5.17 shows a more complex template which involves a passenger first walking through the train, and then sitting down. A reference image from the same camera is shown in figure 5.18.

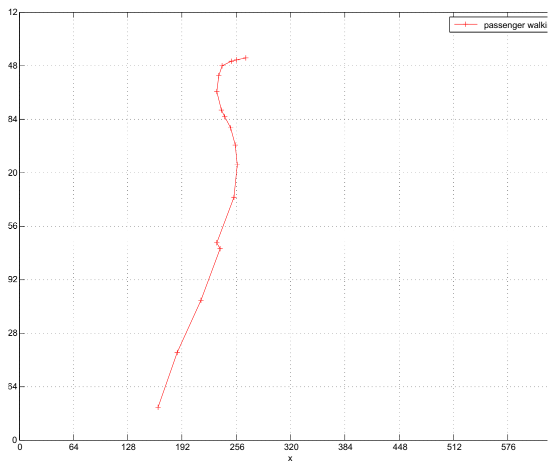


Figure 5.15: Walking passenger template

For comparison, we have plotted some measurements obtain from our camera next to these templates, shown in figure 5.19.

We can clearly see that the measurement from both the 'walking' passengers are similar to our walking template. The other two measurements plotted in figure

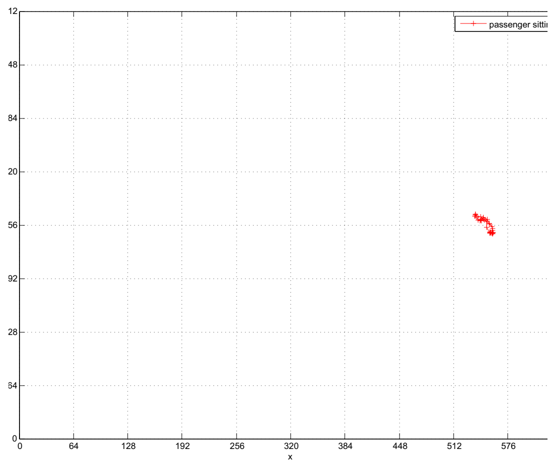


Figure 5.16: Sitting passenger template

5.19 show the locations of passengers showing different behavior. The green line represents a passenger in the top of our view, walking across a seat, while the yellow line shows the locations of a person entering the train from the top of the image, and walking towards a seat, and then sitting down, which is represented by the curve in the line. These last two datasets are clearly different from the first three plots.

We then compute the average difference between each dataset. Since we do not know where a set starts and ends, the beginning and end of a measurement may not perfectly align with our template. The solution is to shift the measurement along the template, and assume the lowest recorded average difference to be the best fit for that measurement. We are interested in the lowest possible value we can find.

We calculate the average distance for each point at these different offsets. The lowest values for the data shown in figure 5.19 are shown in table 5.4.1

We can see that the first three measurements, all containing walking passengers, are able to produce a best fit of about 50 pixels apart per point, while the other

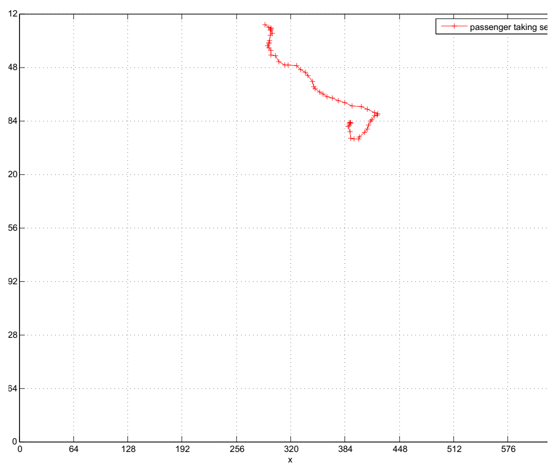


Figure 5.17: Template for passenger walking to a seat and sitting down

measurement all produce error over 100 pixels average difference. It seems from this calculation that the first three paths show more similarity.

### 5.4.2 Scene masking based classification

When observing a path in a scene, we can use the previously constructed scene masks to plot the mask values for each path. We use a part of a scenario in which people are entering the train. In the frames leading up the the scene in figure 5.20, a person enters the train from the top of the image, and walks towards the center. The face detection locations are shown as dots, the track as a red line. We used the scene masks shown in chapter 4.

This person is only detected in the 'walking' mask, therefore the results for the other masks will be zero. We can see the plot for this scene in figure 5.21. We can see in this figure that the 'walking' mask returns positive values, since most detected faces are in the corresponding area. The other two masks do not return

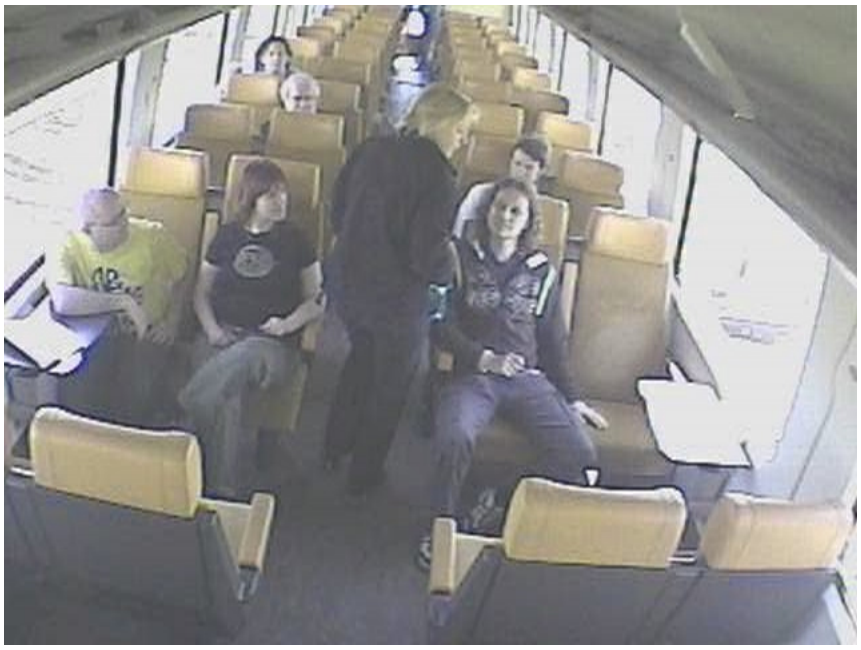


Figure 5.18: Single frame from train camera

Table 5.2: Average distance between corresponding points for measurements in figure 5.19

| Measurement | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 0 | 50 | 24 | 126 | 127 |
| 2 | 50 | 0 | 52 | 119 | 152 |
| 3 | 24 | 52 | 0 | 118 | 110 |
| 4 | 126 | 119 | 118 | 0 | 138 |
| 5 | 127 | 152 | 110 | 138 | 0 |

any values, since no single detected face was in any of their areas.

We can also use the scene masks to divide a measurement into segments, one segment with a passenger walking, another segment with just the passenger sitting. This will also make a measurement more likely to fit a template.

Figure 5.22 shows the track followed by a person first entering the train from the top of the scene, and then walking through the corridor and taking a seat. The plots for the same three masks are shown in figure 5.23. As we can see in this case, as the person enters the train, the mask for the 'walking' behavior returns positive values, and then declining values as the person exits the designated walking area.

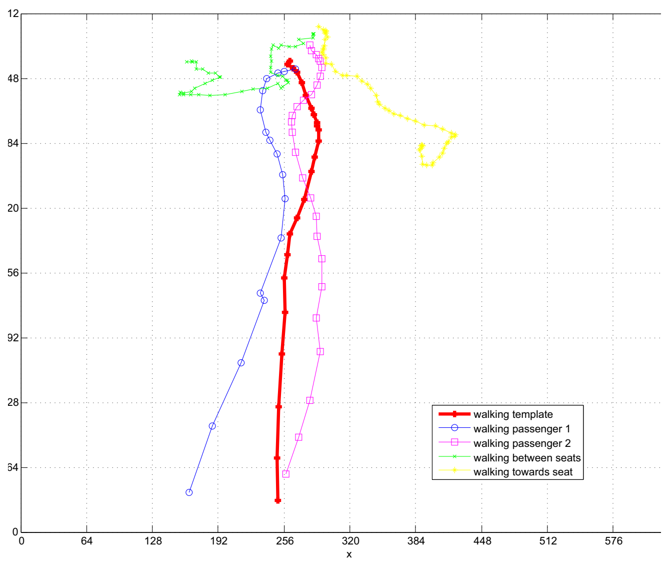


Figure 5.19: Walking template compared with several measurements

At this time the mask for the 'sitting' behavior starts returning higher values as the person enter the seating area and remains there until the end of this scene. We can therefore conclude that this person was first walking when detected, and next sat down. The 'error' mask values remained at zero, since this track did not contain any values outside of the other two mentioned masks.

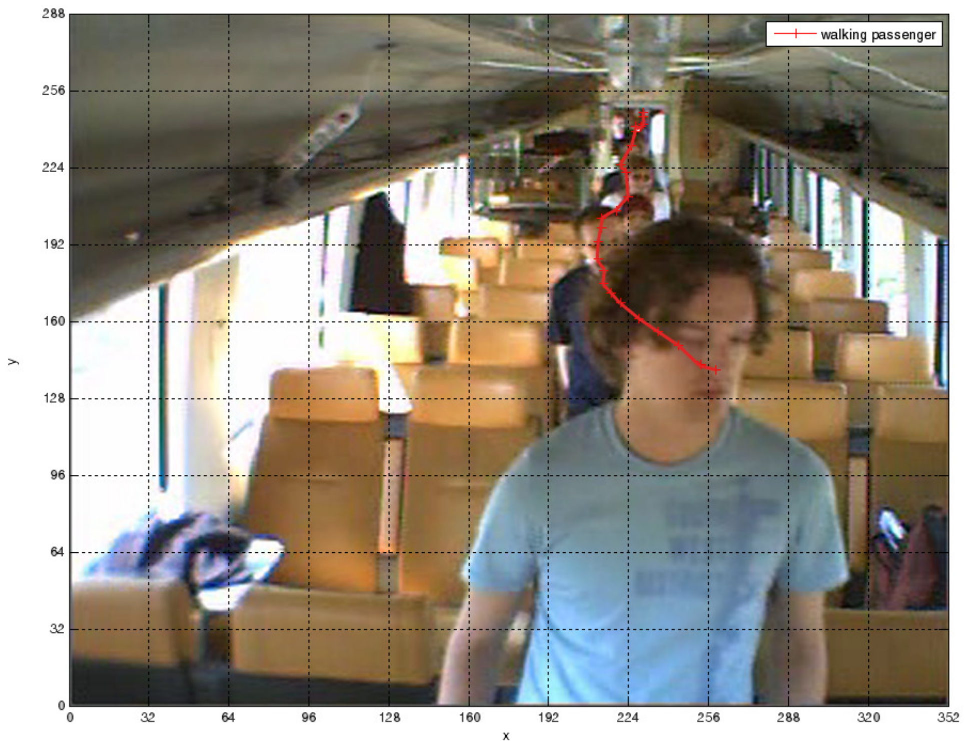


Figure 5.20: Person being tracked inside the train, with tracking overlayed on last frame of scene

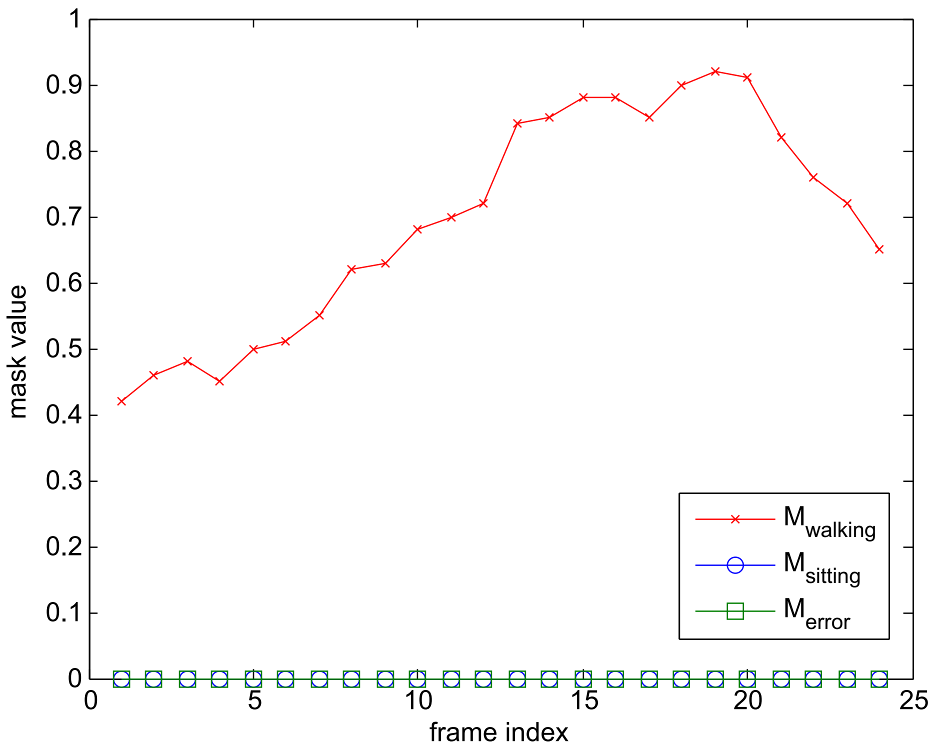


Figure 5.21: Mask values for person tracked in figure 5.20

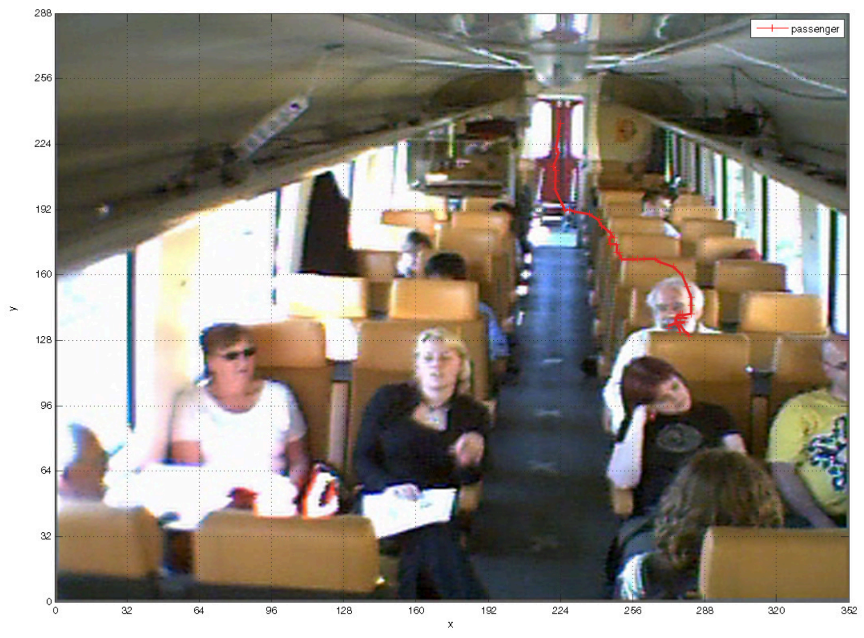


Figure 5.22: Person being tracked inside the train, with tracking overlayed on last frame of scene

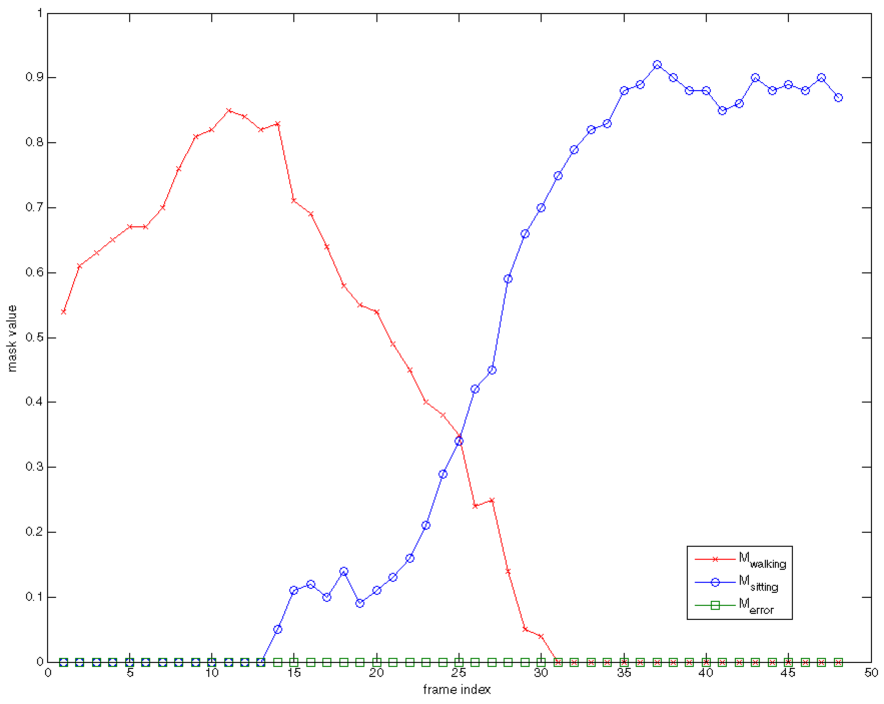


Figure 5.23: Mask values for person tracked in figure 5.22

# Chapter 6

# Conclusions

## 6.1 Conclusions

In this work we present a design and implementations for a system to detect and interpret human behavior. We discussed how several out of the box methods can be used to detect low level features, and how to use these methods to develop a high level rule based reasoning system that can combine these features into recognized behaviors.

As to our research goal,

*To design a system composed of a network of video cameras, based on using intelligent computer vision techniques, to be used for automated video surveillance and crowd monitoring,*

we have found that to design such a system we need to implement several intelligent computer vision techniques,and integrate them so they can work together.

The first challenge was to provide a good video stream for input. While it is possible to achieve this using video cameras, it turned out later that the resolution does not provide enough detail for accurate face detection or other algorithms concerning graphics. Video postprocessing to adjust for lighting conditions, camera orientation and color balance proved to be a feasible task however.

Next we faces the challenge of detecting humans. We focused on two approaches, to detect skin and to detect faces. Skin detection turned out to be very complicated to use since the skin tones highly resemble colors present in the everyday environment. Further more the success rate that was achieved was not good enough for accurate detection, and it also adds a new problem in that arms and legs are also detected, which then needs to be taken into account. We concluded that this method was not the best available method for the task at hand.

Face detection using feature based classifiers is much more promising. When provided with detailed data, face detection based on the method proposed by Viola & Jones is indeed very accurate. Since however, the level of detail in the provided camera hardware was limited, most of the faces in the video data were too small for accurate detection. A limitation of this method is also that it is computationally expensive. To achieve the desired detection rate for this purpose, we need images so large that real-time data acquisition would require more computing power. The resulting size of the data would also need to be taken into account, especially for preprocessing. Also it would require better cameras and faster (multiple) processors. Since this is one of the crucial steps in our system as proposed in our design, this is crucial to the effectiveness of a surveillance system.

The next challenge was to track the detected humans. When a steady stream of true positives is obtained, it is relatively easy to apply existing filtering algorithms, or simpler point matching algorithms to the data. We modified an existing greedy search method, which turned out to be very well suited for this purpose, since it was fast, and the high linearity of the data meant that it was still possible to achieve accurate and usable results, even when the data contains many gaps due to lower face detection rates.

Given this processed data, we are able to use pattern matching and similar techniques to classify the human behavior in our data. Again we found that when there is decent resemblance between templates and measurements, this method performs well. Most of the situations we are interested in, are simply out-of-the-ordinary situations, which could be detected by comparing values such as movement direction, speed, and duration to predefined scenarios.

Our goal was to design a system that can be used for real time video surveillance and crowd monitoring. We found that existing techniques can contribute to such a system, and given the right circumstances and application of available hardware can perform a valuable task in detecting aggression. We can therefore conclude that the design of a fully automated, intelligent surveillance system is within the possibilities of current technology and can be achieved by integrating available state of the art computer vision and tracking algorithms. A significant amount of work however still needs to be done of the task of human detection and scene interpretation for such a system to be truly able to replace a human operator. In the meanwhile however, computer based surveillance systems that apply some type of intelligent reasoning can already be a valuable tool to aid in supervising large areas such as a train consisting of multiple cars. A fairly simple system can already be of great use simply to quickly alert personnel of possible escalations in the train and to pinpoint locations where human attention might be required.

## 6.2 Future work

At the moment, most gains could be made by improving human detection rate. This problem is currently mostly limited by computational power, and image resolution. Some gains are also being made in the accuracy of multi-view face detection [20], which is another obvious next step in being able to accurately detect humans.
Multi-camera based detection can be added to such a system to provide a larger field of view, but also to combine data across several video streams, which can then be used to help construct a three-dimensional image of a scene. Some preliminary work has been presented in [6, 5].
Another addition to such a system would be to use more modalities than just vision to detect behavior, most notably sound. Sound localization can be used to quickly point a camera to a sound source, and provide more information to a system thus helping it to better interpret a scene. Being able to identify noise or even speech can greatly increase the accuracy of aggression detection, as well as provide whole new means of detection a whole range of new behaviors and situations.

# Bibliography

[1] Real-time face detection and tracking for mobile videoconferencing. *Real-Time Imaging*, 10(2):81–94, 2004.

[2] E. Acosta, L. Torres, A. Albiol, and E. Delp. An automatic face detection and recognition system for video indexing applications. *Acoustics, Speech, and Signal Processing, 2002. Proceedings. (ICASSP '02). IEEE International Conference on*, 4:IV–3644– IV–3647, 2002.

[3] A. Albiol, L. Torres, and E. Delp. An unsupervised color image segmentation algorithm for face detection applications. In *IEEE International Conference on Image Processing*, pages 681–684, Octtober 2001.

[4] A. Bobick and J. Davis. Real-time recognition of activity using temporal templates. In *WACV '96: Proceedings of the 3rd IEEE Workshop on Applications of Computer Vision (WACV '96)*, page 39, Washington, DC, USA, 1996. IEEE Computer Society.

[5] R. Collins, A. Lipton, and T. Kanade. A system for video surveillance and monitoring. In *American Nuclear Society 8th Internal Topical Meeting on Robotics and Remote Systems*, 1999.

[6] F. Cupillard, A. Avanzi, F. Bremond, and M. Thonnat. Video understanding for metro surveillance. *Networking, Sensing and Control, 2004 IEEE International Conference on*, 1:186–191 Vol.1, March 2004.

[7] D. Datcu and L. Rothkrantz. Facial expression recognition in still pictures and videos using active appearance models: a comparison approach. In *CompSysTech '07: Proceedings of the 2007 international conference on Computer systems and technologies*, pages 1–6, New York, NY, USA, 2007. ACM.

[8] D. Datcu, Z. Yang, and L. Rothkrantz. Multimodal workbench for automatic surveillance applications. *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1–2, June 2007.

[9] H. Ferwerda, G. Verhagen, and E. de Bie. Onderweg naar een veiliger openbaar vervoer. *Ministerie van Verkeer en Waterstaat, Adviesdienst Verkeer en Vervoer*, June 2005.

[10] W. Hu, T. Tan, L. Wang, and S. Maybank. A survey on visual surveillance of object motion and behaviors. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 34(3):334–352, Aug. 2004.

[11] P. P. Jure Kovac and F. Solina. Human skin colour clustering for face detection. *Submitted to Eurocon 2003 -International Conference on Computer as a Tool*, 2003.

[12] W. Niu, J. Long, D. Han, and Y.-F. Wang. Human activity detection and recognition for video surveillance. In *IEEE Int. Confenrence on Multimedia and Expo*, Taipei, Taiwan, 2004.

[13] I. K. Sethi and R. Jain. Finding trajectories of feature points in a monocular image sequence. *IEEE Trans. Pattern Anal. Mach. Intell.*, 9(1):56–73, 1987.

[14] S. van Hese. Real-time localization and tracking of multiple people in a closed environment with facial detection using a multi-camera setup. Master's thesis, Delft University of Technology, 2008.

[15] C. Veenman, E. Hendriks, and M. Reinders. A fast and robust point tracking algorithm. *Image Processing, 1998. ICIP 98. Proceedings. 1998 International Conference on*, pages 653–657 vol.3, Oct 1998.

[16] V. Vezhnevets, V. Sazonov, and A. Andreeva. A survey on pixel-based skin color detection techniques. In *in Proc. Graphicon-2003*, pages 85–92, 2003.

[17] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. volume 1, pages I–511–I–518 vol.1, 2001.

[18] P. Viola and M. Jones. Robust real-time face detection. *Int. J. Comput. Vision*, 57(2):137–154, 2004.

[19] L. Wang, W. Hu, and T. Tan. Recent developments in human motion analysis. *Pattern Recognition*, 36(3):585–601, 2003.

[20] B. Wu, H. Ai, C. Huang, and S. Lao. Fast rotation invariant multi-view face detection based on real adaboost. *Automatic Face and Gesture Recognition, 2004. Proceedings. Sixth IEEE International Conference on*, pages 79–84, May 2004.

[21] Z. Yang, A. Keur, and L. Rothkrantz. Behaviour detection in dutch train compartments. In *Proceedings of Euromedia 2008*, pages 52–57. Eurosis, April 2008.

[22] Z. Zhang. Modeling geometric structure and illumination variation of a scene from real images. In *ICCV*, pages 1041–1046, 1998.

[23] W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld. Face recognition: A literature survey. *ACM Comput. Surv.*, 35(4):399–458, 2003.

# Appendix A

# BEHAVIOUR DETECTION IN DUTCH TRAIN COMPARTMENTS

Z. Yang, A. Keur and L. J. M. Rothkrantz
Faculty of Electrical Engineering, Mathematics and Computer science
Delft University of Technology
Mekelweg 4, 2628CD Delft, The Netherlands
E-mail: {Z.Yang, L.J.M.Rothkrantz}@tudelft.nl

## KEYWORDS

Aggression detection, Dutch train compartment, aggressive behaviour, Multi-modal cameras

## ABSTRACT

Aggressive behavior in public places can cause great distress on the part of innocent bystanders. This paper describes research done to automatically detect forms of aggression by recognising the behaviour of people in a train. A dataset was gathered in a real train with semi professional actors performing aggressive and non-aggressive scenarios. We developed a system to recognize a number of predefined behaviours from features extracted from the sensor data.

## INTRODUCTION

Safety in public places has gained a lot of attention in the past few years. The need for increased surveillance in public places as a guard against terrorist attacks and other forms of aggression, have made people more tolerant of cameras and microphones in public areas. With the increased number and complexity of these devices, people also expect a higher level of safety. Up until now, there has been limited success in living up to these expectations. The Dutch railway company (NS) for example, strives to decrease the number of incidents on Dutch trains by equipping them with cameras (e.g. the trains in the Zoetermeer Stadslijn). The primary role of these cameras is to increase the feeling of security of the passengers and to have a deterring effect on people with bad intentions. However, the camera images have to be inspected manually. With the growing number of camera images to process, the chances of detecting aggression manually becomes very small.

The goal of an ongoing project at the Man-Machine Interaction (MMI) group in Delft is to solve this problem by creating a system to automatically detect aggression as it is happening or is about to happen. In this paper we explore methods and techniques to describe normal and unusual behaviour in a train compartment. First we describe the train compartment and the situations we want to detect. We also specify the particular problems that we have to cope with in our environment such as varying light conditions and occlusion. Faced with these problems we present our solution which uses off-the-shelf classification algorithms.

The remainder of this paper is structured as follows. First we give an overview of the background and the related work in the area. Then we describe the data that was captured in the train. Next the classification of the behaviour that we want to detect. Afterwards come the detection methods and the results. We finish with a discussion and conclusions.

## BACKGROUND

With the availability of inexpensive sensors and the ever increasing processing power at our disposal, the number of surveillance and surveillance related research projects has increased. The most commonly used modalities for this purpose are video (Foresti et al., 2005; Javed et al., 2003), audio (Clavel et al., 2005; Härmä et al., 2005) or a combination of both (Beal et al., 2002). We observe that in complex surveillance environments, such as in public transport systems, the combination of multiple modalities is more common, e.g. PRISMATICA for railways (Velastin et al., 2002) and ADVISOR for metro stations (Cupillard et al., 2004).

The usual approach to the surveillance problem is to view the individual events (e.g arm motions, gestures) as related parts of a bigger scenario e.g. fighting, ticket checking. A scenario is defined as a combination of states, events or sub scenarios. This means that in the representation of the scenario, the influence of the individual events on the outcome of the scenario is also included e.g. the occurrence of shouting might cause the ticket checking scenario to escalate. At runtime, the surveillance system tries to infer the consequences of the activity/scene recognized based on this prior knowledge. Bayesian networks can be used for the inference, but other approaches have also been proposed, including multi-layered HMMs (Zhang et al., 2006) and CHMMs (Oliver et al., 2000).

As suggested above, the surveillance system can be divided into two steps. A first step to detect the features and events in the incoming sensor data and a second step to combine these events (over time) into activities and scenarios. For surveillance/activity recognition in relatively controllable environments (e.g. rooms, offices) data can be collected quite easily. Thanks to the controlled environment, feature extraction and event recognition can also be robustly performed.

In the train compartment however, we have to cope with more challenging circumstances. These include the varying (and unpredictable) light conditions throughout the course of the day, occlusion and echos as a result of the confined space of the compartment etc. Over the years however, huge improvements have been gained in classification algorithms. Technology evaluations, like the Face Recognition Grand Challenge (FRGC), have shown a huge progress in face recognition over the last 5-10 years (Phillips et al., 2005). When looking into the details, this progress was mainly driven by algorithmic innovations and improvement in sensor technology.

## METHOD

In this paper our goal is to recognize specific behavior based on the recognition (and tracking) of some features in the input data over time. The surveillance system can be divided into two general steps. The first step detects the features and events in the incoming sensor data and the second step uses automated reasoning to combine these events (over time) into activities and scenarios (figure 1). The reasoning method is based on expert knowledge gathered after interviews with security experts from the Dutch Railways (NS).

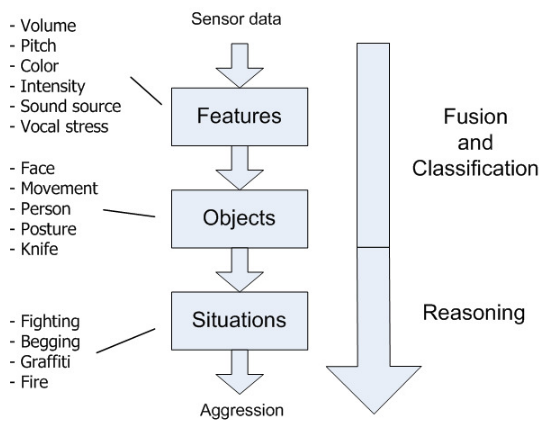


Figure 1: Overview of the aggression detection process

From interviews with experts we compiled a list of behaviours to detect and the features that the human experts use themselves to detect these behaviours. Next we gathered data of these behaviours in a train. Finally we used off-the-shelf classification algorithms to extract the features from the data and implemented our own algorithms to combine the detected the behaviours.

## Aggressive behavior

The Dutch Railways (NS) has a system to classify incidents that occur in a train. The NS tailors this classification toward the procedures that should be taken when an incident of a certain category occurs (see table 1).

Table 1: Incident categorisation used by the NS.

| Category | Description |
|---|---|
| A | Suspicious behavior |
| B | Robbery and theft |
| C | Violence |
| D | Serious public inconveniences |
| E | Small public inconveniences |
| F | Vandalism |
| G | Accident |
| H | Fire |

Based on this classification, we created scenarios to be performed by actors in a real train, trying to get at least one scenario per category. In this paper we will focus on these scenarios (listed below).

1 Suspicious behaviour: a passenger prefers to stand in an empty compartment. Features to watch for are: the compartment is empty or almost empty, a passenger stands in hallway, passenger does not move forward or backward.

2 Small public inconvenience: a beggar enters the compartment and starts asking for money. Features to watch for are: a passenger walking along the hallway stopping periodically and speaking (with normal volume) to passengers. The passenger does not take a seat.

3 Serious public inconvenience: overcrowding. The most important feature is the number of people in the compartment.

4 Ticket checking: a conductor enters the compartment and checks the tickets of the passengers. Features to watch for are: a person dressed in blue with a blue hat walks along the hallway stopping periodically and speaking (with normal volume) to passengers. The person receives an object from a passenger and gives it back after a while. The person does not take a seat.

5 Enter train: one or more persons enter the train. Features: People come into the train from the entrance doors. Some take a seat if there is a free seat available.

## Data

The aim of the data collection experiment is to gather data that can be used to test the aggression detection algorithms. Due to the scarcity of this kind of recordings and the privacy issues involved, we hired semi-professional actors to perform the scenarios described above in a real train. We used multiple microphones and cameras to record the actions. The location of the sensors in the train compartment and their orientation is shown in figure 2. Most scenarios were performed in the middle of the train, where the two cameras in the middle have the largest overlap.

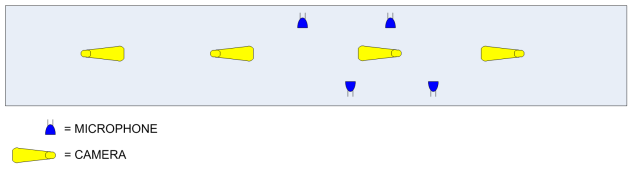


Figure 2: The locations of the sensors seen from a top view of the train compartment

The scenarios are recorded in sequences which total up to about one and a half hours of audio and video data. The data contains the scenarios as described earlier as well as recordings of normal and spontaneous situations. All the data of the sensors is stored in separate streams (four audio streams and four video streams). The four video cameras captured video at about 13 frames per second, at a resolution of 640x256 pixels (see figure 3).

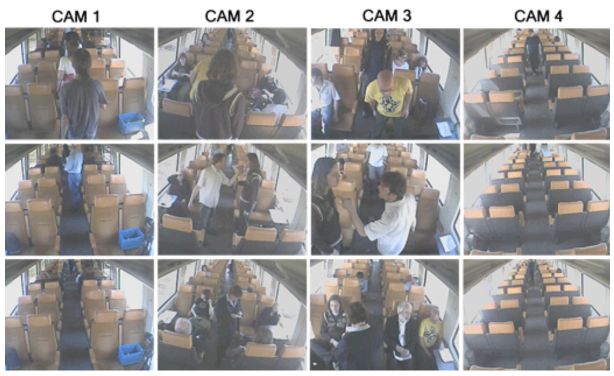


Figure 3: Each scenes as captured by the four cameras

Each microphone captured sound generated by the actors performing the scenarios at a sample rate of 44100Hz with a 24 bit sample size. Each track is synchronized in hardware with sample accuracy. The audio data can be addressed in a single synchronized project consisting of the four streams of the four microphones, or as separate mono audio streams for each individual microphone (figure 4).

## BEHAVIOUR RECOGNITION

Automated surveillance systems require the ability to recognize scenarios and behaviour from data. It is not sufficient to extract features and recognize objects since these have to be put in the correct context to determine the correct situation. For the scenarios we have defined earlier in this paper we have a list of features that need to be calculated at each time frame.

At each time step we determine:

- Number of people in the compartment
- Total movement (compared to the previous frame)

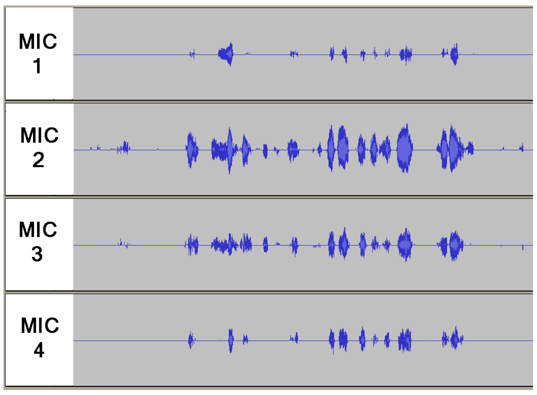


Figure 4: Four waveforms of a shouting scene recorded by the microphones. The waveforms are different in energy yet similar in form

- Total volume (over 100 ms)
- For each detected person position, pose and speed are determined.

By combining the feature vectors over time and using knowledge of the location of fixed objects in the train (such as seats), the behaviour of people in the train can be determined.

## Preprocessing

Our work differs from others by the fact that our system has to work under a more problematic setting. The challenging circumstances we have to cope with in train compartments include the varying (and relatively unpredictable e.g. snow, rain, tunnels) light conditions. The preprocessing step consists of reducing noise in the video stream.

The raw video data consists of a sequence of jpeg frames with a resolution of 640x256 pixels interlaced. Therefore, the true resolution of the images should be 640x512 pixels (a 4:3 aspect ratio). As the Voila & Jones frontal face detection algorithm was trained for larger faces (larger window size) than the faces that normally occur in our video images, we further upscaled the images during preprocessing. (The scaling factor was obtained by trial and error until the classifier performed well for a number of preselected test images from our dataset.)

The raw camera images recorded during the experiments in the train are not directly usable in classification algorithms. The camera is somewhat rotated causing horizontal lines to be slanted in the recorded images. Finally, the camera faces downward with an unknown angle, so that the images recorded are a perspective projection of objects in a 3-D scene onto a 2-D image.

The method for image adjustment is based on a camera model called the Direct Linear Transformation (DLT). The DLT model describes a model for camera calibration using a linear transformation that takes into account the zoom, pan, and tilt of the camera. The DLT method is a linear transformation so it is computationally cheap, but it is unable to compensate for non linear effects such as radial distortion.

The imaging process produced by the cameras can be interpreted as a sequence of three projective transformations. Given a point $p = (x_w, y_w, z_w, 1)$ in homogeneous world coordinates and a point $q = (f \cdot x_i, f \cdot y_i, f)$ in image coordinates corresponding to the projection of $p$ onto the image, the mapping of $p$ to $q$ can be expressed as:

$$q = K \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot M \cdot p \tag{1}$$

where $K$ represents the intrinsic parameters of the camera and is given by:

$$K = \begin{bmatrix} \sigma_x & \sigma_\theta & u_0 \\ 0 & \sigma_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2}$$

With $(u_0, v_0)$ the coordinates of the principal point, and $\sigma_x$ and $\sigma_y$ the scale factors in image $u$ and $v$ axes. The parameter $\sigma_\theta$ describes the skewness of the two image axes. In practice it accounts for the skewness due to non-rectangular pixels. However, in most cameras the pixels nowadays are almost perfectly rectangular and thus $\sigma_\theta$ is very close to zero.

$M$ represents the extrinsic parameters of the camera and is given by:

$$M = \begin{bmatrix} \ddots & \vdots & \iddots & \vdots \\ \dots & R & \dots & T \\ \iddots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

Where $R$ is the rotation and $T$ the translation which relates the world coordinate system to the camera coordinate system. Figure 5 shows the images before and after adjustment.

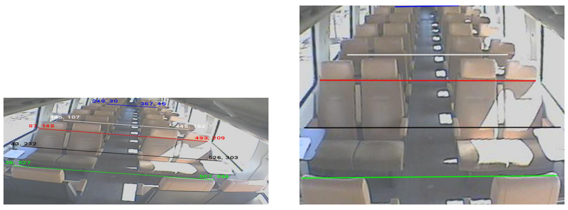

(a) (b)

Figure 5: Comparison of an original image from a train camera (a) with the same image after preprocessing (b)

## Face detection

The method we considered for the purpose of person detection is face detection. When a face is detected in an image, obviously this means a person has been detected as well. We used the face detection method implemented in the OpenCV library, which is based on the method proposed by Viola & Jones (Viola and Jones, 2001).

The Viola & Jones method is capable of accurately detecting faces for which the classifier is trained, in reasonable time. It is however not very robust under noisy circumstances, and the frontal face classifier used is very susceptible to changes in orientation. In larger frontal faces of sizes around 100x100 pixels, we achieved a detection rate close to the rates reported in the literature. However, if the size of a face drops below this figure, detection rates fall dramatically, to a point where they hardly contribute to detection at all. Faces down to a resolution of 16x16 pixels can be detected, however the reduced size of detectable features and the higher signal to noise ratios in these smaller areas result in a very high rate of false negatives and false positives. From the detected faces in the data almost half were false positives.

Given the number of false positives, determining the number of people by counting the number of faces, is inaccurate at the least. In addition, we did not have enough actors to capture data of an overcrowded train. At the peak of occupation, the train compartment was fairly crowded at most. To determine the number of people in the current frame more accurately, we first filter all positives from areas where no faces are expected to be found using a mask. This excludes areas such as the windows and the ceiling, where false positives commonly occur. We analyze the measurements of a limited number of frames up to the current frame. The theory is that the number of people in a scene will not change abruptly, but instead change gradually. If for example, in one frame we detect 4 faces, and none in the next, a scenario not uncommon with a low detection rate, we assume the scene to still contain 4 people.

## Action recognition

Our approach for activity recognition is by comparing the characteristics of the trajectory of people. We apply greedy nearest-neighbor matching to construct most probable tracks from the coordinates of detected faces. To guard against false positives, we apply a mask focused on the area around the corridor and the seats. To account for the low detection rate, we search over a maximum of 10 frames increasing the search area by 5 pixels every frame without a face found. This produces satisfactory results, due to the test data containing little occlusion of actors. Alternatively, other prediction methods, such as linear- and Kalman-filtering are widely used (Wang et al., 2003). The paths thus obtained (see figure 6) are compared to some predefined trajectory templates of actions such as entering the compartment and sitting, walking through the corridor, begging etc. The resulting measurement vector is compared to the template. The sum of the Euclidean distances between the current trajectory coordinates and each template trajectory coordinates is calculated and the action template with the smallest cumulative difference is selected.

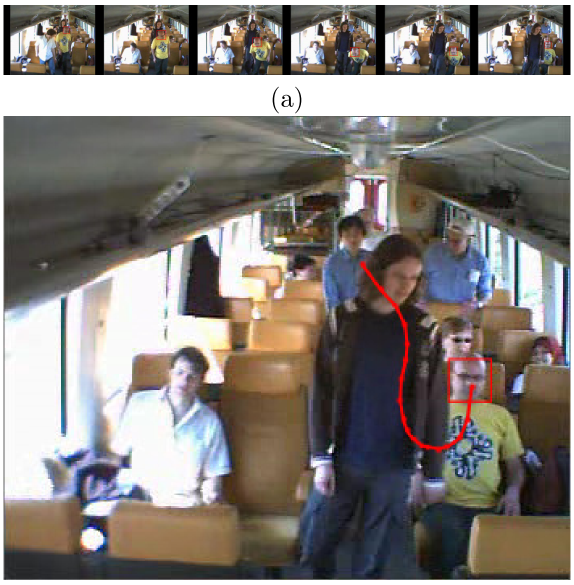


Figure 6: Individual frames with detected faces (a) and calculated path (red) of a person overlayed against a video image (b)

The results of the action recognition algorithm are somewhat disappointing at the moment. There are just a handfull of trajectories correctly recognized. This is partly due to the low detection rate and the high number of false positives. More importantly, we think the bad performance is caused by the way people walk. People tend to wobble while they are walking, this effect is amplified when people are walking near the camera. These sideway movements corrupted the speed measurements to such an extend that they where left out of the trajectory recognition algorithm. A solution would be better smoothing of the tracks or applying a normalisation measure determined by the distance to the camera. The distance to the camera can be determined by the size of a person's face or body. An additional benefit of working with distances is that positions can be translated into 3-D coordinates instead of the currently used position on the 2-D projection plane of the image.

## Behavior interpretation

Our goal is to define a simple set of rules based on interviews with experts, to recognize the predefined behaviours from observed data. Currently, the behaviour recognition is implemented as a rule based decision system that combines incoming features (number of people in the compartment, total volume, position, paths and speed of people in the scene etc.) into a conclusion.

The rule based system contains rules that describe the salient features of each scenario. As features are detected over time, these features are asserted into the rule base system as facts. Those rules with their features satisfied gain a higher score. If the score of a scenario reaches a certain threshold, that scenario is concluded to be the true scenario (figure 7).

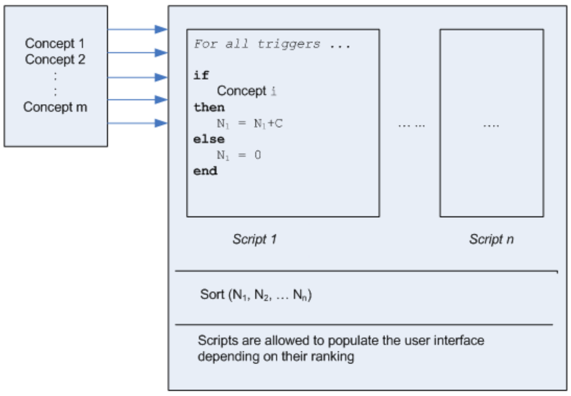


Figure 7: Reasoning scheme for behavior recognition

Features can be entered but also removed from the rule based system, making the system dynamic. To deal with uncertainty in the reasoning system, we will look into (dynamic) Bayesian networks. Since, it is not possible to model all scenarios and their particularities we plan to adopt techniques from emergent behaviour theory.

## FUTURE WORK

To improve the performance of face recognition, background modeling techniques could be used. Faces can only be detected in foreground pixels. Being able to reduce the search area to only the foreground pixels will greatly reduce the running time of a face detection algorithm. The simplest background model is taking the difference between two frames, and considering the previous frame to be the background. By using the produced motion history image (MHI) we can limit our search to the foreground pixels only. Since we have to deal with varying lighting conditions, this method is not expected to work well. A more robust method that is widely used is the median filter. This method takes the median value of a pixel over all the frames in the stream that have been detected until the current time index, and constructs a background image from that. Alternatively, if an image of the scene without people in it is available, we can use this instead to perform an offline background subtraction. Other methods rely on statistical analysis to construct a probability model of a single pixel.

Detecting skin tones in an image will effectively allow us to localize persons in a scene as well. Skin detection can be done in several ways, most of which are computationally inexpensive. The most basic method is simply determining for each pixel whether it belongs to an certain empirically determined color range, in this case that of skin tones. We can take advantage of the fact that the RGB values of skin tones are highly correlated for skin tones. The same holds for the YCbCr color space (Albiol et al., 2000). This correlation is quite specific, but not unique to skin tones. We experienced difficulties applying it in train compartments, specifically because the color of the upholstery was similar to skin color.

The key to differentiate between some scenarios depends on the recognition of certain salient objects or people. The conductor checking for tickets and the beggar scenarios for example can be differentiated by the detection of the conductor. Since conductors in the Netherlands wear specific uniforms, it is worthwhile to develop algorithms to detect the conductor specifically.

To deal with uncertainty in the reasoning system, we will look into (dynamic) Bayesian networks. Since, it is not possible to model all scenarios and their particularities we plan to adopt techniques from emergent behaviour theory. The idea is to have the scenario emerge as a completed puzzle from the detected features (puzzle pieces) instead of the fixed scenarios in the expert system approach.

## DISCUSSION AND CONCLUSIONS

In this paper we presented our work so far in the development of an aggression detection system for train compartments. Most of the work is still in a preliminary stage and many tasks need still to be done.

Nevertheless, we have managed to develop a prototype for simple behavior recognition in a train. We used off-the-shelf algorithms to detect low level features from data and we developed a high-level rule based reasoning system that combines the features into recognized behaviors. Rules of thumb used by security expert have been translated into rules for the reasoning system.

Since most of the classification algorithms that we used are trained (and meant) to work in lab environments, better fine tuning of the algorithms to suit the train compartment might improve results. For example, for person detection we used face detection. The downside of this method however is that only specific views of faces, such as frontal or profile, will yield positives. We suggest therefore that this method be used in conjunction with other person detection methods to increase the detection rate, as well as decreasing the likelihoods of false positives.

Currently the reasoning system is only capable of recognizing predefined scenarios from facts. We need to expand this with uncertainty and reasoning about unanticipated scenarios.

## ACKNOWLEDGEMENTS

This research was done as part of an ongoing project at the TUDelft funded by the MultimediaN project. We like to thank the NS/ProRail for providing us with a train to do recordings in and a train conductor to give advice.

## REFERENCES

Albiol, A., Torres, L., Bouman, C. A., and Delp, E. J. 2000. "A Simple and Efficient Face Detection Algorithm for Video Database Applications". In *Proceedings of the IEEE International Conference on Image Processing*, Vol. 2, pp. 239–242.

Beal, M. J., Attias, H., and Jojic, N. 2002. "Audio-Video Sensor Fusion with Probabilistic Graphical Models". In *Proceedings of the 7th European Conference on Computer Vision*, pp. 736–752, London, UK. Springer-Verlag.

Clavel, C., Ehrette, T., and Richard, G. 2005. "Events Detection For an Audio-based Surveillance System". In *the IEEE International Conference on Multimedia and Expo (ICME 2005)*, pp. 1306– 1309.

Cupillard, F., Avanzi, A., Brémond, F., and Thonnat, M. 2004. "Video Understanding For Metro Surveillance". In *Proceedings of the IEEE International Conference on Networking, Sensing & Control*, Taipei, Taiwan.

Foresti, G., Micheloni, C., Snidaro, L., Remagnino, P., and Ellis, T. 2005. "Active video-based surveillance system: the low-level image and video processing techniques needed for implementation". *IEEE Signal Processing Magazine*, Vol. 22 No. 2 pp. 25–37.

Härmä, A., McKinney, M. F., and Skowronek, J. 2005. "Automatic Surveillance of the Acoustic Activity in our Living Environment". In *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME 2005).*

Javed, O., Rasheed, Z., Alatas, O., and Shah, M. 2003. "Knight$^M$: A Real-time Surveillance System for Multiple Overlapping and Non-overlapping Cameras". In *Proceedings of the International Conference on Multimedia and Expo (ICME 2003).*

Oliver, N., Rosario, B., and Pentland, A. 2000. "A Bayesian Computer Vision System for Modeling Human Interactions". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 22 pp. 831–843.

Phillips, P., Flynn, P., Scruggs, T., Bowyer, K., Chang, J., Hoffman, K., Marques, J., Min, J., and Worek, W. 2005. "Overview of the Face Recognition Grand Challenge". In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2005)*, Vol. 1, pp. 947–954.

Velastin, S. A., Maria Alicia Vicencio-Silva, B. L., and Khoudour, L. 2002. "A Distributed Surveillance System For Improving Security In Public Transport Networks". *Special Issue on Remote Surveillance Measurement and Control*, Vol. 35 No. 8 pp. 209–13.

Viola, P. and Jones, M. 2001. "Rapid Object Detection using a Boosted Cascade of Simple Features". In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognitio (CVPR 2001)*, Vol. 1, pp. I–511– I–518.

Wang, L., Hu, W., and Tan, T. 2003. "Recent Developments in Human Motion Analysis". *Pattern Recognition*, Vol. 36 No. 3 pp. 585–601.

Zhang, D., Gatica-Perez, D., Bengio, S., and McCowan, I. 2006. "Modeling Individual and Group Actions in Meetings With Layered HMMs". *IEEE Transactions on Multimedia*, Vol. 8 No. 3 pp. 509–520.