Recognizing Postures and Head Movements from Video Sequences

Jacob Knagenhjelm[1] and Adam Olsson[2]

June 21, 2016

Department of Mathematical Sciences
Gothenburg University[1] and Chalmers University of Technology[2]
Master's Thesis in Mathematical Statistics[1]
Master's Thesis in Engineering Mathematics and Computational Science[2]

Recognizing Postures and Head Movements from Video Sequences
JACOB KNAGENHJELM AND ADAM OLSSON

Department of Mathematical Sciences
Chalmers University of Technology and Gothenburg University
SE-412 96 Gothenburg
Sweden
Email gusknagka@student.gu.se, adamol@student.chalmers.se

Recognizing Postures and Head Movements from Video Sequences
Jacob Knagenhjelm and Adam Olsson
Department of Mathematical Sciences
Chalmers University of Technology and Gothenburg University

Abstract

Video analysis is a fast growing field of study, with applications ranging from surveillance systems to face detection. Volvo Group Trucks Technology wishes to apply some of the innovative solutions discovered in recent times to their own study data to minimize the need of manual video processing. This master's thesis, is an initial step in this direction.

Video data of drivers driving in a virtual environment has been gathered from frontal and side view. The frontal view data has been used to estimate gaze direction of the driver while the side view has been used to identify posture.

Gaze has been estimated mainly using the popular Viola-Jones face detection algorithm combined with extracting feature points which are tracked frame to frame using the Kanade-Lucas-Tomasi feature tracker. For posture identification, marks were placed on the side of the driver and tracked using the Kanade-Lucas-Tomasi feature tracker. Training data was used to create clusters using the k-medoids method and driver postures were identified using mark coordinate data from sample videos using mainly a k-nearest-neighbors approach.

Results are promising both for gaze estimation and posture identification even though ambiguity in matching continuous data to discrete evaluations lead to a noticeable amount of errors.

Acknowledgements

This master's thesis would not have been possible without the assistance from our supervisor Mats Rudemo at the department of mathematical sciences at Chalmers University of Technology, as well as our contact at Volvo Group Trucks Technology, Dennis Saluäär, and the rest of the team at Volvo GTT. It has been a pleasure to work along side them.

Contents

1 Introduction

Interactive driving simulators serve as important tools in analyzing the interaction between drivers and vehicle systems in a safe and controllable way. Volvo Group Trucks Technologies (Volvo GTT) use simulation technologies in order to perform early concept development. The simulators register variables such as traffic and vehicle condition (speed, steering and braking). In order to study the interaction between the driver and the vehicle systems, a video of the driver is recorded. The video recordings are analyzed manually in order to study different interactions between driver and vehicle depending on the study at hand.

1.1 Background

In the study analyzed during the course of this project, the situations of interest were those which caused the driver to change his or her posture. There were seven predefined postures for the driver and each time there was a change in the posture, the traffic condition was recorded. This work was very time consuming and it was therefore desirable to develop a tool for automatic analysis of the video recordings to apply to similar studies in the future. Furthermore, analysis of the gaze direction has been made in order to estimate whereto the drivers were directing their attention. Even though this was not part of the original study, it was deemed desirable for future use.

It is worth noting, that during the course of the project Volvo has been working on changing their methods for data gathering, and also the equipment used in doing so. The final results in this report will be focused more on these new methods and equipment since it was considered the most usable for Volvo and future development.

1.2 Purpose

The aim of this thesis is to create MATLAB algorithms which go through video data, identify different driving postures and times when the driver changes posture. Head movements will also be detected, specifically rotations as an estimate of the direction of the drivers primary attention.

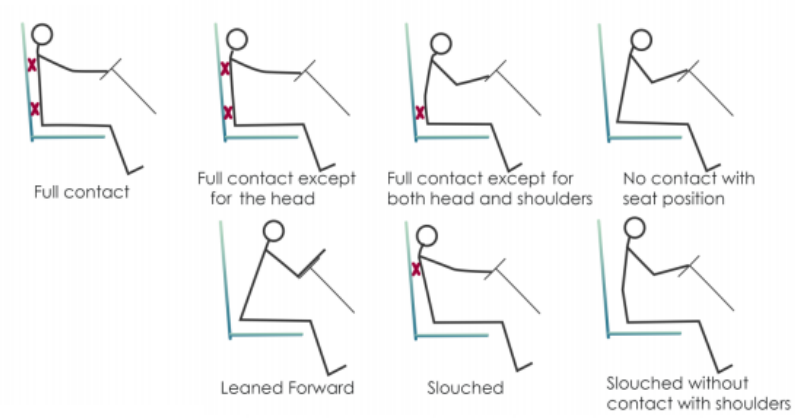

Figure 1: The seven driving postures of interest, taken from [22].

The data used was originally taken from the study mentioned above, where annotations had been created manually regarding the postures of the drivers. However, this data was flawed in that there was no screen behind the driver, allowing for background movement to interfere with the analysis, the video was in gray scale which restricted the use of color based tracking for face detection and the videos were not synchronized between the different angles.

These factors led us to gather new data in color, with synchronized cameras, with little to no background movement which could be more easily analyzed. Seeing as these are all measures that Volvo GTT can take in future studies, there was no reason not to gather this new data and use that instead of the initial data.

1.3 Scope

Posture identification will be restricted to seven predefined postures with the aim of achieving similar analysis to what has been done manually using the data from the study. In the study data, a single mark on the shoulder will be tracked for identification, whilst in analyzing the new data there is in addition a mark in the lower back region as well as middle back region to facilitate identification of postures more clearly.

Face detection will be restricted to using the Viola-Jones cascading classifier [12] for the study data since both color extraction and background subtraction run into major difficulties seeing as the data is grayscaled and there is motion of the people conducting the study in the background. When analyz-

ing the new data, skin color will be extracted in order to detect the face more fluidly during rotation, which otherwise runs into problems simply using the Viola-Jones algorithm.

Only one view will be used for each movement identification. The side view will be used for posture identification and the frontal for gaze estimation, both when studying the initial data and the new data.

2 Previous Work

The estimation of driver posture and gaze direction requires two main steps. Detection of the driver (object detection) and tracking of the motions from frame to frame (object tracking). This section will review some of the work that has previously been done in these (somewhat overlapping) areas.

2.1 Object Detection

Objects in images can be detected in different ways, for example, moving objects can be found through differences between frames [15]. Other methods use properties of the objects such as color and shape [1] [2].

2.1.1 Motion-Based Tracking

Background subtraction refers to a family of motion detection methods that make use of a background image in order to find objects that are moving through the image as the difference between the current frame and the background image. The difference between methods within this family lies in the way the background is modeled. Benezeth et al. presents a comparison of different background models in [15]. The methods are compared with respect to both accuracy and computational cost. The simplest model uses a static background image (fails when the background changes over time, for example due to illumination changes) while some of the more complicated ones model each pixel with some probability distribution (more computationally expensive). The authors find that there is no clear answer as to which method is best - it depends on the situation. However, in many cases a background model that estimates the probability of each pixel belonging to the background using a single Gaussian distribution is sufficiently accurate without being overly expensive.

2.1.2 Color-Based Tracking

Object detection can also be performed by analyzing the colors in the image. In [2] Shaika et al. use a couple of different color spaces in order to identify objects of skin color. The authors find that the standard RGB color space is not the best color space for skin color segmentation and attribute this to the fact that in the RGB space, the chrominance and luminance components are mixed together. Instead it is suggested that the hue and luminance based color spaces HSV and YCbCr are used. In these color spaces the color and the intensity channels are separated even when the image illumination is uneven.

2.1.3 Face Detection

A lot of work in computer vision has been done in the area of face detection. Ghimire and Lee present a method based on a combination of color and edge detection in [1]. The color segmentation is performed in two different color spaces; luminance corrected RGB, and YCbCr. The luminance enhancement is done by converting the RGB image into its HSV representation and applying a nonlinear transformation to the Value channel before converting the image back to RGB space. The results from the segmentation in the two color spaces are then multiplied together for a combined result. Finally the authors use edge detection in the original image in order to separate blobs of skin color that may have falsely been connected.

Perhaps the most commonly applied method for face detection found in the literature is the method proposed by Viola and Jones in [12]. Using a set of simple Haar features they are able to build a very quick and accurate face detector. Using a cascade of weak classifiers the proposed algorithm can quickly discard regions of the images that do not contain a face and move on to regions that may contain a face. The cascade of weak classifiers then build up to a strong classifier. One weakness of this method is that it is sensitive to head rotations, with performance decreasing greatly even for fairly small rotations. With this in mind Viola and Jones extended the method to include features for profile view faces in [13]

2.2 Object Tracking

Lucas and Kanade present an image registration technique for point tracking (KLT-tracker) in [9]. This tracker is applied in combination with a Kalman filter in [10] in order to track human upper body limbs (head, arms and forearms). The KLT tracker is applied to the different limbs and the Kalman filter is used to predict and estimate the best tracked path. The results are quite good, especially for the tracking of the head.

2.2.1 Choosing Features

In order for the tracker to work well it is important to choose features that are easy to track. In [3] a matrix containing pure texture information is created and the eigenvalues of this matrix are analyzed in order to determine if the region can be tracked. If both eigenvalues exceed a certain threshold there is sufficient texture in the region for tracking.

2.2.2 Head Movement Estimation

A lot of work studying the head posture and movements of drivers has been done in relation to detection of driver drowsiness [4] [5] [6]. In [4] Teyeb et al. apply the Viola-Jones algorithm in order to detect the drivers face in each frame and use the properties of the bounding rectangles in order to determine the position of the head. The authors are able to find several different head positions including left and right rotation as well as left, right, upwards and downwards inclination. In this study the first version of the Viola-Jones algorithm was used so the results were only accurate when the head rotation was relatively small.

Instead of following the development of the bounding box, Wagener and Herbst suggest looking for features to track inside the region identified as a face [3]. It is assumed that the face region is already detected and the features that are tracked are based on the texture of the image in the face region. More specifically the image gradients in the x- and y-directions are calculated in order to detect regions with large changes in image intensity. These regions are then tracked using the KLT algorithm. A problem that the authors experienced was that as the head rotates some parts of the face that may have been identified as regions to track can become occluded and tracking therefore lost. The solution to this problem was to find a new feature to track within the region identified as a face.

2.2.3 Eye Extraction

Extracting the eyes of the driver is obviously something that is useful in the estimation of gaze direction. In [5] different methods for eye extraction are compared. Some of them are extensions of face detector techniques such as the Viola-Jones method, others use the fact that the eye is shaped like an ellipse or look at the intersection of gradients. The Viola-Jones method was found to be reasonably accurate and very fast, however, with a few false positives. The method that most accurately found the centers of the eyes was one looking at the intersection of gradients, the drawback with this method is that it is slower.

3 Data

Two different sets of data were used in this thesis work. One was recorded for a previous study and one was recorded during the thesis work. The previously recorded data had the advantage that it had been analyzed so that there was a ground truth for driver postures available, however the video quality was quite poor which made the analysis difficult. Since Volvo GTT were about to change their data gathering for future studies and update their hardware, new data was gathered and analyzed.

3.1 Study Data

The first data set was recorded using surveillance cameras from 4 different angles. One camera was positioned in a frontal view, one from the passenger side, one from above and one viewed the chair to detect changes in position of the chair. The video was recorded in h.264 format which needed to be converted into .avi in order to process the data using MATLAB. All drivers wore white long-sleeved shirts and had black marks on the side of each shoulder as well as on top of each shoulder, in order to assist in image processing.

Figure 2: Video data taken from a previous study where manual annotations had been made noting the driving posture of the driver.

Ten drivers were asked to drive along a predefined track in a simulation environment. The driving was a mixture of city terrain and highway. After the data was gathered, manual analysis of the data was performed and time stamps for change of posture was noted. The idea behind the test was to see under which circumstances drivers change their driving posture and compare

the results between simulator and real world driving.

The video format created some issues with converting and quality was worse than desirable. Furthermore, the video sequences were recorded with a dynamic frame rate. If there was little activity in the camera view, the frame rate decreased and when there was activity, the frame rate increased. This made it difficult to say when an event occurred in real time.

The people conducting the study were freely moving around in the background of the frontal video since at the time they thought they would analyze the video manually. However, this caused issue with our automated analysis of the gaze estimation since there was no screen blocking off the background noise of people which were not of interest.

Lastly, the data was recorded without color which excluded some promising methods of face tracking which are based on color.

3.2 Second Data Sample

The new data was recorded using GoPro cameras. The videos were recorded in color and at a constant frame rate of 25 (frames per second) from the side view and 30 (frames per second) from the frontal view. The cameras were synchronized to start at the same time after connecting them with a remote control.

Because of lack of time to prepare a new study, we, the authors of this paper, as well as our Volvo contact Dennis Saluäär, were the drivers for this new test. Worth noting is perhaps that neither of these drivers were experienced truck drivers.

Figure 3: Video data gathered during the course of the thesis with GoPro cameras.

The drivers wore a similar type of long sleeved t-shirts as in the previous study, however markers were placed not only on the shoulder as before, but now also on the side of the torso (see figure). A dark screen was placed behind the driver in order to reduce the risk of detecting background movement.

Two different types of videos were recorded. First the drivers were asked to sit in each predefined posture for three seconds and then move on to the next one. This was done in order to ensure that there would be data available for each posture to allow training. Second, a driving video was recorded, this time driving was only done in city terrain. This was in order to get as much movement of the drivers as possible to see if the posture and gaze direction changes could be found. The simulation in question had several traffic lights turned off to assure that the driver had to adjust his position and look around and couldn't simply trust the traffic light to allow him to pass through an intersection.

4 Theory

4.1 Morphological Operations

As most images are not perfectly segmented through thresholding, the obtained binary image usually contains some noise. Morphological operations are used to remove noise from binary images [16]. The most basic morphological operations are erosion and dilation, which extend to opening and closing.

Erosion removes noise and irregularities through the use of a structuring element, for example a square of 3×3 pixels with the center pixel as the reference pixel. The erosion is defined as follows:

$$A \ominus S = \{(i, j) : S_{i,j} \subseteq A\} \tag{1}$$

Where A is the set of white pixels and $S_{i,j}$ is the structuring element with reference pixel i,j. Dilation is defined as:

$$A \oplus S = (A^c \ominus S)^c \tag{2}$$

So a dilation of the white pixels is basically an erosion of the black pixels. Opening and closing of images are combinations of erosion of dilation defined as:

$$Opening : \psi_S(A) = (A \ominus S) \oplus S' \tag{3}$$

$$Closing : \phi_S(A) = (A \oplus S) \ominus S' \tag{4}$$

Where S' is the structuring element rotated 180°. Opening of an image is a erosion followed by a dilation. The erosion removes noise in the form of scattered single and small groups of white pixels, but also some of the white pixels on the edge of the desired object. Dilation restores the falsely removed pixels. Definitions and equations in this section are from [16].

4.2 Viola-Jones Face Detection

When creating a face detector, one uses a large amount of images, some of faces and some non-faces. These training images are input into a computer system and various features are calculated for each image. The computer is told which images are faces and which ones are not. Using this knowledge the system can be taught to detect faces using the feature values it calculates over the different face images and non-face images. Once the system is trained

in this manner, one can input an unknown image and based on the feature values for this image, the system would classify it as being a face image or not.

Viola and Jones [12] created a novel approach to detect faces which has since become standard and implemented in for instance OpenCV, which has support in several different languages. They proposed a system of face detection based on Haar features which were calculated quickly using integral images. These were used to train a strong classifier using Adaboost which was cascaded over images to quickly detect faces and disregard non-faces.

4.2.1 Haar Features

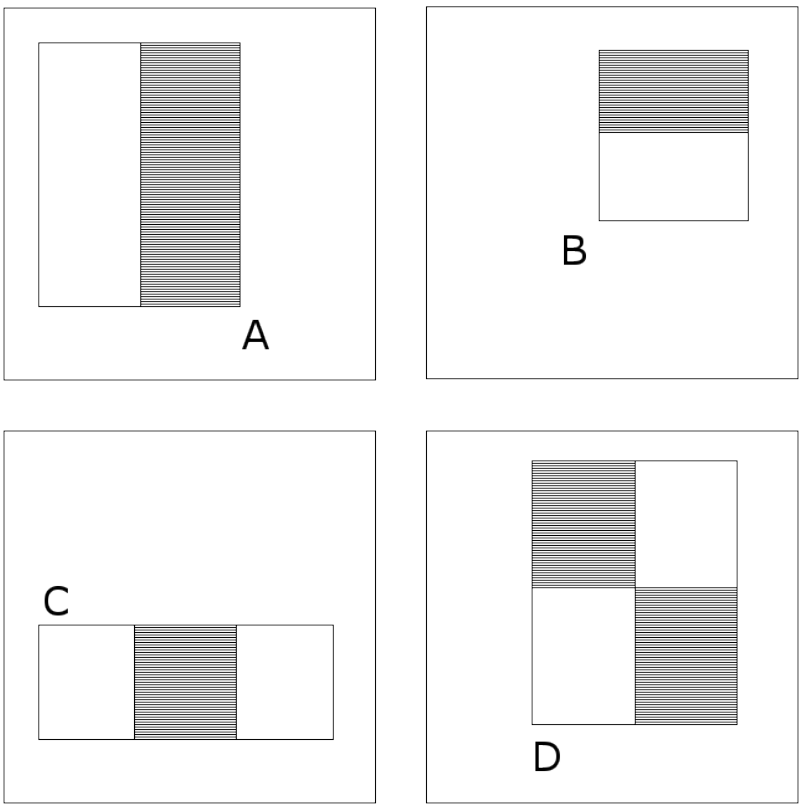

Figure 4: The feature types used by Viola and Jones. The features were varied in size and moved over all areas of both face and non-face images. The feature value was calculated by summing the pixels in the dark areas and subtracting by the pixel sum over the white area. Image taken from [19].

During the training process, Viola and Jones used so-called Haar features (name from their similarity to Haar wavelets) which were different rectangular compositions consisting of white and black areas. The features were varied in size along all possible areas and run over 24×24 images of both faces and non-faces. In total, when considering the possible sizes and position of the features, they had over $160'000$ features to test on each 24×24 image. Each feature consisted of taking in the pixel values in the part of the frame it was placed, summing the pixels in the dark area and subtracting by the pixel sum over the white. This difference of sums produced a single value which was the output of the feature.

4.2.2 Integral Images

The feature calculations consisted of taking sums over pixels values and then calculating the difference between regions. This had to be done for over 160'000 features on each image and it had to be done over hundreds of thousands of images. This would be a very calculation heavy task which would take a large amount of time. A solution to this problem, which Viola and Jones presented in their original paper, made this process a lot more efficient. This was the use of so-called integral images.

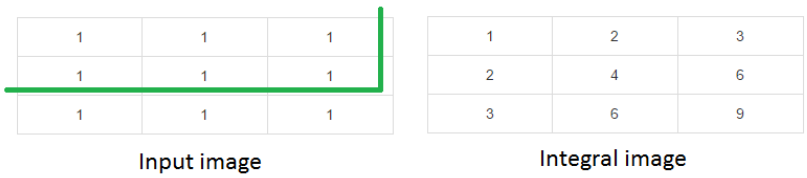

Figure 5: The integral image matrix is the sum of the entries above and to the left of that pixel coordinate in the original input image, including the pixel coordinate value itself.

The integral image is described by the function

$$I(x, y) = \sum_{i \leq x} \sum_{j \leq y} p(i, j), \tag{5}$$

where $I(x, y)$ is the value of the integral image at coordinate (x, y) and $p(i, j)$ is the pixel count at coordinate (i, j).

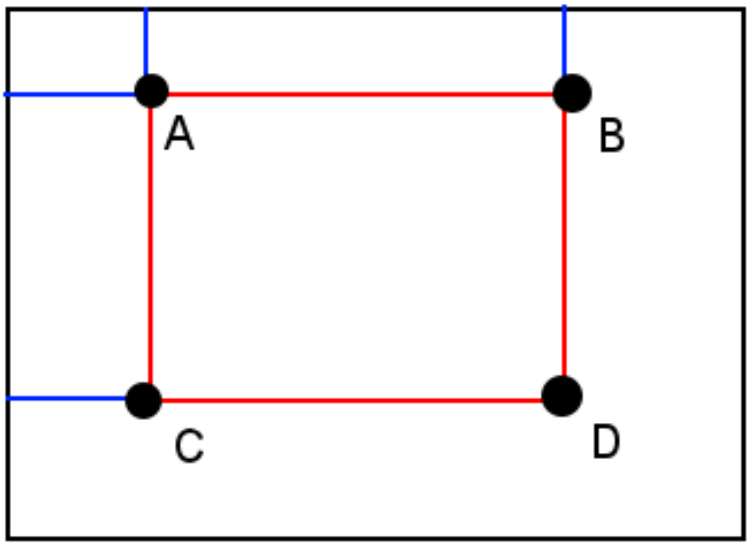

Figure 6: The sum of pixels within rectangle $ABDC$ can be calculated using the values of the integral image at A, B, C and D. Image taken from [21].

Using the integral image, one can quickly calculate the pixels inside a rectangle by a simple sum. Denoting the pixel sum inside D (see figure) by $P(ABCD)$ and the integral image values at A, B, C, D by $I(A)$, $I(B)$, $I(C)$, $I(D)$ respectively, one easily finds that

$$P(ABCD) = I(D) + I(A) - [I(B) + I(C)]. \tag{6}$$

4.2.3 Adaboost

Adaboost was used to select a subset of the original 160'000 features so that only a lesser amount of them had to be checked in order to determine which images contained faces.

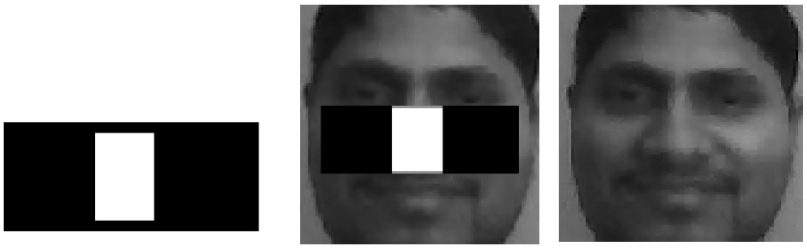

Figure 7: An example of how the features are used by Viola-Jones. The nose bridge region is lighter than the eyes. The current feature placement would not give a remarkable difference to non-face images, but moving it up to the eyes would. Image taken from [19].

Figure 8: Another example of the feature usage. The eye region is darker than the upper-cheeks. Image taken from [19].

For instance, the eye region is darker than the upper cheeks. This means that when the feature that is the difference of sums of pixels between these two regions is calculated, a high value will be found in face images. Another typical example is over the nose bridge which is lighter than the eyes. When the correct feature calculates this difference a large value will be found, which can be used to differentiate between face images and non-face images. So these features would output high values for face images, but not necessarily high values for non-face images. This means that they are typical examples which Adaboost would choose.

If one instead considered a feature consisting of checking only the area on one cheek for example, this would generally not produce anything significant, neither in face images or in non-face images. Hence this feature would be discarded by Adaboost.

After Adaboost chose the relevant features, these were given weights in order to together build up to a single strong classifier. Let x be a given image, $F(x)$ denote the strong classifier, $f_i(x)$ the features for $i = 1, ..., N$ where N is the number of total features chosen by Adaboost and α_i the respective weights, then

$$F(x) = \sum_{i=1}^{N} \alpha_i f_i(x). \tag{7}$$

It is in actuality this strong classifier which determines if a given image is a face image. The output of the weak classifiers (f_i) above is either 1 or 0 depending on if it classified the image as a face or non-face and some threshold for F is set to determine what is accepted for the image to pass as a face image.

4.2.4 Cascading

In every 24×24 image, it is necessary to check the N features from Adaboost. If checking an image of say 640×640, a 24×24 section is moved over the image and in each section all the N features are calculated. If the strong classifier exceeds the set threshold it is determined that a face is found in this region. Instead of running all the N features in each 24×24 part of the image, Viola and Jones used cascades.

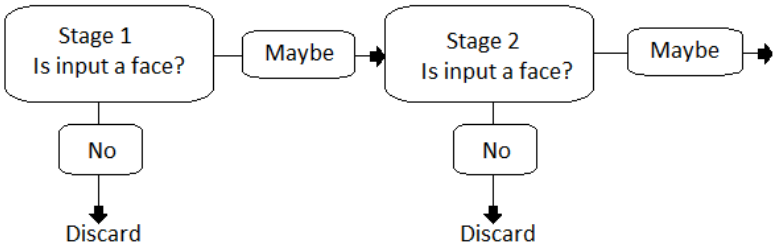

Figure 9: If the section passes through one stage of the cascade it continues to the next. If it doesn't, the section is discarded and the scan moves on to the next section of the image.

The N features are divided into different stages. The first n_1 are grouped together in the first stage, the following n_2 in stage two, etc. Then, when a section of an image is tested, each stage is tested in order. If the section passes through the current stage it moves on to the next until all stages are accepted. If one stage fails, the section is disregarded and the scan moves on

to the next section.

The strength of this approach is the increased speed of not having to check all features for every part of an image, but rather, quickly disregarding non-face areas and moving on. Seeing as most parts of an image from say a surveillance camera do not contain any faces, this increases the speed of the scan by a tremendous amount since almost the entire frame consists of non-face sections.

Worth noting is that after the original 24×24 section is scanned over an image, larger section areas (increasing by a factor of 1.25 in each iteration) are also checked and the image is repeatedly scanned over for each section size.

4.3 Kanade-Lucas-Tomasi Feature Tracking

In 1981 Bruce D. Lucas and Takeo Kanade published a paper [9] describing how they had created a new image registration technique. Several formulas in this section are taken from [9].

The main idea behind the technique was to define functions $F(x)$ and $G(x)$ for two images respectively and find the vector h which minimized the distance between $F(x + h)$ and $G(x)$ within some region of interest R around the initial point x.

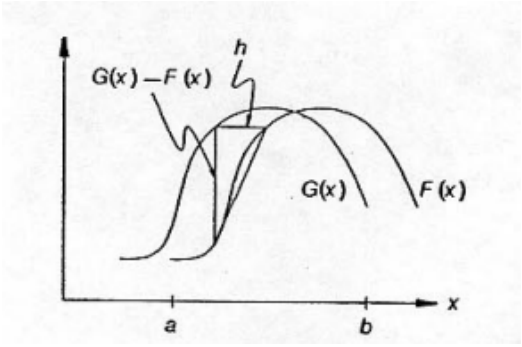

Figure 10: Illustration of KLT in the one dimensional case. Image taken from [9].

The one-dimensional case of this problem can be illustrated by the picture in Figure 10 above. A natural approach to estimating the value of h for least

discrepancy in this case, would be linear approximation. We know that

$$F'(x) \approx \frac{F(x+h) - F(x)}{h} \approx \frac{G(x) - F(x)}{h}, \tag{8}$$

which gives

$$h \approx \frac{G(x) - F(x)}{F'(x)} \tag{9}$$

Naturally, the approximation using the derivative 8 requires h to be reasonably small. If more than one value for x is of interest, the h vector is calculated for each and averaged using individual weights. The approximation 8 is good when F is approximately linear, hence the weights should depend on the linearity of $F(x)$.

Seeing as the second derivative of a function is a measure for linearity, using the inverse of

$$\frac{G'(x) - F'(x)}{h} \approx F''(x), \tag{10}$$

as the weight is a natural choice. Hence the weight function depending on x can be set as

$$w(x) = \frac{1}{|G'(x) - F'(x)|}, \tag{11}$$

where the h factor has been dropped since it will be used for averaging anyway. Combining the above, one gets

$$h = \frac{\sum_x \frac{w(x)(G(x)-F(x))}{F'(x)}}{\sum_x w(x)}, \tag{12}$$

and after calculating h, one moves x to $x + h$ and repeats the process.

However, this linear approximation method has problems when $F(x)$ is level since the derivative which is divided by is zero and it is hard to generalize to higher dimensions with the gradient. Kanade and Lucas corrected this issue using the Taylor series approximation $F(x + h) \approx F(x) + hF'(x)$ and searched for the h which minimized the L_2 norm between $F(x+h)$ and $G(x)$, i.e. $E = \sum_x (F(x + h) - G(x))^2$.

Minimization was calculated by finding where the differential was zero

$$
\begin{aligned}
0 = \frac{\partial E}{\partial h} &= \frac{\partial}{\partial h}\sum_x (F(x+h) - G(x))^2 \approx \\
&\approx \frac{\partial}{\partial h}\sum_x (F(x) + hF'(x) - G(x))^2 = \\
&= \sum_x 2F'(x)(F(x) + hF'(x) - G(x))
\end{aligned} \tag{13}
$$

Moving over the h term and dividing by its factor, one gets an approximation

$$
h \approx \frac{\sum_x F'(x)(G(x) - F(x)}{\sum_x F'(x)^2}. \tag{14}
$$

Note that this is the same as the previously derived approximation, only now the weight function $w(x)$ is equal to $F'(x)^2$. Note that there is still a chance of dividing by zero, but only if $F'(x)$ is zero everywhere, whereas before division by zero occurred as soon as $F'(x)$ was zero anywhere. Furthermore, it is now possible to generalize to higher dimensions since the gradient is squared.

When constructing the generalization to higher dimensions, one still wishes to minimize the L_2 norm of the error

$$
E = \sum_x [F(x+h) - G(x)]^2 \tag{15}
$$

just that now x and h are vectors of dimension n. The Taylor series analogous to earlier becomes $F(x+h) = F(x) + h\nabla F(x)$. Inputting this in the above and differentiating with respect to h in order to minimize E one gets

$$
\begin{aligned}
0 = \frac{\partial}{\partial h}E \approx \frac{\partial}{\partial h}\sum_x [F(x) + h\nabla F(x) - G(x)]^2 = \\
= \sum_x 2\frac{\partial F}{\partial x}^T [F(x) + h\frac{\partial F}{\partial x} - G(x)] \\
\Rightarrow h \approx [\sum_x \frac{\partial F}{\partial x}^T [G(x) - F(x)]][\sum_x (\frac{\partial F}{\partial x}^T (\frac{\partial F}{\partial x})]^{-1},
\end{aligned} \tag{16}
$$

which is very similar to what was found in the one dimensional case.

Note that the above discussion and derivation has been made in regards to translation of objects. Naturally an object can move in more ways than

translation - rotation, affine transformation and projection being other alternatives. In these cases one no longer searches for the minimizing h between $F(x+h)$ and $G(x)$. Instead a matrix A is added to express these further possible transformations and search for the error

$$E = \sum_x [F(xA+h) - G(x)]^2. \tag{17}$$

Also in this case, Taylor series are used to approximate the deviation of $F(xA+h)$ from $F(x)$, however now it is done in the A and h variables. Solving for this system as above, would now give a set of linear equations which need to be solved simultaneously.

4.4 Meanshift

The mean shift algorithm works by taking an initial point in some space of sample points and an initial circle radius. Within this radius the average centroid value of points within the circle is calculated by

$$M_h(y) = \frac{1}{n}\sum_{i=1}^{n} x_i - y_0 \tag{18}$$

where n is the number of points x_i within the circle, and y_0 is the initial point.

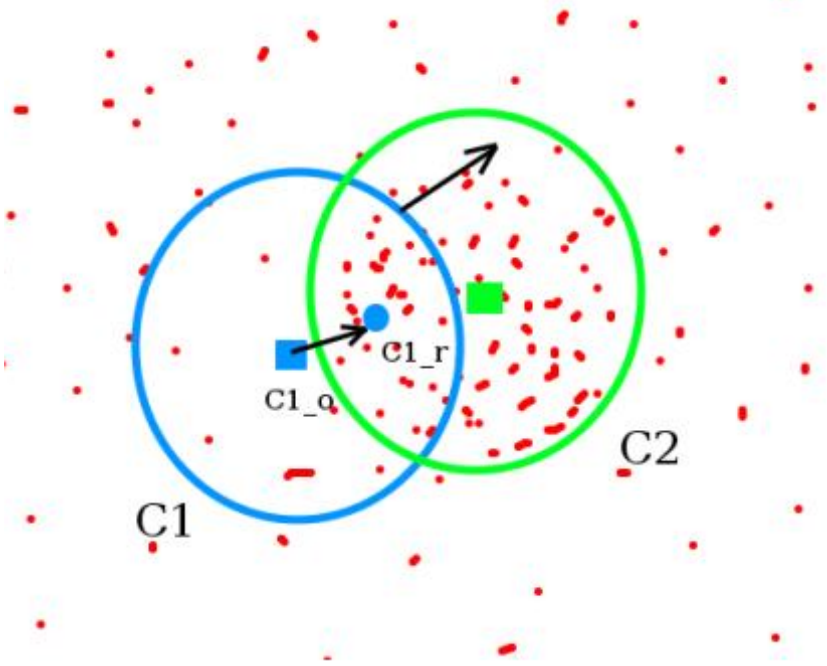

Figure 11: An initial point C1_o is chosen with some radius. Within this radius the mean is calculated (C1_r) and the center point is shifted to this new point. The iteration will eventually converge to the circle C2. Image taken from [20].

Basically, the mean is calculated and then one shifts the center to this new point. This newly found coordinate is taken as a new center point and the process is repeated with the same radius as before. In this manner, one eventually converges towards the center of density of the data points.

One can also weight the mean shift algorithm in different ways in relation to the distance to the initial point y_0

$$M_h(y) = \frac{\sum_{i=1}^{n} w_i(y_0)x_i}{\sum_{i=1}^{n} w_i(y_0)} - y_0. \tag{19}$$

These weights can be chosen according to different distributions. Gaussian is a typical choice, while a Uniform distribution would yield the original equation. Another example of a distribution is a histogram, which would produce a one dimensional data sample (as opposed to the two dimensional illustrated in the figure above). Given an image with N frames, one can divide the possible pixel values of $0 - 255$ into n bins. Each bin then gets the value of the number of pixels within the specified region. Lastly, the bin values are

averaged so that they sum to 1.

In implementation, when tracking a model from one frame to the next, one first chooses a target model to track in the initial frame. Then a feature space (RGB, YCbCr, HSV, etc) and feature is chosen. A region of interest around the initial model is chosen (depending on how far the model is predicted to have moved). The target model is described by a vector $q = \{q_1, ..., q_n\}$ such that $\sum q_i = 1$ where q_i are the bin values of the histogram from the chosen feature.

Lastly the most similar candidate model within the region of interest in the next frame, is found. The candidate models are described by vectors $p(y) = \{p_1(y), ..., p_n(y)\}$ where y is a localization vector which differentiates between the different parts of the frame within the region of interest.

The candidate models are then compared with the target model by using the Bhattacharyya coefficient which is essentially the dot product between the models

$$\rho(p(y), q) = \sum_{i=1}^{n} \sqrt{p_i(y) q_i}. \tag{20}$$

One easily sees that the distance d between the models is $d(y) = \sqrt{1 - \rho(p(y), q)}$ and hence maximizing the Bhattacharyya coefficient is the same as minimizing the distance between the models.

Applying Taylor series expansion to ρ around $p(y_0)$ one finds that

$$\rho(p(y), q) \approx \rho(p(y_0), q) + \frac{1}{2} \sum_{i=1}^{n} p_i(y) \sqrt{\frac{q_i}{p_i(y_0)}} \tag{21}$$

Now, we know that $p(u) = C \sum_{x_i \in S} k(||x_i||^2) \delta(S(x_i) - u)$ where $S(x_i)$ is the color at x_i, δ is the delta function, and k is the kernel of the distribution. Using this substitution, the second term of the right hand side in the equation above becomes

$$\frac{C_h}{2} \sum_{i=1}^{n_x} [\sum_{u=1}^{m} \delta[S(x_i) - u] \sqrt{\frac{q_u}{p_u(y_0)}}] k(||\frac{y - x_i}{h}||) \tag{22}$$

After setting $[\sum_{u=1}^{m} \delta[S(x_i) - u] \sqrt{\frac{q_u}{p_u(y_0)}}] = w_i$ one sees that it is possible to apply the mean shift algorithm in order to maximize these weights

$$M_h(y) = \frac{\sum_{i=1}^{n_x} w_i(y_0)x_i}{\sum_{i=1}^{n_x} w_i(y_0)} - y_0. \tag{23}$$

Several equations in this section was taken from [23].

4.5 Background Subtraction

In order to analyze a video sequence it is necessary to extract the object of interest or foreground in each frame of the video. There are several ways of doing this, one general method is called background subtraction. There are several different versions of background subtraction, differing in accuracy and computational cost. The general idea is to compare the current frame to some background image and calculate pixel wise differences using some distance measure. Pixels that differ greatly in intensity compared to the background are considered foreground pixels. This general idea of the motion mask $\chi_t(s)$ is described in [15] as follows:

$$\chi_t(s) = \begin{cases} 1 & \text{if } d(I_{t,s}, B_s) > \tau \\ 0 & otherwise, \end{cases} \tag{24}$$

Where τ is a threshold, d is a distance function of the intensities of the image $I_{t,s}$ at time t and pixel s and the background B_s at pixel s. The different models for background subtraction differ mainly in the way that B is modeled and in the distance function d [15]. A binary image can then be created by setting the pixels identified as foreground pixels to one and the background pixels to zero.

Perhaps the simplest way to perform background subtraction is to use a fixed image of the background, with no foreground objects present. However this method is obviously not robust to changes in the background during the course of the video sequence. Changes in illumination, occlusion and objects having the same color as the background can cause problems [18]. Another method is to compare each frame to the previous one. This method is more robust towards changes in illumination but only parts of the moving object will be detected [15].

There exists several statistical models for updating the estimation of the background (background models) in order to deal with the problems that might arise.

4.5.1 Moving Mean

A very simple improvement to the use of a static background that deals with the problem of illumination changes is to update the background after each frame through the following formula:

$$B_{t+1,s} = \alpha I_{t,s} + (1 - \alpha) B_{t,s}, \tag{25}$$

Where $\alpha \in [0, 1]$ is a constant, which determines the rate at which the background changes [15].

4.5.2 Gaussian Mixture Model

This method models each pixel individually as a mixture of Gaussian models in order to better deal with animated backgrounds. The probability of a certain pixel s, taking on a certain color at a given time t, is given by [15]:

$$P(I_{s,t}) = \sum_{i=1}^{K} w_{i,s,t} N(\mu_{i,s,t}, \Sigma_{i,s,t}) \tag{26}$$

Where $N(\mu_{i,s,t}, \Sigma_{i,s,t})$ is the i^{th} Gaussian model and $w_{i,s,t}$ its weight. The method is introduced and described in detail in [15], where depending on computational power, K is set to between 3 and 5. Each new pixel value $I_{t,s}$ is compared to the K Gaussian distributions and a match is defined as values that are within 2.5 standard deviations of the mean. The weights are then updated according to [15]:

$$w_{k,s,t} = w_{k,s,t-1} + \alpha(M_{k,s,t} - w_{k,s,t-1}) \tag{27}$$

Where α is the learning rate and $M_{k,s,t}$ takes on value 1 if pixel s at time t matches the k^{th} Gaussian and 0 otherwise. If there are no matches, the Gaussian with the lowest weight is replaced by a new Gaussian with the value at pixel s as the mean, high variance and a low prior weight. The mean and variance of the unmatched distributions are left unchanged and for the matched distributions, in accordance with [15], the parameters are updated as follows :

$$\mu_{s,t} = (1 - \rho)\mu_{s,t-1} + \rho I_{s,t} \tag{28}$$

$$\sigma^2_{s,t} = (1 - \rho)\sigma^2_{s,t-1} + \rho(I_{s,t} - \mu_{s,t})^T(I_{s,t} - \mu_{s,t}) \tag{29}$$

and ρ is given by:

$$\rho = \alpha N(I_{s,t} \mid \mu_k, \sigma_k) \tag{30}$$

In order to model the background the Gaussian distributions are sorted according to w/σ and the top B chosen according to:

$$B = argmin_b\Big(\sum_{k=1}^{b} w_k > T\Big) \tag{31}$$

Where T is a threshold chosen as the minimum portion of the data that should be accounted for by the background.

4.6 Clustering and Classification

Clustering methods are used in order to find groups in a data set, while classification methods are used to assign observations to an already known group. In other words, building a classification rule requires a data set of labeled observations while clustering can be used in order to label a set of unlabeled observations. Obviously, in clustering, the observations that are close together should be placed in the same cluster. The goal is to minimize within cluster distances and maximize between cluster distances. The following subsection will present a few different clustering and classification methods. All definitions and equations here are from [17].

4.6.1 Clustering

There are a number of different clustering methods, parametric and non-parametric.

One very popular non-parametric clustering method due to its simplicity and speed is called k-means. It is an iterative process to find k clusters. The process is initialized by choosing k observations at random as the initial guesses for the cluster centroids. Clusters are created by assigning observations to the nearest centroid according to euclidean distance. Once all observations have been assigned to a cluster, the mean of each cluster is calculated and taken as the new estimate of the cluster centroid. This process is then repeated until convergence is reached. A drawback of this method is that it can be sensitive to the initial values which means that it might have to be performed several time in order to ensure that stable clusters are found. Because the method uses euclidean distance and calculates means, it is sensitive to noise and outliers.

Partition around medoids (PAM) is another non-parametric clustering method that is quite similar to k-means, but it is more robust. Instead of recalculating a new estimate of the centroids at each iteration, the observation that has the smallest overall distance to the other observation in the group is taken as the medoid. This means that other distance measures (for example correlation) can be used instead of euclidean distance. Both PAM and k-means tend to look for clusters of spherical nature.

Parametric clustering is a way to take cluster shape into account. Most common is the multivariate normal assumption. That is, assuming that the clusters can be represented by ellipsoid blobs in space.

4.6.2 Classification

Given a set of labeled observations, new observations can be classified into the existing groups based on the known information. One simple classification method is a voting rule called k-nearest-neighbors (k-NN). It is based on a majority vote among the k (often 3 or 5) nearest neighbors. Each observation is classified as belonging to the same group as the majority of its neighbors.

Other examples of classification methods are linear and quadratic discriminant analysis, which are more similar to the parametric clustering above. Groups are assumed to be represented by specific distributions, such as for instance multivariate normal distributions. New observations are classified based on the probability of belonging to a certain group. These methods are a bit more complicated and require assumptions about the distributions of the groups.

5 Method

5.1 Posture Recognition

In order to identify the drivers postures, two different features were chosen to track. The first was based on tracking the marks placed on the side of the body. They were extracted using color thresholding and then tracked using KLT. The second feature was based on extracting the body of the driver. After extraction, an ellipse was fitted around the contour and the features of this ellipse were noted in each frame.

5.1.1 Mark Tracking with KLT

With the study data, the video was in gray-scale and hence thresholding was only possible in this color space. After thresholding the marks were extracted and in the initial frame the points for the KLT tracker was initialized. Because of noise in the data, the point eventually drifted off, away from the desired mark. This lead to adding a continuous detection of the mark using thresholding and blob analysis to continuously track the mark and add in a drift of the KLT point to move towards this found blob if it was within a given radius.

This meant that other noise in the image of appropriate size also affected the drift, but this noise was fleeting whilst the shoulder mark was detected consistently and hence the tracking worked reasonably well. Difficulties arose when the driver moved a lot and created shadows which were otherwise not there. Since binarization in gray-scale was the only option it proved difficult to find the mark during these segments, which were really the segments of utmost interest for us. Additionally, another big issue with this data was that there was only a single mark on the drivers shoulder. This made it very hard to differentiate between all the different driving postures and further analysis was deemed futile. In analyzing the study data, only the first feature mentioned above was tried before the choice was made to gather new data and focus on that instead.

In analyzing the new data, there were now three separate marks along the side of the driver. One on the shoulder as before, and two on the side of the torso. This meant that after tracking the marks, it was possible to classify what defined the different driving postures in regards to the mark positions and later test this on the test drives.

The marks on the side of the driver were black and yellow, and the yellow parts were used to locate the marks by thresholding on the intensities in each of the three color channels (red, green and blue). Since there was not much yellow in the image outside of the marks, the location of the marks could be found. After the thresholding noise was removed through different morphological operations, so that the binary image only contained blobs of the appropriate size.

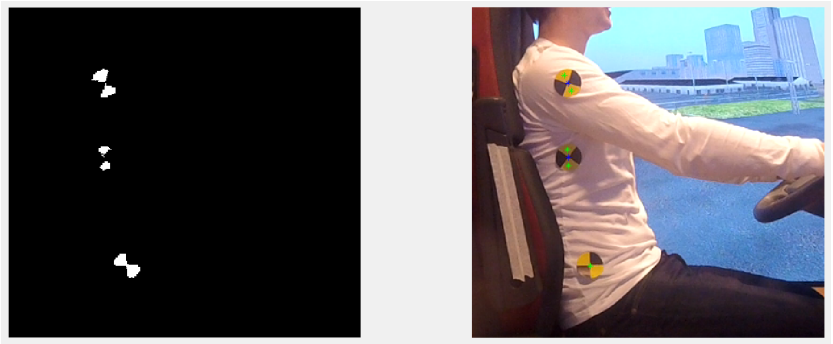

Figure 12: On the left are the marks after binarization based on thresholding in RGB space and noise removal through morphological operations. On the right are the calculated mark centers (blue stars) estimated from the centroids of the white blobs (green stars).

Again the tracking was performed through applying the KLT algorithm. The points in the image used for initialization of the algorithm were calculated through the centroids of the white blobs in the binary image. Often each mark was represented by two or more white blobs and the center of the mark was estimated to be the mean of the centroids of the white blobs. Figure 12 shows an example of the binarization and estimated mark centers for the new data. As before, noise in the image caused the points to drift away from the true position over time and made the measurements less accurate. This problem was dealt with in the same way as in the analysis of the study data, with a drift towards the detected blobs within a specified radius.

Arm movements during the test drives meant that some of the marks would occasionally become occluded. In these situations, the KLT tracker automatically continues to track the best matching point, but this proved to be a big source of error as it would for example start tracking the elbow once the middle mark became occluded. Once the mark reappeared, the previously used technique of drifting the point back towards a detected blob would sometimes help restore the tracker to the correct location. However, sometimes

the tracker had moved too close to one of the mark other marks causing the drifting technique to pull the tracker towards the wrong mark. This problem was handled by re-initializing the point tracker every time the number of detected marks changed.

Figure 13: Example of one mark being occluded by the arm of the driver. In the left image, the point tracker is allowed to continue tracking when the mark is occluded, resulting in the blue star on the elbow. In the right image the point tracker has been re-initialized to only track the two visible marks.

Figure 13 above shows an example of what happens when a mark is occluded. On the left the point tracker has kept tracking the best matching point, resulting in the blue star on the elbow. On the right the tracker has been re-initialized to track only the two visible marks. The technique of re-initializing the point tracker every time the number of visible marks changed meant that the dimensions of data could change from frame to frame and therefore the classification of postures became somewhat more complicated.

5.1.2 Ellipse Fitting

The improved quality of the video sequence meant that the body could be extracted much more accurately, which in turn allowed additional analysis. The RGB color space was used in order to find the shirt of the drivers, by looking at the proportions of each color channel and the sum of the three channels. In order to better analyze the body, the arm (which had the same color as the body) needed to be removed. This was done in the binary image using a simple column sum and setting a low threshold.

Once the body was identified and a binary image had been created, the properties of the identified blob could be used to estimate the posture of the driver. The shape of the body was determined to best be estimated by an ellipse and the postures were estimated by looking at the orientation of the ellipse, that is, the angle of the major axis relative to the x-axis.

Figure 14: Example of the white blob resulting from thresholding on the values in RGB space representing the body of the driver. After noise removal through morphological operations an ellipse is fitted to track the orientation of the body.

Figure 14 above shows an example of what the binarization and estimated ellipse of the body looks like for two different postures from one of the training videos. As the posture has changed, it can be seen that the orientation as well as the length of the major and minor axis and the centroid of the ellipse have changed. These changes could possibly be used in the future as features when determining which posture the driver is in at a given time.

5.1.3 Clustering and Classification

The data resulting from the mark tracking and ellipse fitting was recorded from the training videos where the drivers would sit and hold each posture for three seconds. The pause at each posture should mean that there is a higher density of data points recorded at each of the postures.

In order to account for the potential of altering video placement between recordings, the data needed some manipulation before meaningful clustering could take place. The data was normalized using the initial position of the marks for each recording and then adding a different constant to the three

different y-coordinates in order to separate out the three marks.

Once the data had been normalized, k-medoids clustering was applied in order to find groups corresponding to the different predefined postures. The k-medoids method was chosen because of its simplicity and the fact that it seemed to fit the data quite well. In order to avoid the possible local minima that can result from k-medoids clustering, the clustering process was repeated five times and the best results in terms of within cluster distances is chosen. The resulting cluster labels were then used to classify new data based on the k-NN method with three neighbors.

Since only data from the visible marks in each frame was recorded, the posture in each frame needed to be classified using different amounts of data, depending on if any marks were occluded. Clustering of the training data was performed separately based on all possible combinations of marks (it was assumed that at all times of classification at least one mark was visible). After the clusters were created, classification of the posture in each frame was done according to the model corresponding to the visible marks in that frame. Obviously classification based only on the bottom mark would be very difficult as movement here is minimal. However, the instances when only the bottom mark is visible are rare and in such cases, the orientation of the body is used.

5.2 Gaze Estimation

As mentioned earlier, this thesis dealt with two separate data samples: the original study data, and the data which was gathered during the course of the project. In analyzing the study data, several methods were tried to estimate the gaze direction. One idea was to track the drivers faces and in using the resulting bounding box, one could identify gaze direction similar to how [4] analyzed head postures of drivers in order to identify them as drowsy. In order to accomplish this, the original Viola-Jones algorithm was combined with its extension for profile view face detection.

When it became clear a new data gathering would occur, color-based and motion-based binarization was investigated to produce similar bounding box data, but perhaps more accurate. These methods had been unusable with the original study data seeing as there were no background images, there was frequent movement in the background by the people conducting the test, and the images were black and white.

During this time another approach was tried, consisting of detecting the drivers face with the Viola-Jones algorithm when possible, and inside the facial region extracting feature points to track using the Kanade-Lucas-Tomasi feature tracker similar to [10]. This method proved a strict improvement of the aforementioned face trackers, since the resulting data was a distribution of points over each frame, and not just a bounding box of the facial region.

This data was more accurate than that which was previously found. When processing the newly gathered data, the VJ and KLT approach was prioritized over the others for face tracking for this reason. However, in order to detect minor movements of the drivers gaze, the main method was supported by methods of eye extraction and tracking. In accordance with the analysis of eye extraction methods in [5] we chose to use the extension of the Viola-Jones algorithm for eye detection and also color-based binarization of the face and filtering out elliptical blobs as eyes. The Viola-Jones extension surpassed the binarization method and was hence used in the final results.

5.2.1 Viola-Jones Profile View Detection

When processing the original study data, the Viola-Jones algorithm for frontal view face detection was combined with its extension for profile view face detection. In order to get a high frequency of matches in profile view, the algorithm required a low threshold for acceptance. This resulted in a large number of bounding boxes being tracked in each frame as a result of plenty of false positives which needed to be filtered out.

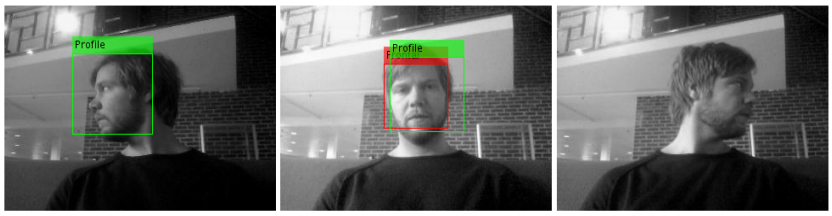

Figure 15: The extended Viola-Jones algorithm found profile views in some cases (left), but not to a degree which was required (see right). Further, in order to keep the threshold low to detect profile view faces to a reasonable extent, it often detected frontal view faces as being in profile (center) along with several false positives.

Filtration was based on using a linear filter over the past few frames to predict the faces position in the next frame and accept the closest match as the

true face. Another filtration was to calculate the overlap of area between the face detected in the prior frame with the ones detected in the next. This gave a number of tracks varying in likeliness and over a few frames tracks could be disregarded as false.

Later on, data was gathered using a webcam device and the created filters were once again applied to this new data. This approach was not applied to the data gathered later on since it produced results which were strictly worse than other methods.

5.2.2 Motion-Based Face Tracking

In the interim between acknowledging that a new data gathering would be necessary and the time of its execution, a motion-based approach to face tracking in order to determine gaze direction was made. A background image was taken from a webcam device and subtracted from each frame in a video sequence of a fake driver turning his head from side to side, emulating driving conditions as closely as possible in regards to camera distance and positioning. Seeing as lighting conditions would be constant during a test, and the background would not change, a static background image approach was chosen.

The background subtraction yielded an extraction of the driver which could later be thresholded to get a binary image of the facial region. This region was contained within a bounding box and the information extracted from the bounding box was used to classify the gaze direction.

This approach was not applied to the initial study data since there were no background frames available and plenty of movement in the background. It was also not applied to the gathered data since it was deemed to give strictly worse results than other tried methods. Furthermore, it would require certain procedures be adhered to in future studies which could prove cumbersome, such as saving a background image in a separate file under a specified naming convention or similar.

5.2.3 Color-Based Face Tracking

Similar to the motion-based face tracking a color-based approach was also applied during the aforementioned interim. This was done by detecting the face using the Viola-Jones algorithm during the first frames of each video

sequence and within this region of interest the nose was found using an extension of the same algorithm. The nose region was used to create a Camshift tracker which used the color of the nose in different color spaces to track the face pixels.

The analysis was made on data gathered from a webcam device and being surpassed by the VJ and KLT method it was not further pursued for the final results. However, similarities can be drawn to the eye extraction using color-based binarization to be discussed later in this section.

5.2.4 Viola-Jones and KLT

In each frame, faces were detected and bounding boxes around faces were created. Inside these bounding boxes, feature points [11] were detected similar to [3]. These points were tracked frame to frame using the Kanade-Lucas-Tomasi feature tracker, even when a face was not detected by the Viola-Jones algorithm.

Figure 16: Driver with bounding box around face (yellow) and KLT feature points (green marks). The points are approximately normally distributed around the center of the image when the driver looks straight ahead (left), but when the driver turns his head (right) the mean and standard deviation are greatly changed.

These points were distributed differently depending on the rotation of the drivers face and the mean and standard deviation of the distribution of points

were calculated and used to determine gaze direction.

This method was applied both to the study data and the data gathered later on, but proved much better in processing the new data with higher resolution, more data points and less data loss from conversion.

5.2.5 Eye Extraction

Further efforts of estimating gaze direction was made by extracting the eyes of the driver in order to detect more minor changes. Even though [5] claimed an intersection of gradients approach might be best, it was deemed too slow computationally and also would require too much time to implement.

The skin was initially binarized using RGB, YCbCr and HSV color spaces and various thresholds were tried for each dimension in the aforementioned color spaces. As a result of the discussion in [2] most attention was given to YCbCr and HSV, but RGB was still tried.

Another approach was to once again extract the nose region as in the color-based face tracking method. When extracted, a histogram was created in each variable of the current color space and each pixel in the entire frame was mapped to a likelihood value of the bin value which it fell into in this histogram. After this mapping was done, a threshold was set on the new image in order to create a binary representation of the skin region.

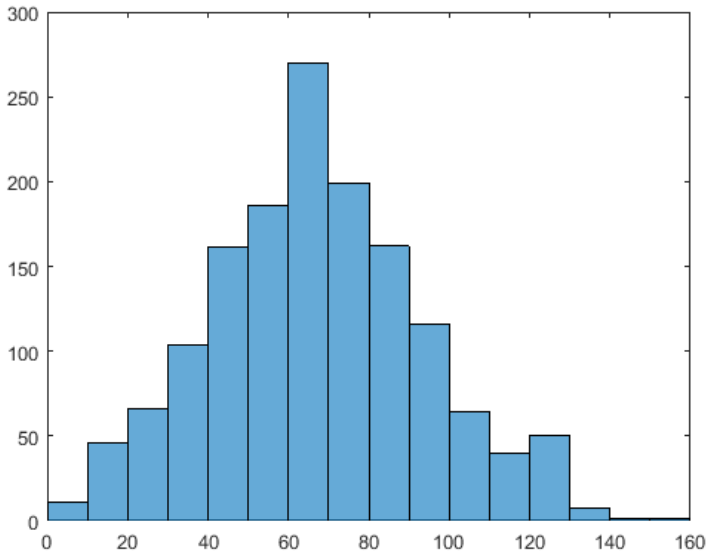

Figure 17: Example of a histogram in the blue channel of the RGB color space for a detected nose region. The elements of the blue channel of the frame would be mapped to the values of the bin they resided in and averaged over the total number of pixels. For instance, a pixel with blue channel value of 35 would be mapped to approximately 100 divided by the total number of pixels (sum over all the bin values).

In each case the resulting binary image contained blobs representing the eyes along with varying degrees of noise. The eyes were filtered out by approximating blobs as ellipses as well as filtering on area. Only frames in which a face could be detected from the frontal view camera was analyzed in order to avoid too much noise cluttering the analysis.

Additionally, the eye detector trained by Yu Shiqi [14] was used to detect eyes within the region of interest found after having first detected faces with the Viola-Jones face detector. This eye detector is an extension of the original Viola-Jones algorithm and produced results which proved more desirable than the color-based binarization approach.

6 Results

As mentioned previously; during the course of the project, different data samples have been used. Originally the data consisted of the grayscaled h.264 videos from the study done before the thesis project started. However, while working with this data, Volvo decided they would improve their data gathering hardware for the future. Therefore new data was gathered which more closely simulated the conditions of which future studies are to be conducted. Hence, the results in this section focus heavily on this new data.

6.1 Posture Recognition

Posture recognition was done mainly through the tracking of marks placed on the drivers side. The position of the marks was recorded in each frame, and k-medoids clustering along with k-NN classification was used to classify the posture at each given time. The orientation of the body was also tracked by estimating the angle relative to the x-axis of the white blob resulting from binarization of the body.

The original goal was, as described in the scope section, to detect seven postures. The difference between posture one and two is that the head leaves the backrest. This would not be detectable using the marks on the shoulder and torso, but would instead require binarization of the head. This has not yet been accomplished due to lack of time. Because of this, position one and two have been combined for a new definition where the head position was irrelevant. Furthermore, after discussions with Volvo, it is agreed upon that all postures are not equally important to be detected and hence it is of interest to investigate if the algorithms can more accurately find a subset of the positions. Therefore, analysis has been done on 6, 5 and 4 clusters respectively.

6.1.1 Mark Tracking with KLT

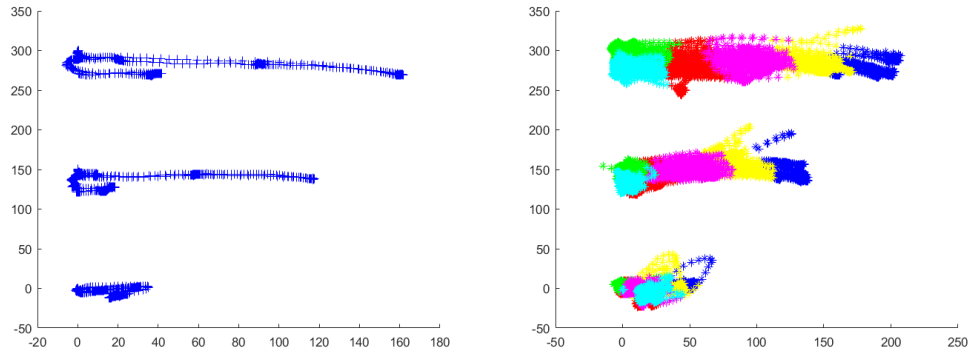

Figure 18: A plot of the recorded mark data (left) from one of the training videos, areas with a lot of observations represent the posture pauses. The right plot shows the result of combining data from all of the training videos and applying k-medoids clustering looking for 6 clusters shown with 6 colors in the right plot.

Figure 18 above shows the typical data resulting from tracking the marks in one of the training videos (left) and the clusters found when applying the k-medoids method to the combined data from all of the training videos (different colors represent different clusters), looking for 6 clusters (right). The three groups separated out in the y-direction represent the three different marks

When a few of the training videos were used as training data and the rest as test data, the k-medoids clustering and k-NN classification showed quite promising results. The videos of the actual driving present a larger challenge as they contain noise in the form of movements that are not well defined by the predefined postures. This noise was mainly a result of arms moving in front of the middle mark and shoulder displacement as a result of steering, moving the top mark and sometimes causing a false prediction.

Figure 19: An example of a situation where the prediction changes from posture 3 (left) to posture 4 (right) due to turning of the wheel. The ground truth stays constant.

Recall that when the training videos were recorded the drivers paused at six different postures. It is therefore natural to look for six different clusters in the resulting data. Figure 20 shows the within cluster distances from each point to the cluster medoid, for clustering applied to the training data. Distances are shown for different numbers of clusters, ranging from 2 to 10. Small distances are desired, but naturally searching for more clusters will result in smaller distances, therefore it is natural to look for the point where the improvement gained from adding another cluster is small.

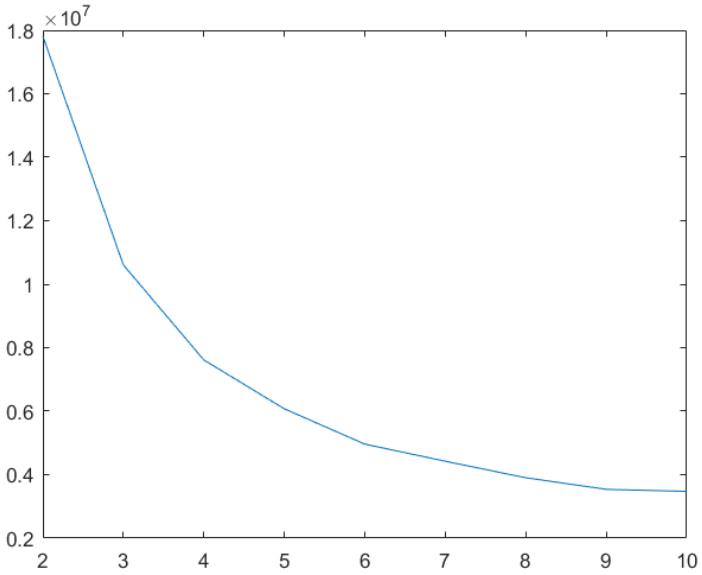

Figure 20: Within cluster distances from each point in the cluster to the cluster medoid. Small distances are desired but adding more clusters naturally decreases the distances, therefore the point where the improvement from adding another cluster is small is what is looked for.

It can be seen in Figure 20 that there is some evidence for six clusters being a good choice for the number of clusters present in the data. However it can also be seen that the improvement in within cluster distances is perhaps starting to decrease already when adding a fifth cluster. In the first attempt to classify the posture of the driver, the clusters were created using the assumption that there were 6 reasonable clusters in the data to be found. However, it seemed like this was too many, especially since the slouching position was not utilized during the short drives in the test videos. Therefore clustering using only five positions (taking the two slouching positions as one) was performed.

In order to test the results of the posture classification based on the mark data, a ground truth was created manually by carefully going through each video, pausing and noting every time the posture changed. The ground truth was then compared to the predictions to see how well they agreed. When comparing the ground truth to the predictions made through k-medoids clustering looking for 5 clusters the rate of correct predictions was found to vary a bit from driver to driver with the maximum rate being about 0.93 and the lowest about 0.63. Overall the rate of correct predictions with 5 clusters was found to be 0.76.

The confusion matrix in Table 1 below shows the relations between the ground truth and the predictions. The 5 different postures are represented in numerical order in the rows and columns. Rows represent the ground truth and columns the predictions. During the test drives the driver was in posture two for the overwhelming majority of the time, and it was also classified correctly quite often.

		Prediction				
		2	3	4	5	6
Ground Truth	2	15743	372	6	0	71
	3	2918	1458	1206	99	521
	4	8	80	584	182	48
	5	0	10	4	17	4
	6	0	0	0	0	0

Table 1: The confusion matrix showing how the predictions were spread out in relation to the ground truth. Elements on the diagonal represent correct predictions. Postures described (as related to the predefined ones in the introduction) were postures 2 through 5 with 6 and 7 grouped together.

In Table 2 below, each value from the confusion matrix in Table 1 has been divided by the row sum in order to get the percentage of correct predictions for each posture. Clearly posture two which is by far the most common posture has the highest level of prediction correctness at 97%. Posture 4 and 5 also have a majority or near majority of correct predictions, however posture number 3 is quite often taken as being posture number two.

		Prediction				
		2	3	4	5	6
Ground Truth	2	0.972	0.023	0.000	0.000	0.004
	3	0.471	0.235	0.195	0.016	0.084
	4	0.009	0.089	0.648	0.202	0.053
	5	0.000	0.286	0.114	0.486	0.114
	6	0.000	0.000	0.000	0.000	0.000

Table 2: Confusion matrix over the prediction against the ground truth, written as a fraction of each row sum. Each element represents the fraction of times that a row value (ground truth) was interpreted to be the corresponding column value (prediction).

A lot of the incorrect predictions are classifying the posture to be one of the closest ones on either side. For example, posture 3 is often taken as being either posture 2 or posture 4. This could be because the difference between postures is small and sometimes changes are gradual, making it difficult to manually create an objective ground truth. The fact that posture 5 is classified as posture 3 so often (29% of the times) is more difficult to explain. Also posture 6, the slouching, never occurs during the test drives, but we can see in Table 1 that posture 3 is quite often classified as posture 6. This makes sense because one of the slouched postures was quite similar to posture 3 with the shoulders away from the backrest.

Because of these issues, the clustering and classification was repeated using only four clusters in an attempt to better capture the postures that actually occurred during the test drives. There is also some support for using 4 clusters in Figure 20 where a slight decrease in the rate of improvement of the within cluster distances can be seen.

Figure 21 shows the clusters that are found when applying k-medoids clustering to the training data for all three marks. It seems reasonable that the clusters represent the predefined postures 2 through 5, it can also be seen that the clusters representing posture 2 and 3 (red and blue in the figure) seem to be overlapping somewhat, illustrating that there is only a small difference between the two.

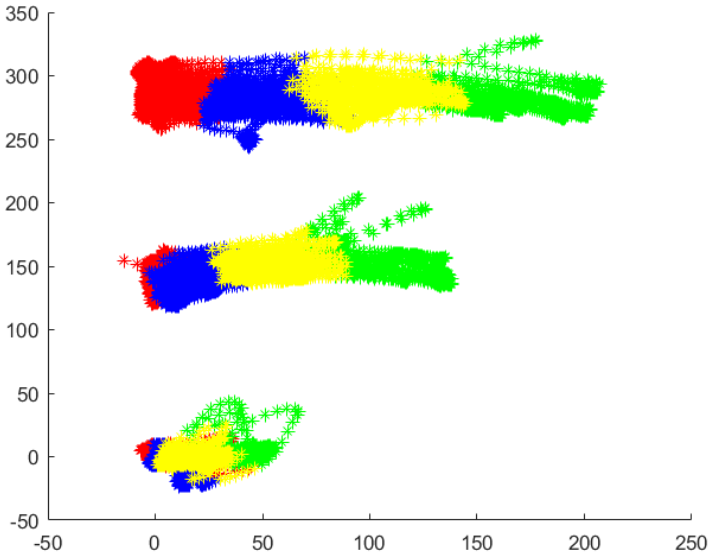

Figure 21: Clusters found when using the training data from all three marks and looking for four different clusters. Knowing the predefined postures it seems reasonable that the clusters represent postures 2 trough 5.

The rate of correct predictions when attempting to classify 4 postures instead of 5 was found to be 0.78 (compared to 0.76 previously). This improvement in prediction rate is quite small and it comes at a cost of losing the possibility to classify one of the postures, so perhaps it is not a desired change in the application of the method.

Below, in Table 3 the confusion matrix from the classification using four postures is presented. It can be noted once again that posture number 3 is the one that is most difficult to classify. As mentioned before, there is very little difference in the definitions between postures 2 and 3 which can also be seen in the confusion matrix as a lot of observations defined as posture 3 in the ground truth are classified as posture 2.

		Prediction			
		2	3	4	5
Ground Truth	2	15798	378	4	12
	3	3345	1638	1179	40
	4	33	83	728	58
	5	0	5	18	12

Table 3: Confusion matrix for the ground truth and the predictions when only classifying 4 postures.

Table 4 shows the proportions of correct classifications written on the diagonal, again posture 2 is often classified correctly. The proportion of correct predictions of posture 4 has increased a bit compared to the previous classification while the proportion of correct predictions of posture 5 has decreased. The proportion of correct predictions for posture 3 has increased slightly but is still very low.

		Prediction			
		2	3	4	5
Ground Truth	2	0.976	0.023	0.000	0.001
	3	0.539	0.264	0.190	0.006
	4	0.037	0.092	0.807	0.064
	5	0.000	0.143	0.514	0.343

Table 4: Classifications written as fractions of the row sum.

6.1.2 Ellipse

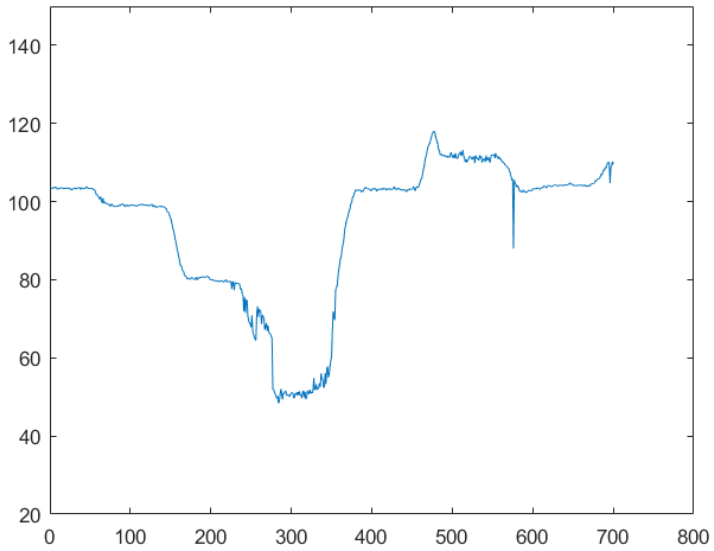

Figure 22: An example of the how the orientation of the body changes during the course of one on the training videos. The angle is estimated through the ellipse that fits the binarized image of the body. The angle seems to change along with the posture changes. The angle in degrees is represented on the y-axis and the frame number is represented on the x-axis.

Figure 22 above shows the development of the angle of the body over time for one of the training videos. We can see that there are quite large changes when the posture changes (with some noise). This method required the binarization to work well, as the orientation of the ellipse was sensitive to objects falsely detected as parts of the body. One large source of such errors were the arms of the drivers, which is natural since the arms have the same color as the body. In the training videos it was possible to cut the arms out of the image, however there was no steering in the training videos so this was more difficult to do with accuracy in the videos from the test drives.

Even though the data from the test drives contained much more noise than the data from the training videos, as can be seen in Figure 23, it is possible to see trends in the data relating to the posture. For instance, between frames 800 and 1300 we can see that the plot of the angle stays fairly constant and this corresponds to a portion of the data where both the ground truth and

the prediction achieved from the mark tracking is constant (at posture 2).

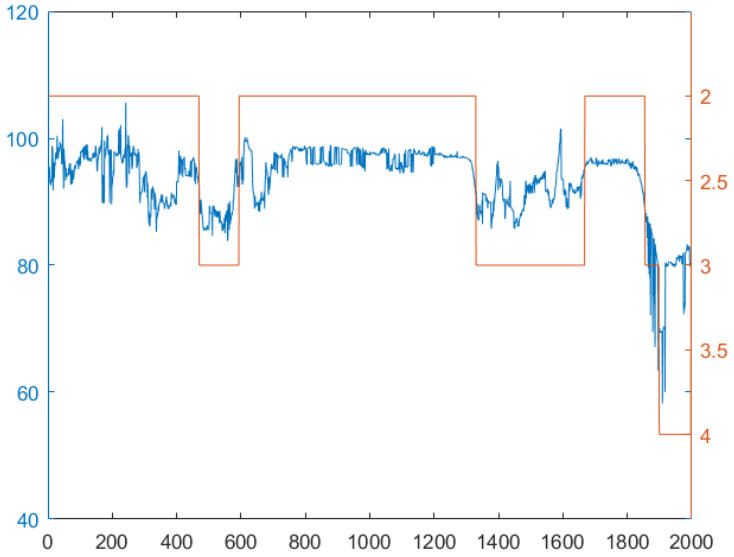

Figure 23: Example of the recorded orientation of the body from the first 2000 frames for one of the test drives plotted together with the ground truth for those frames. It can be seen that the angle and the ground truth are correlated. The value angle is represented by the blue y-axis on the left and the value of the ground truth by the red y-axis on the right. Frame numbers are represented on the x-axis

In Figure 23 above, the development of the body orientation is plotted along with the ground truth for that section of the data. Even though there is a lot of noise in the data it seems like there is some correlation between the ground truth and the angle.

The training data was used to build a model and classify the data from one of the test drives. When looking at the within cluster distances it seemed like there were four or five clusters in the data. This can be seen in Figure 24 below, the improvement in within cluster distances becomes very small when adding a 6th cluster.

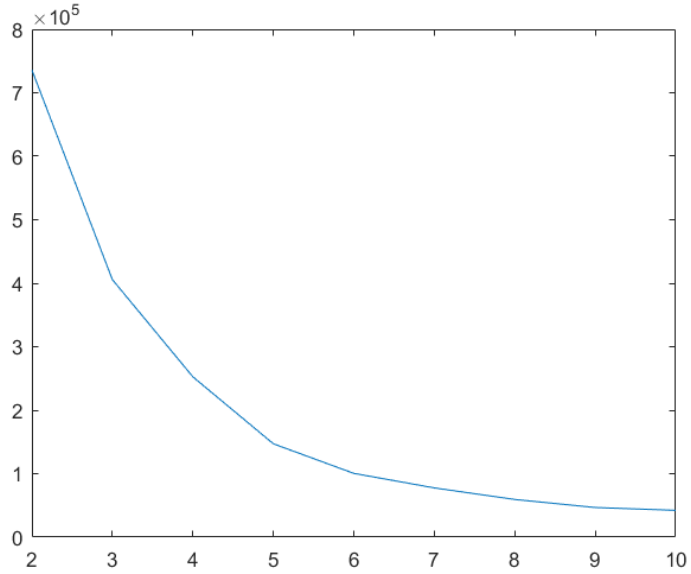

Figure 24: Within cluster distances plotted as a function of the number of clusters. The distances are represented on the y-axis and number of clusters on the x-axis. It can be seen that the curve levels off when a 6th cluster is added.

When using four clusters to train the model (representing postures 2 through 5) a correct prediction rate of 0.6147 was achieved. Again, this data contains a lot of noise due to false detection which will affect the accuracy of the prediction.

		Prediction			
		2	3	4	5
Ground Truth	2	8881	7311	0	0
	3	811	4708	669	14
	4	1	105	753	43
	5	0	4	31	0

Table 5: Confusion matrix for the predictions made using the angle of the body blob. Predictions seem to be biased toward posture 3. Classifying for 4 different postures

Table 5 above presents the confusion matrix for the predictions made using the orientation of the body blob. A lot of observations are classified as

posture three. This is interesting compared to the previous method where more observations were classified as posture 2. The fact that the two methods seemed to be biased towards different postures is something that can be investigated further in the future.

Combining the features by using data from both methods and using clustering and classification on four clusters yields the following confusion matrix:

		Prediction			
		2	3	4	5
Ground Truth	2	15787	230	0	175
	3	3647	1527	1000	28
	4	25	136	692	49
	5	0	14	9	12

Table 6: Confusion matrix for the predictions made using data from both the mark tracking and the orientation of the body. Classifying for 4 different postures.

In Table 6 it can be seen that it is still posture 3 that is most difficult to classify correctly. The rate of correct predictions was again around 0.77. Table 1 below shows the results from adding classification for a fifth posture using the combined data. This resulted in a decrease in the rate of correct predictions to about 0.67.

		Prediction				
		2	3	4	5	6
Ground Truth	2	14891	28	22	175	1076
	3	4205	167	1684	27	119
	4	98	0	744	60	0
	5	13	1	21	0	0
	6	0	0	0	0	0

Table 7: Confusion matrix for the predictions made using data from both the mark tracking and the orientation of the body. Classifying for 4 different postures.

6.2 Gaze Estimation

As described in the method section; originally, gaze direction was estimated by tracking bounding boxes of detected faces. These bounding boxes were found using various methods: the Viola-Jones algorithm combined with its extension for profile view face detection, color-based and motion-based binarization. Finally, these methods were improved upon using the Viola-Jones algorithm combined with tracking feature points inside the detected face region with the Kanade-Lucas-Tomasi algorithm gave more accurate data in the form of distributions of points.

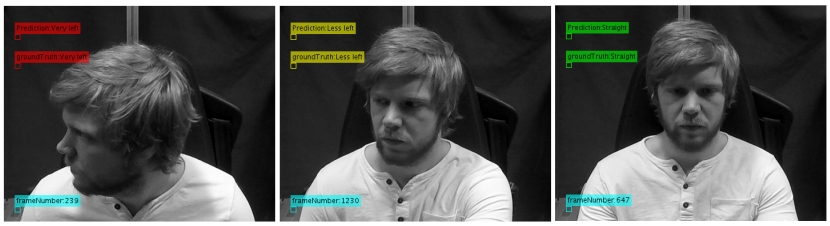

Figure 25: Driver gazing in different directions and annotations inserted as to the noted ground truth together with the resulting prediction.

In order to compare results to the videos from the gathered data, a ground truth was created. The video data gathered was carefully viewed frame to frame and paused so to note changes in driver gaze direction. This data was saved and used for later comparison with the prediction. It is worth noting that even though every effort was made to keep the ground truth verdict cohesive, certain intervals made for difficult choices and ambiguity in the ground truth can not be discounted.

Only the VJ and KLT combination was implemented for the final product since it provided the best preliminary results on the study data as well as during various tests during the period before the new data was gathered. It also provided sufficiently satisfactory results on the main data sample. However, results from prior tests of other methods is still provided in this section.

There were five different states for the drivers gaze direction: pronounced right, slight right, straight, slight left, pronounced left. These five states are written in the ground truth as 2, 1, 0, -1 and -2 respectively.

6.2.1 Viola-Jones Profile View Face Detection

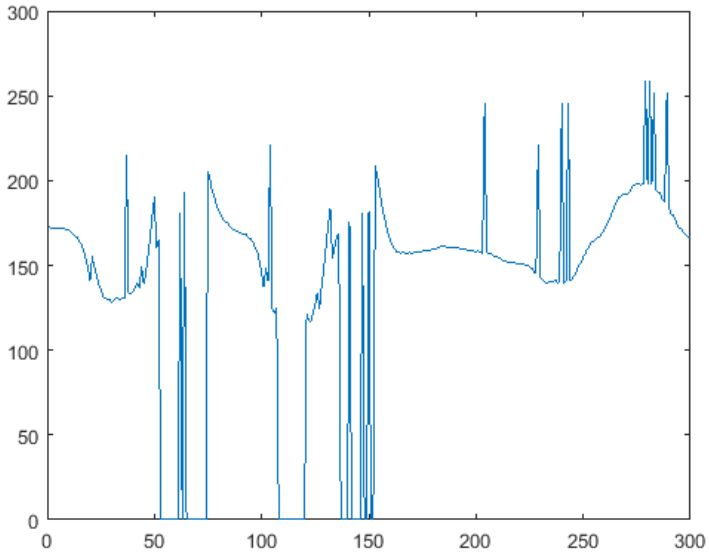

Figure 26: Center x-coordinate for bounding box data over 300 frames calculated with the frontal view Viola-Jones algorithm combined with the extended profile view algorithm. Video data collected using webcam device and the frames were cropped to a size of 300x300 pixels.

The extended Viola-Jones algorithm for profile view face detection gave varied results. It was based on the same principles as the original algorithm, but naturally profile view faces do not provide the same features as frontal view and the implementation showed the algorithm was not as stable as its predecessor.

It did find profile view faces to a reasonable extent, but in order to do so the threshold had to be kept so low it oftentimes detected frontal view faces as being in profile. This could yield false results since it would mean drivers looking straight ahead could be classified as turning, which was the main case to avoid.

However, with the new data gathering which occurred during the course of the thesis, doors opened up to use color-based and motion-based binarization. This could potentially solve the issues that the extended Viola-Jones algorithm had regarding detecting profile view faces, and give consistent bounding box data.

6.2.2 Motion-Based Face Tracking

Figure 27: An example of motion-based binarization. Background image (top left), current frame (top right), difference of current frame and background frame (bottom left) and binary image of thresholded difference (bottom right).

The motion-based approach to face tracking worked reasonably well as seen in Figure 27. The idea was to get a binary image of the face and track the bounding box around it. This was tested in the interlude of noting that the first study data was of too awkward quality to work with, and gathering the next set of data. Hence, this was recorded and tested using a webcam device.

In the figure, no morphological operations have yet been performed on the binary image and one can still clearly see the outline of the face. However, this approach would require every future study to capture background images, saving them separately and possibly using some form of naming convention which the program could later find. This could prove cumbersome and this approach was therefore not pursued further, especially since other methods proved to yield strictly better results.

6.2.3 Color-Based Face Tracking

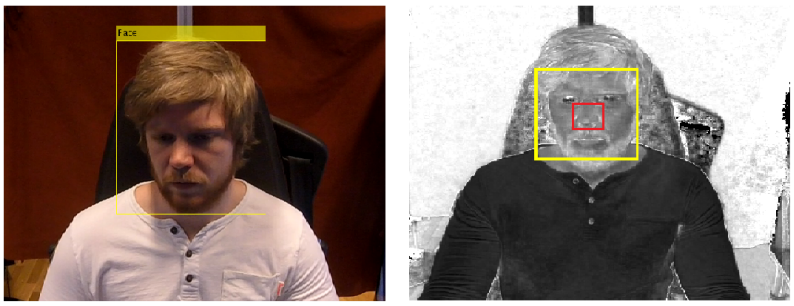

Figure 28: In the first frames of each video, the drivers face was detected and within this region the nose was detected (right). A Camshift tracker applied to this region in the saturation channel of the HSV color space was used to track a bounding box surrounding the drivers face.

Also a color-based approach to face tracking was tried, with a similar goal as to the motion-based approach, of tracking a bounding box around the detected face. The nose region was detected using the extended Viola-Jones nose detection algorithm and the saturation channel in the HSV color space of this region was used for Camshift tracking. As seen in Figure 28, a fairly well-fitting bounding box could be detected.

This approach yielded reasonable results, but the combined VJ and KLT method (as described in the method section) yielded more accurate data when it came to tracking of the face and was hence used instead of this method.

6.2.4 Viola-Jones and KLT

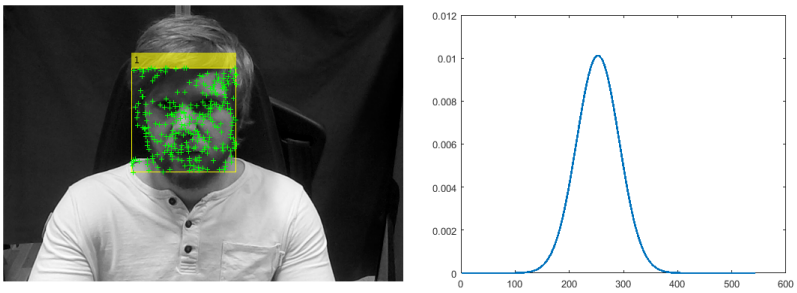

Figure 29: Feature points inside detected face regions were fitted to normal distributions and the mean was calculated and used to estimate gaze direction. The x-axis for the distribution (right) ranges from 0 to 600 (the frame width of the image) and y-axis represents the number of pixels at that x-coordinate as a percentage.

Using the Viola-Jones algorithm to detect faces, extracting feature points and tracking them using the Kanade-Lucas-Tomasi feature tracker proved to be simply a better alternative to tracking the drivers face with the extended Viola-Jones algorithm, color-based or motion-based methods described previously. The data in form of distributions of points compared to a lone bounding box around the face, was more accurate an indicator of the drivers gaze direction. The mean at each frame was compared to that of an estimated center value and deviations were deemed to either indicate slight or pronounced turn depending on how big a deviation occurred.

The mean of the tracked feature points proved to be a very good estimate of the ground truth, with a correlation of approximately 80% without any modification. As seen in Figure 30 it detects the large movements extremely well with obvious peaks and valleys. However, it proved less satisfactory in detecting slight movements of the gaze direction even though this was expected seeing as this is obviously a more difficult task and the mean of the feature points does not change to any great extent when considering small deviations in gaze direction.

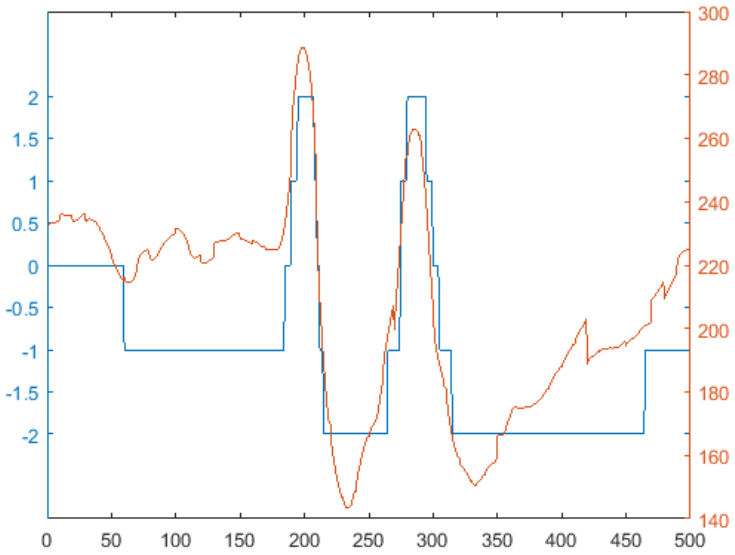

Figure 30: Mean of the feature points (red) plotted together with the ground truth (blue) for 500 frames of the data. The five states in the ground truth pronounced right, slight right, straight, slight left and pronounced left are written as 2, 1, 0, -1 and -2 respectively.

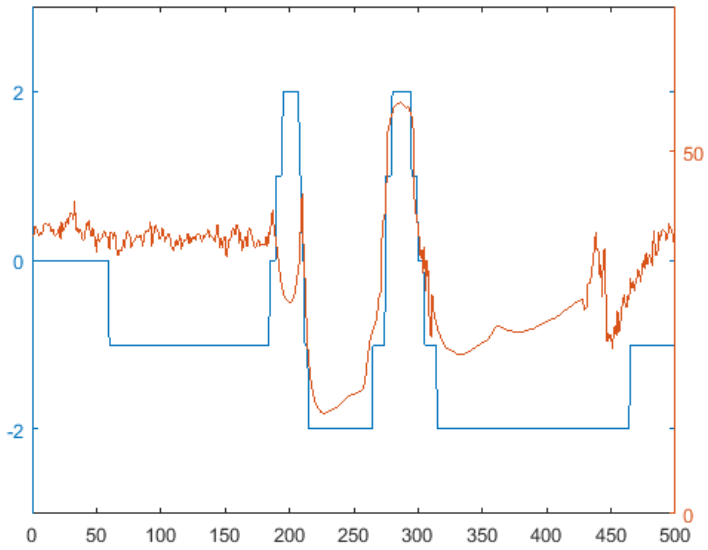

Figure 31: The standard deviation of the feature points (red) plotted together the ground truth (blue) for 500 frames of the data. The five states in the ground truth pronounced right, slight right, straight, slight left and pronounced left are written as 2, 1, 0, -1 and -2 respectively.

Efforts were made to process the standard deviation data and use this for filtration. However it proved unreliable and perhaps superfluous and was hence not used after initial testing.

Hopes of combining the data of the mean with the standard deviation proved less fruitful than anticipated. One issue was that occasionally when the drivers turned their heads, some feature points would remain around the neck area opposite the turning direction. This would create a large peak in the standard deviation data on that turn (Figure 31 at approximately frames 270 to 300). Other times these sort of points were totally lost and a very tight bounding box around the face was instead found (Figure 31 at approximately frame 210 to 265).

This meant that both very large and small values indicated the same thing, and there could be huge jumps in the sigma data because of this, even when the driver was pausing while turned in a single direction. With further processing and logic this issue could probably be solved, however the mean data itself proved so useful it was deemed unnecessary and instead the eye data was used to detect minor head movement to further improve the process.

6.2.5 Eye Extraction

Figure 32: Several methods were tried for color-based binarization. Thresholding in RGB color space (top left), HSV (top right), YCbCr (bottom left) and histogram based likelihood thresholding in RGB (bottom right).

As seen in Figure 32, different color spaces yielded varying results, with the histogram thresholding providing the best binary image, even though thresholding in the YCbCr color space worked well as well. The histogram approach was chosen as the best fit and eyes were detected by morphological operations modifying the image to contain mainly the face and thereafter searching for ellipse shaped blobs within this region.

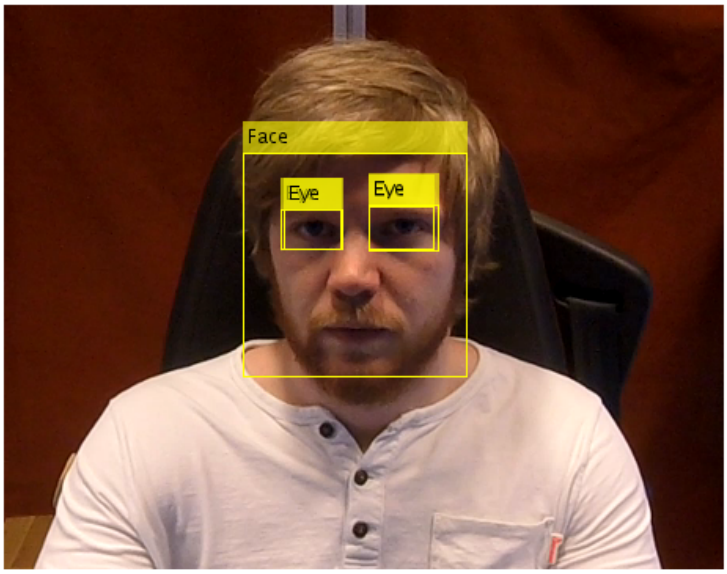

Figure 33: The drivers eyes detected using the extension of the Viola-Jones algorithm by Yu Shiqi.

However, in the end the Viola-Jones extended algorithm for eye detection proved satisfactory and it circumvented several issues which existed with the color-based approach. It was hoped that eye extraction using binarization would produce better results than the Viola-Jones algorithm, but the testing that was done proved this not to be reasonable. It also proved unnecessary since the major errors with the Viola-Jones algorithm came as a result of the driver turning more than approximately 15 degrees to either side. However, in this range the mean of the feature point distribution proved more than satisfactory and hence this was a suitable range to restrict ourselves to when tracking the eyes.

Further, color-based binarization was susceptible to errors if the nose region didn't provide a good enough fit to the rest of the face. Lighting changes for instance could develop into a future problem. This could be solved by manually setting thresholds, but seeing as the Viola-Jones extension worked well and did not require this user interaction, it was deemed a better choice.

This realization lead to prioritizing the Viola-Jones extended algorithm approach for eye tracking rather than using blob analysis on the binary image after color thresholding which was the original plan.

6.2.6 Eye Data

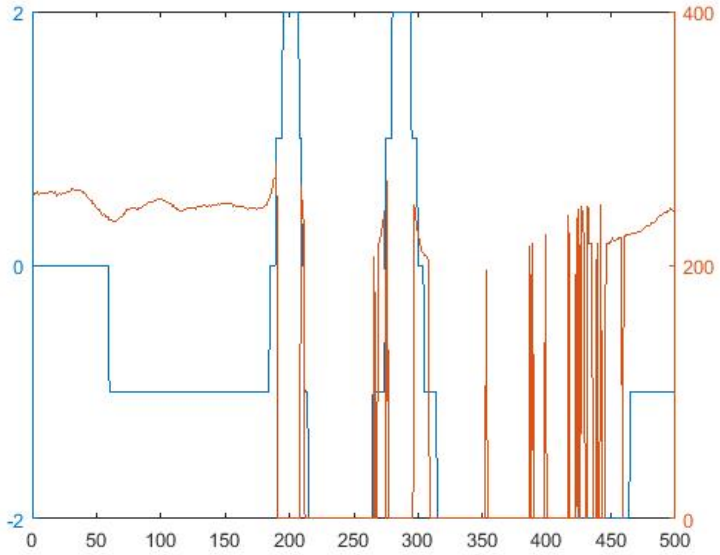

Figure 34: The average of the eyes x-coordinates (red curve) plotted together with the ground truth (blue curve) for 500 frames of the data. The five states in the ground truth pronounced right, slight right, straight, slight left and pronounced left are written as 2, 1, 0, -1 and -2 respectively.

As seen in Figure 34 the eyes are lost during extreme head turns (evident by the plots dipping down to zero in the figure) whilst being good indicators of the ground truth when they are found. In order to more simply find a correlation between the eye data and the ground truth, a single feature in the form of the average of the eyes x-coordinates was chosen, instead of using both eyes values separately.

In order to not let horizontal head movement affect the results, the values were taken in relationship to as estimated average of the eye data which corresponded to the driver looking straight ahead.

6.2.7 Combining Results

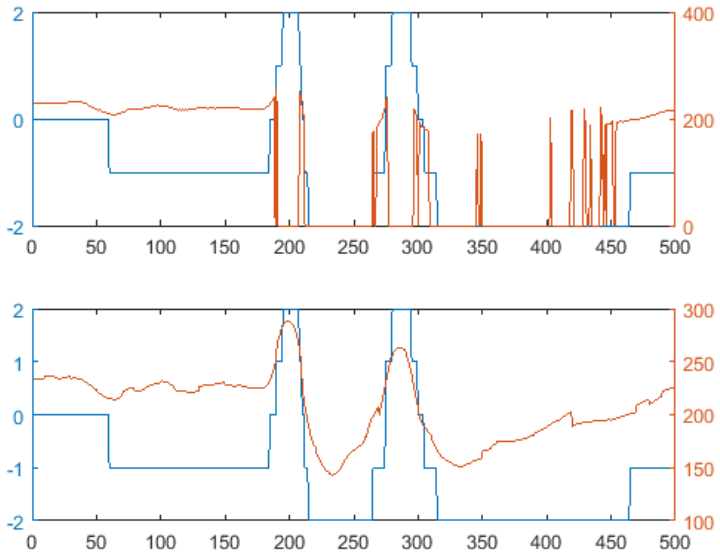

Figure 35: The mean of the feature points (red curve, bottom) and average of the eyes x-coordinates (red curve, top) plotted together with the ground truth (blue curve, top and bottom) for 500 frames of data. The five states in the ground truth pronounced right, slight right, straight, slight left and pronounced left are written as 2, 1, 0, -1 and -2 respectively.

In order to get a good prediction of the ground truth, the eye data had to be combined with the mean data to create a complete picture over the interval. Frames where eyes were detected were predicted using the eye data whilst frames without used the mean of the feature points distribution.

The sections without eye data had the values of the mean of the feature points compared to an averaged value taken as an estimated center point while the sections without had the averaged eye data compared to an estimated center point for the eye data.

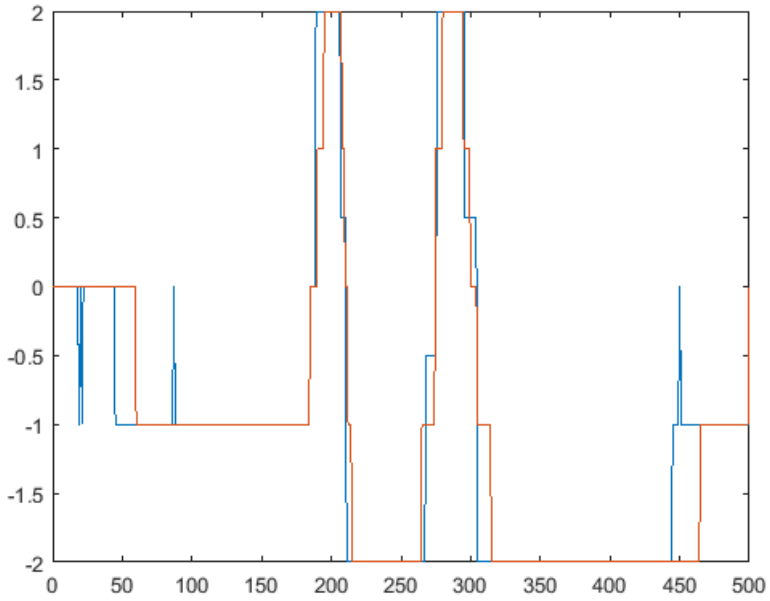

Figure 36: The algorithms estimation of the driver gaze direction (blue) plotted together with the ground truth (red) over 500 frames of data. The five states in the ground truth pronounced right, slight right, straight, slight left and pronounced left are written as 2, 1, 0, -1 and -2 respectively.

Figure 36 shows an example of the end result over 500 frames with a correlation to the ground truth of 93.7% for this particular section. Hence the methods described proved very good at predicting the ground truth and even the errors that do occur can at least partially be contributed to human error. In creating the ground truth, the difference between slight turn and extreme turn becomes ambiguous in certain intervals and one could have easily chosen differently when creating the ground truth. Similarly in the change between straight and slight turn.

As seen in Figure 36, some errors occur when the prediction deems it to be an extreme turn a few frames before it was considered an extreme turn in the ground truth. Having recorded in 30 frames per second, a few frames difference is negligible and it would be almost impossible to get all of these intervals correct. The spike seen at approximately frame 450 is the main error in the data above together with the spikes before frame 50. However, such small amounts of errors is to be expected and is certainly within acceptable margins of error.

		Prediction				
		pronounced left	slight left	straight	slight right	pronounced right
Ground Truth	pronounced left	2598	1109	196	72	101
	slight left	500	1473	272	8	0
	straight	93	1943	7788	2728	92
	slight right	26	279	1200	2497	912
	pronounced right	5	49	230	740	3066

Table 8: Confusion matrix over the prediction against the ground truth.

		Prediction				
		pronounced left	slight left	straight	slight right	pronounced right
Ground Truth	pronounced left	0.637	0.272	0.048	0.018	0.025
	slight left	0.222	0.654	0.121	0.004	0
	straight	0.007	0.154	0.616	0.216	0.007
	slight right	0.005	0.057	0.244	0.508	0.186
	pronounced right	0.001	0.012	0.056	0.181	0.750

Table 9: Confusion matrix over the prediction against the ground truth, written as percentage of each row sum. Each element represents the fraction of times that row value (ground truth) was interpreted to be the corresponding column value (prediction).

In Table 8 and Table 9, one sees that the overall confusion matrix is quite accurate. The diagonal is generally weightier than other elements, even though there are quite a few misunderstandings between straight and slight turn as well as between slight turn and pronounced turn.

Important to note is the type of errors that occur. When one looks through the data and compares prediction to ground truth in cases where they differ, one notices that the prediction is not always bad.

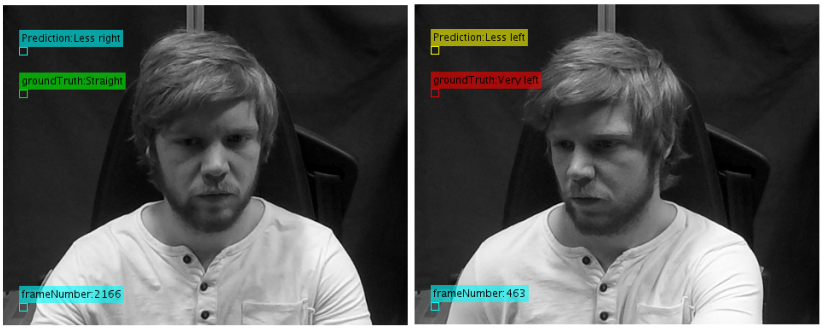

Figure 37: Sometimes the prediction is not necessarily incorrect, even if it differs from the ground truth. When creating the ground truth it was subjective what to call straight, slight turn or pronounced turn. To the left one sees an example where the ground truth is set as looking straight ahead, but the prediction saying slightly to the right is very understandable since the algorithm doesn't look at the pupils of the driver. Also sometimes the line between pronounced turn and slight turn is hard to find (right) and something which could easily have been taken as a slight turn was here written in the ground truth as a pronounced one.

Gaze Direction	Modified Value
pronounced left	0.937
slight left	0.988
straight	0.985
slight right	0.884
pronounced right	0.844

Table 10: The diagonal elements of the confusion matrix, modified to include values from adjacent columns, accepting e.g. that pronounced turns might be slight ones.

Now, if one would allow for the fact that in many cases pronounced turns can be interchanged with slight ones and slight ones can be interpreted as straight, one gets extremely well fitting results as displayed in Table 10.

The true correctness of the prediction lies somewhere between these extreme values (67% being completely correct and 94% if allowing for adjacent matches) and worth noting is that correlation coefficients between predictions and ground truths for the three drivers tested were 0.81, 0.84, 0.76).

However, one also finds a few instances where the algorithm has failed. This is often due to the driver turning his head unusually quick resulting in the loss of many KLT feature points and if he then avoids looking straight ahead for a longer time there is very little data available to make the prediction.

7 Discussion

7.1 Posture Recognition

Two methods have been looked at with the goal of automatically detecting the posture of the driver. The first, and main, method consisted of tracking the three marks that were placed on the drivers side and the posture in each frame was classified based on the position of the marks. The results for this method alone showed a correct prediction rate of 0.76 when attempting to find 4 postures. The other method looked at was to apply thresholding in order to find the body of the driver and use the properties of the resulting white blob to estimate the posture. Looking at only the orientation of the blob in relation to the x-axis, testing showed a correct prediction rate of 0.61.

It is interesting to note that the two features were biased in different directions when classifying the observations. The method of classifying based on the marks falsely predicted a lot of observations defined as posture 3 to be posture 2 while looking at the orientation, it was more likely to classify observations defined as posture 2 to be posture 3. When the two features were combined the correct prediction rate was found to be about 0.77. In total there were 23331 frames of testing data and 12508 frames of training data.

The correct prediction rates are not perfect. However, it is important to remember that some of the posture changes are very gradual and it is sometimes difficult to say with an objective certainty what the posture is in a given frame even when the video is paused. The continuous movements of the driver are being modeled by a discontinuous set of postures and it is difficult to say exactly when the transition from one state to another occurs.

When the video is played along with a printed prediction for each frame, it looks like the prediction is correct most of the time. With this in mind the results seem quite promising. Most of the errors are from predicting posture 2 when the ground truth is 3, here subtle differences and difficulties in defining the ground truth might play a role.

For future improvements, adding binarization of the head from the side view angle would give additional data. This data could prove useful not only in detecting the first posture position, which was disregarded during this thesis, but also provide useful data which could potentially improve the accuracy with which the other positions are detected. Another future improvement would be to fine tune the binarization of the body blob in order to find an

efficient way of excluding the arms. This would help reduce the noise, which in turn would make classification more accurate.

7.2 Gaze Estimation

The main weight of the gaze estimation section of this thesis has been supported by the Viola-Jones algorithm. It is an established algorithm with plenty of support and easy implementation in various frameworks which made it a natural choice. It has proved powerful throughout the project with the main issues being in regards to its known limitations of having less success in detecting faces in non-frontal images.

To fill this void, different methods were tried including color segmentation, motion-based background subtraction and feature tracking using the Kanade-Lucas-Tomasi feature tracker. All of these methods worked to the extent one could hope, but initial data showed the KLT approach would provide more consistent results with the mean of the feature points proving to be a strong indicator of the drivers gaze direction, as evident by the correlation discussed in the results section. Hence it was prioritized for the end product over the other methods.

However, even though the mean of the feature points excelled at predicting extreme head turns, it needed assistance when it came to lesser turns. This assistance was given by the extension of the Viola-Jones algorithm for eye detection. The eyes position in relation to the head together with the distance between them, is a natural feature to track in regards to head turning and was used to great effect to recognize smaller turns.

All in all the features found have given good results, but further testing is required in order to know how good they work in a more general setting. The main data processed for the final results in this report was from a single session, and if severe lighting or setup changes (for instance camera angle, camera distance or background movement) might change the results is hard to tell.

Every effort was made to gather the data in a way which corresponded to how Volvo would do it in the future, but seeing as this is a new field for them and they themselves are still thinking through how they want this to work in the future, modifications might have to be made when an approach for data gathering is fully established.

For future improvements, it would be beneficial to add functionality for the case when too little data is available. For instance, an unsure state could be added to the aforementioned five gaze directions. A user who has processed the data would be informed of the frames this occurs in and can create a manual verdict for these frames. This approach would ensure a high amount of accuracy and still involve a great deal of simplification for the user since for the most part the gaze direction can be correctly estimated.

References

[1] A Robust Face Detection Method Based on Skin Color and Edges,
Deepak Ghimire and Joonwhoan Lee,
Journal of Information Processing Systems, vol. 9, no. 1, pp. 141-156

[2] Comparative Study of Skin Color Detection and Segmentation in HSV and YCbCr Color Space,
Khamar Basha Shaika, Ganesan P, V.Kalist , B.S.Sathish, J.Merlin Mary Jenithab,
Procedia Computer Science Volume 57, 2015, Pages 41–48

[3] Face Tracking: An implementation of the Kanade-Lucas-Tomasi tracking algorithm,
D. W. Wagener and B. Herbst,
Research Gate

[4] A Drowsy Driver Detection System Based on a New Method of Head Posture Estimation,
Ines Teyeb, Olfa Jemai, Mourad Zaied, Chokri Ben Amar,
Intelligent Data Engineering and Automated Learning – IDEAL 2014: 15th International Conference, Salamanca, Spain, September 10-12, 2014. Proceedings. Pages 362-369.

[5] An Overview and Evaluation of Various Face and Eyes Detection Algorithms for Driver Fatigue Monitoring Systems,
Markan Lopar and Slobodan Ribarić,
Proceedings of the Croatian Computer Vision Workshop, Year 1. September 19, 2013, Zagreb, Croatia.

[6] Drowsy Driver Warning System Using Image Processing,
Singh Himani Parmar, Mehul Jajal, Yadav Priyanka Brijbhan,
International Journal of Engineering Development and Research, ISSN:2321-9939, Vol.1, Issue 3, pp.78 - 83, Dec 2014.

[7] Object Tracking: A Survey,
Alper Yilmaz, Omar Javed and Mubarak Shah,
ACM Computing Surveys, Vol. 38, No. 4, Article 13, Publication date: December 2006.

[8] Study of Real Time Moving Object Tracking,
Shalu, Rajiv Dahiya,
International Journal For Technological Research In Engineering Volume 2, Issue 9, May-2015

[9] Bruce D. Lucas and Takeo Kanade,
An Iterative Image Registration Technique with an Application to Stereo Vision,
International Joint Conference on Artificial Intelligence, pages 674–679, 1981.

[10] Upper Body Tracking Using KLT and Kalman Filter,
Pouya Bagherpour, Seyed Ali Cheraghi , Musa bin Mohd Mokji,
Procedia Computer Science, Volume 13, 2012, Pages 185–191

[11] Good Features to Track,
Jianbo Shi and Carlo Tomasi,
IEEE Conference on Computer Vision and Pattern Recognition (CVPR94) Seattle, June 1994

[12] Robust Real-Time Object Detection,
Paul Viola Michael J. Jones,
Cambridge Research Laboratory, Technical Report Series, CRL 2001/01, February 2001

[13] Fast Multi-view Face Detection,
Jones, M. and Viola, P.,
Mitsubishi Electric Research Laboratories, Technical Report, TR2003-96, Cambridge, MA, August 2003

[14] Eye Detection,
Shiqi Yu,
Shiqi Yu's homepage: http://yushiqi.cn/research/eyedetection

[15] Comparative study of background subtraction algorithms,
Yannick Benezeth, Pierre-Marc Jodoin, Bruno Emile, Helene Laurent, Christophe Rosenberger,
Journal of Electronic Imaging, Society of Photo-optical Instrumentation Engineers, 2010, 19.

[16] Image Analysis and Spatial Statistics,
Mats Rudemo,
Course homepage: http://www.math.chalmers.se/ rudemo/Plans/plan2006.html

[17] Multivariate statistical analysis,
Rebecka Jörnsten,
Course homepage: http://www.math.chalmers.se/Stat/Grundutb/CTH/tms041/1415/

[18] Counting moving people in 2D video sequences,
Johan Havås,
Master Thesis, Department of Mathematical Sciences, Chalmers University of Technology, Gothenburg 2015.

[19] Viola–Jones object detection framework,
https://en.wikipedia.org/wiki/Viola%E2%80%93Jones_object_detection_framework

[20] Meanshift and Camshift,
http://docs.opencv.org/3.1.0/db/df8/tutorial_py_meanshift.html#gsc.tab=0

[21] Summed area table,
https://en.wikipedia.org/wiki/Summed_area_table

[22] Evaluating comfortable driving - Three case studies on driver's behaviour in different driving situations,
Chatzopulou E. and Vlassopoulou A.,
Master Thesis, Department of Product and Production Development, Division of Design & Human Factors, Chalmers University of Technology, Gothenburg 2015.

[23] Lecture 11: Mean Shift,
Dr. Mubarak Shah,
University of Central Florida CAP 5415F Computer Vision Video Lectures 2012