Rijks*universiteit* Groningen

Faculteit der Wiskunde en Natuurwetenschappen

Instituut voor Wiskunde en Informatica

ROBUST VEHICLE DETECTION IN OUTDOOR IMAGES

An assessment of methods

Door:
Heiko Heijenga

April 23, 2003

Begeleiders:
Dr. ir. J.A.G. Nijhuis
Drs. J. Spiekstra

-"I think we have got enough information now, don't you?"
-"All we have is one "fact" that you made up."
-"That's plenty. By the time we add an introduction, a few illustrations and a conclusion, it'll look like a graduate thesis."

Calvin and Hobbes

Abstract

ROBUST VEHICLE DETECTION IN OUTDOOR IMAGES

An assessment of methods

by
Heiko Heijenga

Supervisors:
Dr. ir. J.A.G. Nijhuis
Drs. J. Spiekstra

The detection of vehicles is an important task in traffic monitoring and video surveillance. Traditional non-visual based methods to detect vehicles are often too expensive in maintenance and installation. They also cannot be deployed in every situation due to physical limitations.

A visual vehicle detection system is on the other hand very flexible; it can be installed in nearly every situation and the costs to set up a camera are relative low compared to that of traditional detection systems. However, the detection of objects in digital images is anything but a trivial task. This becomes even more of a problem when objects need to be detected in outdoor images where lighting conditions are unpredictable and constantly changing. Often there are also other objects (i.e. non-vehicles) present in the monitored area which could lead to false detection of vehicles.

A visual vehicle detection system therefore needs a couple of requirements to be stated as robust. In this study an assessment will be made on these requirements and on current technologies and methods to detect vehicles in digital images.

TABLE OF CONTENTS

Chapter 1 Introduction

The detection of vehicles is an important task in traffic analysis and surveillance. The demand for automatic detection systems is ever increasing because the number of vehicles as well as the number of roads is rapidly increasing. Most traffic information systems rely on a variety of sensors for determining parameters of interest. Magnetic loop detectors are currently the most used sensors for detecting passing vehicles. A vehicle passing over one of those loops results in a small current which is used as a signal to the attached detection system. The problem with these induction loops is that a road needs to be closed down when these loops are installed or when a loop is broken, this is not only a very expensive operation but it also results in considerable problems concerning traffic flow. A system that has less impact on the throughput of traffic and is cheaper in installation and maintenance is desirable. Over the past couple of years extensive research is done on systems that make use of fixed video cameras that monitor places where vehicles need to be detected. Vision-based video monitoring systems offer many advantages. Video cameras are relative easy to install and have no disruptive effects on traffic when they are installed or when they need maintenance. Vehicle detection by means of video cameras is also very useful in places covering a large area where it is impractical to use loops in the road like on a parking lot. Detecting vehicles is not the only task such a vision-based system could perform. A much broader spectrum of parameters could be extracted from such a system. Vehicles could be classified according to their shape, lane changes could be detected, vehicles can be tracked, etc.

Research done in the past and present has resulted in many methods and techniques for detecting stationary as well as moving vehicles.

1.1 Problem statement

Detecting non-moving and moving objects has been and still is a great subject in computer vision research. Many methods have been studied and proposed to detect objects from images. We are especially interested in the detection of vehicles in unconstrained outdoor images. There are many articles (e.g. [1], [2], [3]) in which different methods are proposed for this application of object detection. Most approaches have difficulties when there is a great variation in lighting conditions in the outdoor scene especially when many shadows are present. Another difficulty is when the scene consists not only of objects that need to be detected but also contain other objects that are not of interest. The vehicle detection method therefore needs to distinguish between the objects of interest and those that are not.

It is also likely that in some situations multiple vehicles are present in the current frame which leads to the possibility that a vehicle could be blocked by another vehicle what makes it difficult to properly detect the individual vehicles.

Another point where problems arise is when the supposedly fixed video camera is in fact not completely stationary. When a camera is for example mounted on a pole it is possible that due to the wind the camera swings in every direction. This ego motion of the camera needs to be considered when developing a vehicle detection system that makes use of such cameras and must detect moving vehicles.

Finally it is desirable that the detection of vehicles can be done completely autonomous and in real-time, though this is not a subject in this study.

1.2 Requirements

Summarizing all the considerations mentioned is the previous section; a vehicle detection system can be stated as robust when it has the following characteristics.

- Only vehicles are detected
- Can deal with various ambient lighting conditions and shadows
- Vehicles in complex scenes can be detected
- Is able to detect partial occluded vehicles
- Multiple vehicles in a single scene can be detected
- Movement of the camera can be dealt with
- Vehicles can be detected in real-time

1.3 Research question

What approaches are possible to reliable detect vehicles by using a fixed stationary video camera, taking into account the problems mentioned above?

I narrow down my research by only looking at regular motorized vehicles on four or more wheels. The algorithm has to hold the characteristics mentioned in the previous section with the exception of the real-time part. Whether an algorithm can execute in real-time depends highly on the used hardware and the optimizations in the code, this is beyond the scope of my research.

I'm trying to answer this question by examining existing methods and techniques to figure out their suitability for the detection of vehicles taking into account the problems mentioned in the previous section.

To make a comparison between all the methods mentioned in this study I will create at the end of each described method a table (see Table 1) in which the requirements of section 1.2 will be evaluated.

Table 1. Evaluation table.

1. Vehicles are detected in "vehicles-only" situations	rating
2. Only vehicles are detected in complex situations	rating
3. Changes in ambient lighting conditions can be handled	rating
4. Partial occluded vehicles can be detected	rating
5. Multiple vehicles can be detected	rating
6. Movement of the camera can be dealt with	rating

The points on which the evaluations are done are:

- ***Vehicles are detected in "vehicles-only" situations***
  With a "vehicles-only" situation I mean that the observed situation contains just vehicles. A highway viewed from an overhead camera is such a situation (see Figure 1) because the only large moving objects that are to be expected are vehicles.

Figure 1. A "vehicles-only" situation.

- ***Only vehicles are detected in complex situations***
  Complex situations are situations where not only vehicles are present but also other moving objects like people or motorcycles (see Figure 2).

Figure 2. A busy road in Bangkok.

- ***Changes in ambient lighting conditions can be handled***
  Changes in intensity resulting from, for example, different weather conditions should have no impact on the detection performance. This also includes shadows being cast by vehicles and other objects.

- ***Partial occluded vehicles can be detected***
  Vehicles that are not completely visible should also be detected. The vehicles can be partially occluded by other vehicles or by other objects.

- ***Multiple vehicles can be detected***
  The detection system must be able to detect vehicles even when there are multiple vehicles present in the image.

- ***Movement of the camera can be dealt with***
  A moving camera resulting from, for example, the blowing wind may not affect the detection performance. This won't be a problem when we are merely looking at still images but it's definitely a thing to consider when using video images.

The evaluation points are rated with +'s (good) and –'s (bad) according to the information available in the article(s) or otherwise to my own knowledge and sense.

Chapter 2 Applications

The number of visual vehicle detection systems that are installed to observe some scenery is constantly increasing. This is because of various reasons; it is for example generally less expensive to let a computer attached to a camera monitor an environment than to let a person continuously watch a monitor attached to the camera. In addition, since hardware becomes faster and cheaper everyday it is becoming more and more possible to create an inexpensive and robust autonomous visual vehicle detection system.

2.1 Vehicle identification

There are numerous systems that identify passing vehicles by reading their license plate. Before a system can read and recognize a plate it must detect the vehicle. Because the recognition stage uses a digital image from a camera it would be convenient if the vehicle itself is also detected using the same camera. This results in a system that is completely independent on external equipment such as loop detectors.

Such a system can for example be used to control the access to a parking area in the Intrada Parking system developed by Dacolian [4].

Figure 3. Access control situation where a vehicle is detected and identified by its license plate.

2.2 Traffic Analysis

One of the most used applications for vehicle detection is that for traffic analysis ([5], [6]). Roads are becoming more crowded with vehicles everyday. It is important to let the traffic flow unhindered over long distances. To achieve this it is necessary to know for example how many vehicles are currently present on the road and how fast they are going. Traffic lights should be adjusted according to the traffic parameters recovered from the detection system to accomplish a steady traffic flow. A visual traffic analysis system is

very well suited for this task as it can detect vehicles and track them to determine the required parameters.

An example situation is measuring traffic parameters at an intersection. A interesting parameter can for example be the number of vehicles going in a certain direction. Based on this information one can decide if the intersection can cope with the amount of traffic that passes.

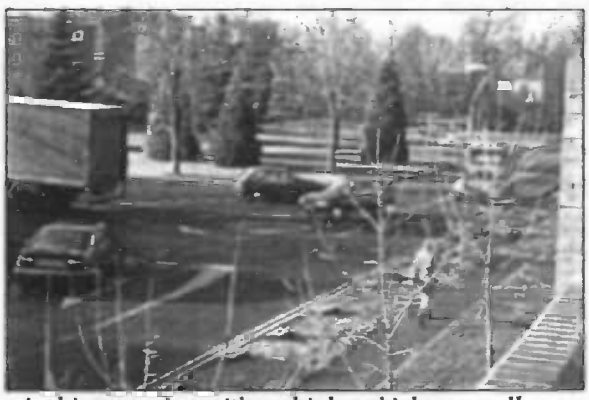


Figure 4. A typical intersection with multiple vehicles as well as one moving person. The scene also contains occluded vehicles and several shadows cast by vehicles as well as by other objects.

2.3 Incident detection

Automatic determining the speed and direction of a vehicle can provide detection of a possible incident. An incident can for example be detected if the speed of the vehicle drops suddenly to zero, when the vehicle changes suddenly to an unreasonable direction or when a vehicle is detected in an area where it may not stop such as on a road verge. A possible incident can also be detected when the speed of the overall traffic drops suddenly which could be an indication of an incident further down the road. Appropriate actions can be taken subsequently when an incident is detected. An example of such a system is the VIP/I Incident Monitor developed by Traficon [7].

Figure 5. A vehicle stops in the detection area, this is noticed within seconds.

2.4 Vehicle classification

A vehicle detection system that is also capable of classifying the detected vehicles could be very useable [8]. It is often necessary to know what kind of traffic is passing some point; a vehicle can for example be a normal car, a truck or a bus. Using this information it is possible to determine the travel activity by different types of vehicles on a road. This can be useful information when new roads (with the same traffic composition) need to be made or when an existing road needs to be repaired.

Another application is toll collecting. The amount of toll that needs to be paid on a toll road depends often on the type of vehicle. With a vehicle classification system the vehicle type can be determined automatically.

Figure 6. A typical toll road.

Chapter 3 Video-based vehicle detection

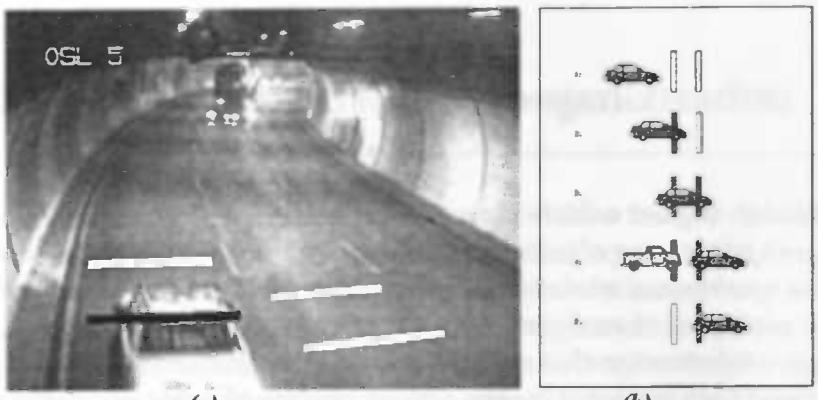
Recent vehicle detection systems use video technology and digital image-processing algorithms to detect significant changes in image sequences. The computational resources currently available make it possible to analyze video frames in real-time. It is therefore reasonable to make use of the extra information that is present in image sequences compared to still images. In video frames it is very easy to analyze changes occurring in consecutive frames. Various algorithms are available to detect these changes and they all have in common that they make explicit use of image sequences rather than of still images to detect vehicles. This implies that these systems are not able to detect a vehicle if presented with a single image and almost in every case cannot detect stationary vehicles.

In this chapter I will present several methods used that detect the significant changes in image sequences which could be the result of a passing vehicle.

3.1 Virtual trigger lines

For many years it was common to use induction loops embedded in the road to detect vehicles. Actually this is still the most used method for simple vehicle detection. The problem with these loops is, as mentioned before, the cost to install them and the disruptive side effects they have on the traffic when being installed or repaired. An induction loop generates a signal caused by a disturbance of the magnetic field when a vehicle drives over it, this signal can be used to detect vehicles driving over the loop.

These loops can also be implemented as virtual loops in a visual vehicle detection system [9]. The main task of the presented method is to count the passing traffic to estimate the density of the traffic. In the first stage of the algorithm low-level features are extracted from predefined regions in each video frame. The second stage consists of a pre-classification on each frame independently after which the resulting sequence is used as an input to a classification algorithm using a hidden Markov model (HMM).



(a) (b)

Figure 7. Trigger lines; (a) detection windows across two lanes, (b) activation of the loops when vehicles pass by.

The presented method uses two detection windows positioned across the road, see Figure 7. In an initialization step where no vehicles are present an estimate is made about the background grey levels and edges in the detection window. Now when a vehicle enters a detection window this is noticed by a significant change in grey levels and edges. To be eligible to be classified as a vehicle there must be another significant change in the detection window in a predefined time interval; i.e. when the vehicle is leaving the detection window. When that second change is taking to long then the current grey levels and edges are most likely occurring from changes in the lighting conditions and are therefore now considered as the current background.

In the preprocessing stage the detection windows are assigned a state; a detection window is either occupied (1) or unoccupied (0). The sequence of states of the two detection windows when a vehicle is passing is used as a two-dimensional feature. This feature is further analyzed by an algorithm using a HMM. The HMM is used because the sequence of states is not completely perfect because of noise and other external circumstances. For a complete description and implementation of the algorithm and its usage of the HMM I refer to [9]. The basic idea is that the sequence of states is analyzed resulting in a decision about the presence of a vehicle.

Experiments were done in a situation where traditional loop detectors are already present, giving the opportunity to compare the results. The situation where the experiments took place was a three lane road where approximately 15.000 vehicles passed per day. Daylight as well as nighttime situations were used. From a 200 day test the researches examined three different periods with different weather conditions in those periods.

- Summer: 7 days and nights in July
- Autumn: 7 days and nights in October
- Winter: 4 days and nights in January and February

The results were compared to the ground truth, i.e. the results from the traditional loop detectors; in the summer a deviation (above or below the ground truth) of 1.7% was found, in the autumn the deviation was 1.9% and in winter it was 2.2%.

This method shows a very good performance, even in challenging lighting situations. The performance is highly dependable on predictable traffic as is monitored in the test situation. Traffic monitored in the test situation is always going in the same direction and it is very unlikely that other objects (i.e. non-vehicles) are to be expected in the scene.

Vehicles are detected in "vehicles-only" situations
When the traffic isn't too heavy then this method will perform good. But the HMM will fail to detect single vehicles when many vehicles are driving head-to-tail.

Only vehicles are detected in complex situations
The article doesn't mention the effect non-vehicles have on the system. My own idea is that also non-vehicles such as people will be detected as vehicles.

Changes in ambient lighting conditions can be handled
The system in the article uses a background estimation algorithm to handle changing light conditions.

Partial occluded vehicles can be detected
A partial occluded vehicle won't result in the state transitions in the HMM that are expected. As a result partial occluded vehicles won't be detected.

Multiple vehicles can be detected
Multiple vehicles will only be detected when there are multiple detection areas set-up.

Movement of the camera can be dealt with
Movement of the camera will definitely result in the detection windows be activated. This will result in the failure of detecting vehicles.

Table 2. Performance evaluation of virtual trigger lines.

1. Vehicles are detected in "vehicles-only" situations	+
2. Only vehicles are detected in complex situations	-
3. Changes in ambient lighting conditions can be handled	+
4. Partial occluded vehicles can be detected	-
5. Multiple vehicles can be detected	-
6. Movement of the camera can be dealt with	-

Chapter 4 Moving object segmentation

The most common approach to detect objects in video images consists of two largely independent stages; a segmentation step that segments the foreground objects (i.e. moving objects) from the background and a second step where some higher level reasoning is carried out to identify and classify the moving objects. In this chapter I will present several algorithms for the first step. The segmentation of moving objects needs to be effective in complex scenes where lighting can change significantly and where not only objects of interest (i.e. vehicles) are present, but also other objects. Vehicles are also not restricted to a certain direction, speed or trajectory. Taking all these considerations into account it can be seen that the segmentation of vehicles from the background is not a trivial task at all.

Moving object segmentation is based on the extraction of particular visual and/or motion features. The extraction of static visual features such as texture, color and edge is often not suitable in outdoor images since images contain noise, non interesting objects/regions and other distractions. So we often can not rely just on using static visual features. We therefore use motion features to segment the moving objects from the background.

The following sections give an overview of available methods that extract a moving visual object from a sequence of images. These methods are used as a preprocessing step for algorithms that analyze the extracted MVOs[1] to determine whether it is a vehicle.

4.1 Frame differencing

The most simple and possibly fastest technique available for detecting motion in a sequence of images is to calculate the difference between two consecutive frames [10]. Pixels, whose gray-value changes in two consecutive frames become visible in the difference image whereas pixels that don't change or change very little won't show in the difference image, see Figure 8.

Consider the sequence of video frames I_n : then the difference per pixel between consecutive frames is defined as:

$$D_n(i,j) = \left| I_n(i,j) - I_{n-1}(i,j) \right|$$

[1] Moving visual objects

In [10] the authors pointed out that this method is very sensitive to image noise. They therefore adopted the double-difference image where a logical AND operator is performed between two consecutive difference images and thresholded by T. The double-difference operation is defined as:

$$DD_n(i,j) = \begin{cases} 1 \text{ if } D_{n+1}(i,j) > T \wedge D_n(i,j) > T \\ 0 \text{ otherwise} \end{cases}$$

This double-difference showed to be quite immune to image noise compared to the single-difference. This is explained by the fact that image noise is often non-repeatedly.

(a) (b)

Figure 8. Frame differencing; (a) current frame, (b) difference between current and previous frame.

So for segmenting moving objects from a video sequence this method is performing well enough. The major drawback of this technique is that the segmented object is divided into multiple copies of the object as can be seen in Figure 8b. This effect is even worse when the vehicle is moving fast and/or the frame rate of the video is low. It also can be seen that when the vehicle stops for a moment it is impossible to detect it with frame differencing.

Although this method won't classify segmented objects as being vehicles, nevertheless I will give an evaluation of the requirements mentioned in 1.2.

Vehicles are detected in "vehicles-only" situations

This method will segment vehicles in these situations without problems.

Only vehicles are detected in complex situations

In complex situations with moving non-vehicles this method will obviously not segment only vehicles. Therefore this method is not useable in complex situations

Changes in ambient lighting conditions can be handled

Changing intensity of all pixels in the image will result in the whole image being classified as moving object.

Partial occluded vehicles can be detected

As long as the partial occluded vehicle is moving it will be detected.

Multiple vehicles can be detected

When multiple vehicles in the image are present and moving they all will be segmented.

Movement of the camera can be dealt with

A moving camera will definitely give problems here. The result of a moving camera is that every pixel in the image will be labeled as a moving object.

1. Vehicles are detected in "vehicles-only" situations	+
2. Only vehicles are detected in complex situations	-
3. Changes in ambient lighting conditions can be handled	-
4. Partial occluded vehicles can be detected	+
5. Multiple vehicles can be detected	+
6. Movement of the camera can be dealt with	-

4.2 Background subtraction

A widely used method to detect objects is by comparing the current frame with a background image on a pixel-by-pixel base. A background image should be clear of any objects that need to be detected. Subtracting the background from the current image results in an image where only the parts of the image show that are different than the background. Consequently an object that entered the field of view of the camera would be segmented from the background. The effect of noise in the image is often eliminated by only marking pixels whose value changes in consecutive images where the difference is greater than some preset threshold.

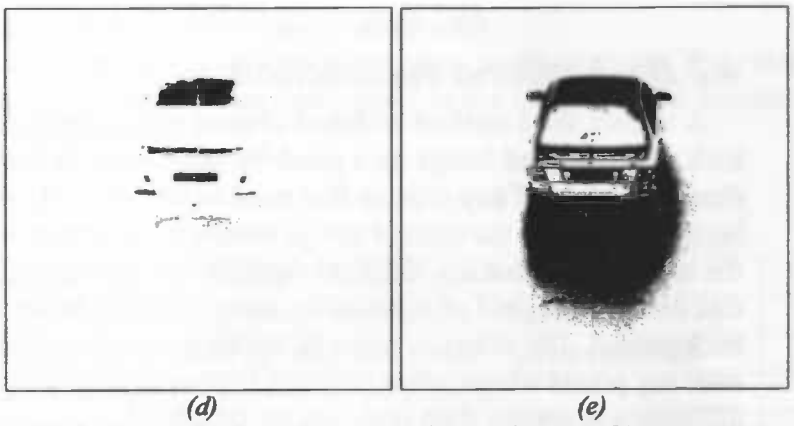
The process to create a binary image mask that shows all pixels where the difference between the gray values is large enough is shown below. The input image is denoted by **I**, the background image by **B** and the threshold by **T.**

$$D_n(i,j) = \begin{cases} 1 \text{ if } |B(i,j) - I(i,j)| > T \\ 0 \text{ otherwise} \end{cases}$$

(a)

(b)

(c)



(d) (e)

Figure 9. Background subtraction; (a) background image, (b) current image, (c) image mask where threshold was to low, (d) threshold to high, (e) acceptable image mask.

Creating a difference image that is of any use is largely dependent of the chosen threshold value. Choosing the threshold to low results in many pixels being misclassified as motion due to image noise, a value to high can result in an object being missed. An example of background subtraction with different thresholds is shown in Figure 9.

The reliability of this method greatly depends on a good representation of the background image. In an indoor scene this is not a great problem because the background doesn't change. Outdoor sceneries are on the other hand continuously changing because of changes in illumination or a physical change in the background, like a tree in the wind. To get a representative background it is important to have an algorithm that creates and updates an appropriate representation of the background image. For example, light intensity changes in the scene occurring from a cloud moving over should be handled by the background updating algorithm.

Research done in the field of background subtraction consists mostly of developing a good background updating algorithm which can deal with the problems mentioned before. The key requirements for a good algorithm are:

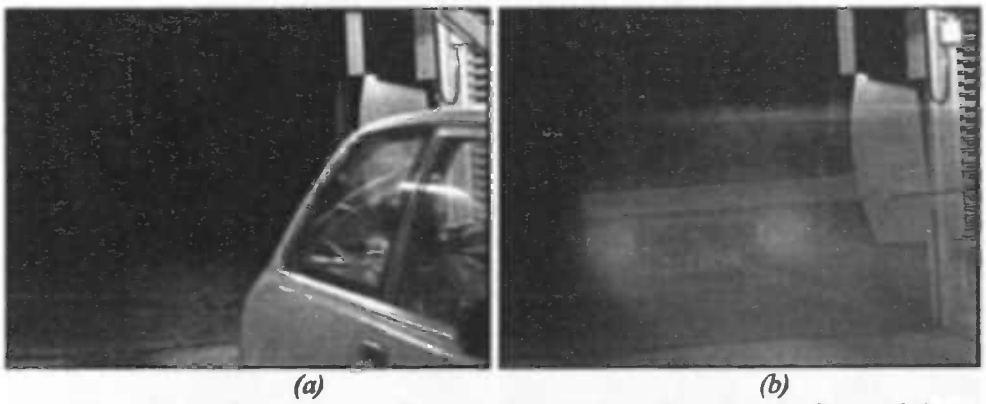
- Can deal with changing light intensities
- Background image can be adopted fast to the current background
- Is able to deal with camera motion

Frame averaging

A simple and commonly used approach to background updating is the averaging technique [11]. This simply implies that the background is represented as the average of a number of frames from the past. To calculate this average it is necessary to store the frames that need to be averaged in memory. The computational load to store all the frames and calculate the average is relatively large. A better method to calculate the average is to use the exponential averaging equation [12]. The averaging over $N+1$ frames is done with the following equation:

$$B_t = k \cdot B_{t-1} + (1-k) \cdot C_{t-1} \qquad 0 < k < 1 \quad \wedge \quad k = N/(N+1)$$

Where B_t is the updated background, B_{t-1} the previous background and C_{t-1} the previous frame. The rate at which the background is updated is determined by k. When the speed at which the background is updated is to high then it is possible that slowly moving vehicles become part of the background, see Figure 10b. When the speed on the other hand is to low then the background is not adjusted fast enough to lighting changes. The effectiveness of this background updating algorithm is thus highly dependable on the chosen update speed.



(a) (b)

Figure 10. Background averaging; (a) current frame, (b) current background showing a ghost vehicle blended in from previous frames.

The evaluation of frame averaging method in combination with background subtraction is listed below.

Vehicles are detected in "vehicles-only" situations

This method will segment vehicles in these situations when the vehicles are driving at a constant speed. Otherwise vehicles could get blended into the background as in Figure 10.

Only vehicles are detected in complex situations

In complex situations with moving non-vehicles this method will obviously not segment only vehicles. Therefore this method is not useable in complex situations

Changes in ambient lighting conditions can be handled

Because of the averaging, changes in lighting condition won't have a big impact on the background subtraction

Partial occluded vehicles can be detected

As long as the partial occluded vehicle is moving it will be detected.

Multiple vehicles can be detected

When multiple vehicles in the image are present and moving they all will be segmented.

Movement of the camera can be dealt with

A moving camera will definitely give problems here. The result of a moving camera is that every pixel in the image will be labeled as a moving object.

1. Vehicles are detected in "vehicles-only" situations	-/+
2. Only vehicles are detected in complex situations	-
3. Changes in ambient lighting conditions can be handled	+
4. Partial occluded vehicles can be detected	+
5. Multiple vehicles can be detected	+
6. Movement of the camera can be dealt with	-

Selective updating

One way to get rid of the moving objects that get blended into the background is to only update the background image in the regions where no motion is detected [12]. The first step in this method is to decide which pixels are classified as moving points. The moving points are subsequently excluded from the updating process. Pixels where no motion was detected are replaced with the corresponding pixels in the current image.

$$B_t(i,j) = \begin{cases} B_{t-1}(i,j) \text{ if } |I_t(i,j) - B_{t-1}(i,j)| > T \\ I_{t-1}(i,j) \text{ otherwise} \end{cases}$$

(a) (b)

Figure 11. Selective updating; (a) current frame, (b) current background with corrupted image regions.

This method doesn't suffer from ghosts from slow moving objects blended in with the averaging technique. But for this method to be successful a correct value for the threshold needs to be determined, otherwise pixels get classified incorrectly. Overall this method is not really useful because the background can be corrupted really fast like in Figure 11b and changes in consecutive frames result in a quick change of the background image which isn't really desirable. The result of this is that after some time the representation of the background is so corrupted that it can't recover to a proper representation of the background because of all the clutter.

This background updating technique performs not good when looking at the requirements. This is the result of the background getting dirty really fast. On almost every point in the requirements this method consequently gets a negative evaluation.

1. Vehicles are detected in "vehicles-only" situations	-
2. Only vehicles are detected in complex situations	-
3. Changes in ambient lighting conditions can be handled	+
4. Partial occluded vehicles can be detected	-
5. Multiple vehicles can be detected	-
6. Movement of the camera can be dealt with	-

Selective updating with averaging

In [11] the author describes a new updating technique that uses a combination of the two previous mentioned methods. It uses the grey level changes in two consecutive frames as a measure of the ambient lighting variations. If this difference is less than some threshold then the background is updated. The update process doesn't simply replace the current background pixels with the current image but it replaces them with an average. The function to describe this method is:

$$B_t(i,j) = \begin{cases} \dfrac{\left(B_{t-1}(i,j) + I_{t-1}(i,j) + 1\right)}{2} & \text{if } \left|I_{t-1}(i,j) - B_{t-1}(i,j)\right| < T_1 \wedge \left|I_{t-1} - I_t\right| < T_2 \\ B_{t-1}(i,j) \text{ otherwise} \end{cases}$$

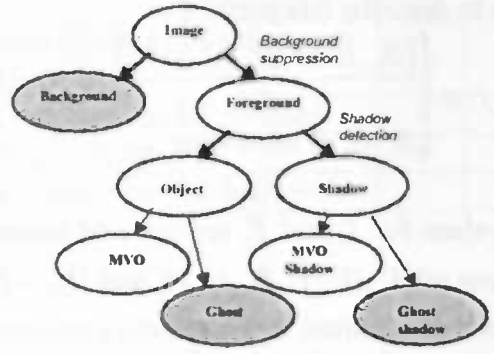
The values for T_1 and T_2 are selected automatically by analyzing the histograms of $\left|I_{t-1}(i,j) - B_{t-1}(i,j)\right|$ and $\left|I_{t-1} - I_t\right|$ for a number of frames. The problems that occurred in the selective updating algorithm in the previous section are greatly reduced with this technique.

Compared to the previous method this method has a good performance when we look at the requirements. The background won't get dirty quick which results in a positive evaluation of most of the requirements.

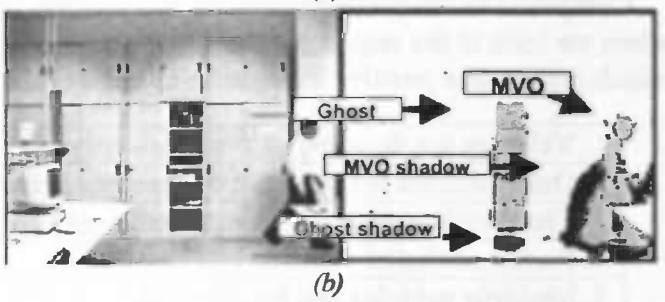
1. Vehicles are detected in "vehicles-only" situations	+
2. Only vehicles are detected in complex situations	-
3. Changes in ambient lighting conditions can be handled	+
4. Partial occluded vehicles can be detected	+
5. Multiple vehicles can be detected	+
6. Movement of the camera can be dealt with	-

Excluding moving objects in background updating

Cucchiara and others introduced in [13] a new technique that solves the problem of deadlock that can occur with the selective background updating algorithm described in the previous section. They use a combination of averaging over the past frames and the knowledge of detected moving objects. The objects that are detected as moving are divided into four categories; moving visual objects (MVOs), shadows of MVOs, objects that aren't really moving objects like a opening cabinet door (ghosts) and the shadows of ghosts.



(a)



(b)

Figure 12. (a) Object classification, (b) detected regions in an indoor scene.

The proposed function of this method is the following:

$$B_{t+\Delta t}(x,y)=\begin{cases} B_t(x,y) & \text{if } (x,y)\in\{\text{MVO}\}\cup\{\text{MVO shadow}\} \\ f\left(C_t(x,y),C_{t-\Delta t}(x,y),\ldots,C_{t-n\Delta t}(x,y),w_b B_t(x,y)\right) & \text{if } (x,y)\in\{\text{BKG}\}\cup\{\text{ghost}\}\cup\{\text{ghost shadow}\} \end{cases}$$

Where *f is* the median function over the past *n* frames. The difficulty with this approach is that each pixel in the current image needs to be correctly classified. Only points that belong to a true moving object or its shadow (an object whose average optical flow is greater than some threshold) are excluded from the background updating. This method overcomes most problems of background corruption but the issue of correctly classifying each pixel is now the greatest problem. The researchers use several rules that classify the regions in the image; these rules are based on area, saliency and motion. The complete control flow path can be seen in Figure 13.

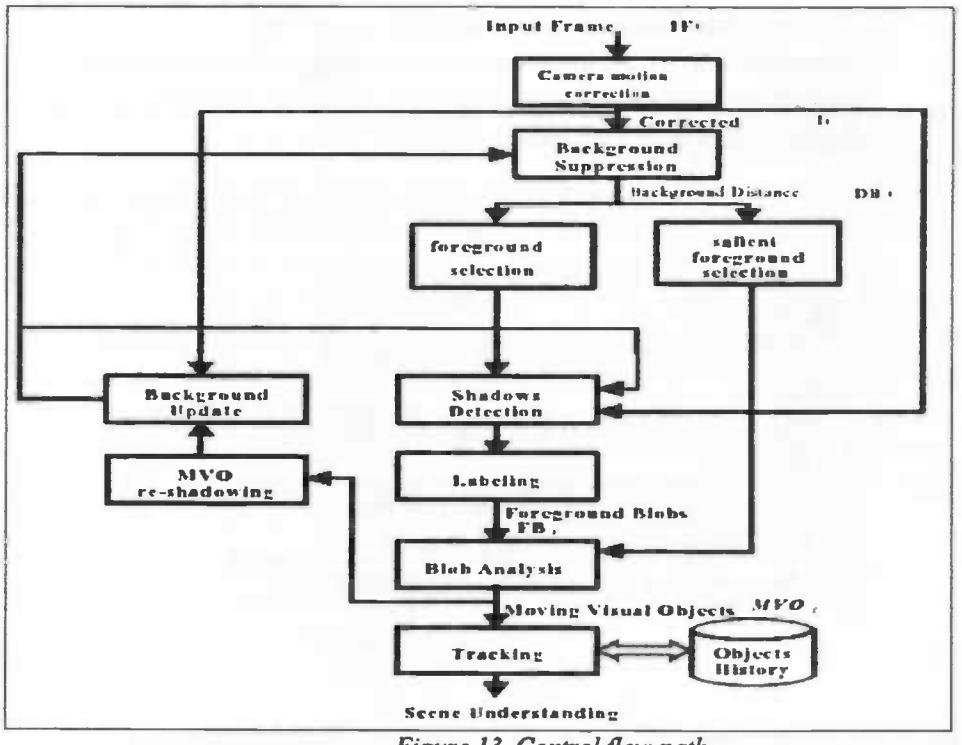


Figure 13. Control flow path.

This method performs better than the previous method on the following point.

Only vehicles are detected in complex situations

Because the moving objects are labeled using this method it will be possible to distinguish between vehicles and non-vehicles. Labeling the moving objects is unfortunately anything but trivial. But when we assume that it is possible then this method will score well at this point.

As we have seen there are various algorithms available to generate a representation of the background for background subtraction. The performance evaluation of background subtraction with the different background updating algorithms is summarized in Table 3.

Table 3. Performance evaluation of background subtraction.

	FA[2]	SU[3]	SA[4]	EU[5]
1. Vehicles are detected in "vehicles-only" situations	-/+	-	+	+
2. Only vehicles are detected in complex situations	-	-	-	+
3. Changes in ambient lighting conditions can be handled	+	+	+	+
4. Partial occluded vehicles can be detected	+	-	+	+
5. Multiple vehicles can be detected	+	-	+	+
6. Movement of the camera can be dealt with	-	-	-	-

[2] Frame averaging
[3] Selective updating
[4] Selective updating with averaging
[5] Excluding moving objects in background updating

4.3 Kalman filtering

Karmann [14] and others [15, 16] described the change of gray values of individual pixels in frame sequences as a signal processing system. They used a Kalman filter in their background updating algorithm to quickly adjust the background when illumination changes while slowly adjusting regions with moving objects.

A Kalman filter makes a prediction of the next state of the system based on previous states. This prediction is assumed to be the best state of a system. The filter subsequently compares the actual measured value with the predicted value and adjusts the estimation by weighing the difference between the measured value and the predicted value. Measured values that are not in correspondence with the system behavior consequential get a lower weight.

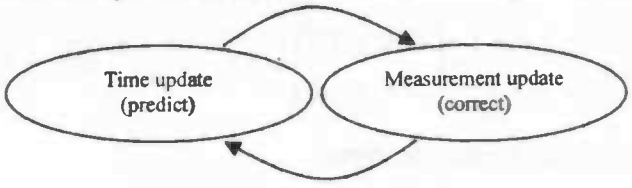


Figure 14. The Kalman filter cycle; the time update predicts the next state and the measurement update adjusts the estimates according to the measured value.

The estimation of a system state at time t is

$$\hat{x}_t = \tilde{x}_t + K_t \cdot [z_t - H_t \cdot \tilde{x}_t]$$

with a prediction term

$$\tilde{x}_t = A_t \cdot \hat{x}_{t-1}$$

Where K_t is the Kalman gain matrix, H_t is the measurement matrix, z_t is the system input and A_t is the system matrix. In [Ridder95] the authors applied the Kalman filter to background updating. The system input are the gray value of the pixel at location (x,y) at time t_i, denoted by $s(x, y, t_i)$. The estimation of the system state at time t_i is $\hat{s}(x, y, t_i)$ and the estimated variety is $\hat{\dot{s}}(x, y, t_i)$. The Kalman filter equation then becomes

$$\begin{bmatrix} \hat{s}(x,y,t_i) \\ \hat{\dot{s}}'(x,y,t_i) \end{bmatrix} = \begin{bmatrix} \tilde{s}(x,y,t_i) \\ \tilde{\dot{s}}(x,y,t_i) \end{bmatrix} + K(x,y,t_i) \cdot \left(s(x,y,t_i) - H(x,y,t_i) \cdot \begin{bmatrix} \tilde{s}(x,y,t_i) \\ \tilde{\dot{s}}(x,y,t_i) \end{bmatrix} \right)$$

with the prediction term

$$\begin{bmatrix} \tilde{s}(x,y,t_i) \\ \tilde{\dot{s}}(x,y,t_i) \end{bmatrix} = A \cdot \begin{bmatrix} \hat{s}(x,y,t_{i-1}) \\ \hat{\dot{s}}(x,y,t_{i-1}) \end{bmatrix}$$

The measurement matrix H as well as the system matrix A is constant.

$$H = [1 \quad 0] \qquad A = \begin{bmatrix} 1 & a_{1,2} \\ 0 & a_{2,2} \end{bmatrix}$$

The value for $a_{1,2}$ and $a_{2,2}$ are set to 0.7 to represent the background dynamics. The Kalman gain matrix is

$$K(x,y,t_i) = \begin{bmatrix} k_1(x,y,t_i) \\ k_2(x,y,t_i) \end{bmatrix} = \begin{bmatrix} \alpha \cdot m(x,y,t_{i-1}) + \beta \cdot (1 - m(x,y,t_{i-1})) \\ \alpha \cdot m(x,y,t_{i-1}) + \beta \cdot (1 - m(x,y,t_{i-1})) \end{bmatrix}$$

where $m(x,y,t_{i-1})$ tells whether the pixel at location (x,y) belongs to the foreground or to the background at time t_{i-1}

$$m(x,y,t_{i-1}) = \begin{cases} 1 & \text{if } d(x,y,t_{i-1}) \geq th(x,y,t_{i-1}) \\ 0 & \text{otherwise} \end{cases}$$

with

$$d(x,y,t_{i-1}) = |s(x,y,t_{i-1}) - \hat{s}(x,y,t_{i-1})|$$

and $th(x,y,t_i)$ is some fixed threshold. From these equations it can be seen that when the difference between the measured background and the estimated

background in greater than the threshold the system uses α as the gain factor and β otherwise. A properly chosen threshold results in a fast adaptation to the estimated background where the intensity changes are small while ignoring large differences resulting from foreground objects appearing in an image sequence.

When looking at the requirements of section 1.2 we will see similar performance as with the averaging with selective updating technique. The use of a Kalman filter is only another method for estimating a background.

1. Vehicles are detected in "vehicles-only" situations	+
2. Only vehicles are detected in complex situations	-
3. Changes in ambient lighting conditions can be handled	+
4. Partial occluded vehicles can be detected	+
5. Multiple vehicles can be detected	+
6. Movement of the camera can be dealt with	-

4.4 Optical flow

When the vehicles that need to be detected are moving and the frame rate of the captured video is high enough, than we can use a method relying on optical flow measurements to detect motion [17]. Optical flow calculation is performed by determining the displacement between regions from the same object in consecutive images. When there is a clustering of non-null optical flow vectors in a certain region of the image there is probably a moving object in that region. With this method it also is possible to distinguish between rigid and non-rigid motion.

The optical flow vectors of a moving vehicle are almost the same (see Figure 16) while a moving person has different optical flow vectors (see Figure 15) because the legs for example don't have the same velocity with respect to each other.

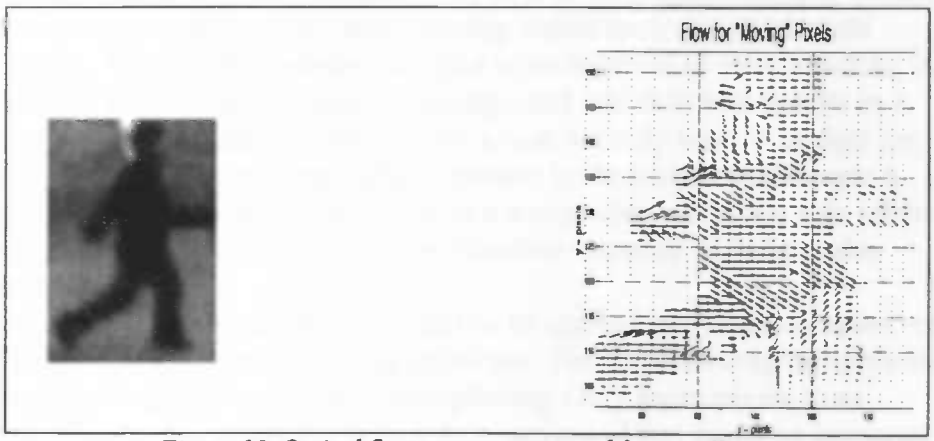


Figure 15. Optical flow vectors computed for a walking person.

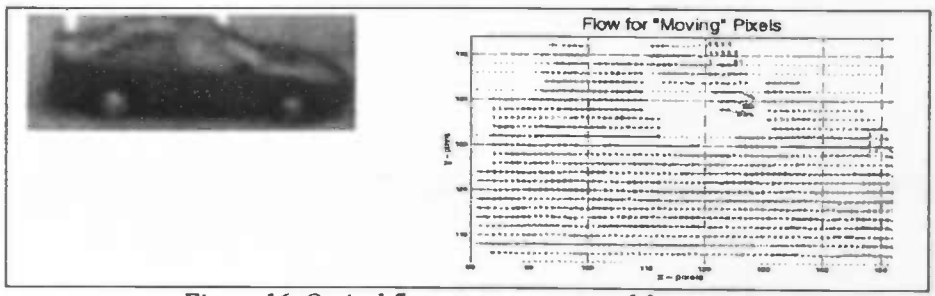


Figure 16. Optical flow vectors computed for a moving car.

Large clusters of flow vectors indicate a single object. Whether this object is indeed a vehicle must be decided by a second stage in the algorithm. So this method is only useable to identify regions of interest.

This method has severe problems when used with a not completely stable video camera. When the camera only swings a little this leads to optical flow vectors all over the image. This effect can be eliminated by using an algorithm that stabilizes the video stream.

The evaluation of the optical flow technique is listed below.

Vehicles are detected in "vehicles-only" situations

In "vehicles-only" situations we can extract vehicles very good.

Only vehicles are detected in complex situations

The only distinction that can be made is whether a moving object is rigid or non-rigid, e.g. vehicle vs. person.

Changes in ambient lighting conditions can be handled

Ambient lighting changes will result in motion vectors all over the image.

Partial occluded vehicles can be detected

As long as the partial occluded vehicle is moving it will be detected.

Multiple vehicles can be detected

When multiple vehicles in the image are present and moving they all will be segmented.

Movement of the camera can be dealt with

A moving camera will definitely give problems here. The result of a moving camera is that every pixel in the image will be labeled as a moving object.

1. Vehicles are detected in "vehicles-only" situations	+
2. Only vehicles are detected in complex situations	+/-
3. Changes in ambient lighting conditions can be handled	-
4. Partial occluded vehicles can be detected	+
5. Multiple vehicles can be detected	+
6. Movement of the camera can be dealt with	-

4.5 Overview and considerations

The methods mentioned in this chapter make in possible to extract a moving object from a sequence of images. These techniques are therefore very useful to detect motion. In many visual vehicle detection systems the segmentation of moving objects is a crucial preprocessing stage. The segmentation algorithms do not, however, identify the moving object as being indeed a vehicle. As a result these methods alone are not suitable to detect only vehicles.

The success of a background subtraction technique to extract foreground objects is highly dependent on the algorithm used to acquire the background image. I have described several methods found in literature that derive a background image from the current and past images. The simplest method was the frame averaging technique. This technique can be used when the scene where the images are taken consists only of a small number moving vehicles with a constant speed and no other moving object such as on a straight motorway. We run into problems when a vehicle drives to slow resulting in the vehicle getting blended into the background which in turn results in a failure in extracting the vehicle. In such a case we only want to update the background where there is no vehicle present in the image. In this case a selective updating technique suffices. In a scene where there are also other moving objects besides the vehicles a selective updating technique also doesn't work.

When background subtraction needs to be applied in a more complex scene it is better to use a more advanced technique. The frame averaging method can be extended with a form of selective updating. Only those pixels from consecutive frames that don't belong to a moving object could be averaged to minimize the effect of objects that get blended into the background image.

Chapter 5 Moving object classification

The moving object segmentation techniques presented in the previous chapter classify regions in the image as being a moving object. These techniques are not able to classify the ROI[6]s, so we need to perform additional actions to determine whether the segmented object is indeed a vehicle. In this chapter I present several algorithms that perform this task. They depend on a correct segmentation of the object.

5.1 Bounding boxes

Looking at the size of the ROI is the most trivial method to determine the presence of a vehicle. The ROI can for example be enclosed by a bounding rectangle. When all the vehicles are pointing in approximately the same direction then the size of the bounding box would be approximately the same for every vehicle. This is because most regular vehicles are approximately the same size. Other objects such as people would enclose a much different sized bounding rectangle.

For this method to be successful it is necessary that the bounding box fits the vehicles exactly and no shadows or other objects are enclosed with the vehicle. The vehicles also need to be covered completely by the field of view of the camera.

How this method performs with respect to the requirements can be seen below.

Vehicles are detected in "vehicles-only" situations

This method will segment vehicles in these situations without problems if the ROI contains only the vehicle and no shadows.

Only vehicles are detected in complex situations

It depends of the size of the non-vehicles that can be expected in the situation in question whether only vehicles will be detected. If there are objects which have the same size as a vehicle than it will of course be classified as a vehicle.

Changes in ambient lighting conditions can be handled

Shadows will give problems.

Partial occluded vehicles can be detected

The size of a partial occluded vehicle will be very different than that of a non-occluded vehicle and thus won't be detected.

[6] Region Of Interest

Multiple vehicles can be detected

When the multiple vehicles can be surrounded with a bounding box independently then they can be classified as vehicles. But when multiple vehicles will be surrounded with one bounding box they can't be detected.

Movement of the camera can be dealt with

No effect, we assume that the MVO extraction algorithm deals with motion.

1. Vehicles are detected in "vehicles-only" situations	+/-
2. Only vehicles are detected in complex situations	+/-
3. Changes in ambient lighting conditions can be handled	-
4. Partial occluded vehicles can be detected	-
5. Multiple vehicles can be detected	+/-
6. Movement of the camera can be dealt with	+

5.2 Symmetry detection

When facing the front side or back side of a vehicle it can be noticed that a vehicle is symmetric. So when these symmetries can be detected it is possible to classify a ROI as being a vehicle. In [18] a system is presented which uses symmetry to detect vehicles. The system is used in an experimental autonomous vehicle to detect other vehicles in front of it.

The symmetry of a ROI is determined by analyzing the symmetry of the gray levels, the horizontal edges and the vertical edges. Combining the symmetry maps of these three results in a final symmetry map showing the symmetry axis of the ROI.

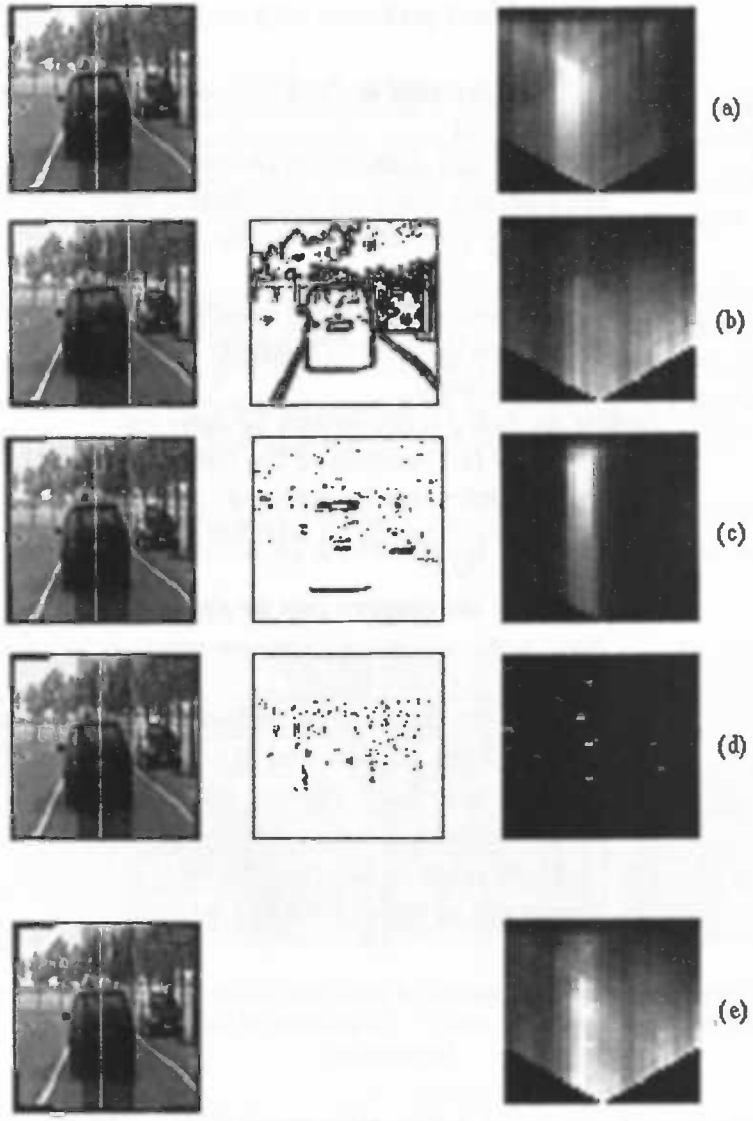


Figure 17. Computing the final symmetry; (a) grey-level symmetry, (b) edge symmetry, (c) horizontal edge symmetry, (d) vertical edge symmetry, (e) total symmetry. In every image on the left side the symmetry axis is superimposed.

The presented system finally tries to find a bounding box enclosing the vehicles by locating it upper and lower corners using the symmetry information. By combining all this information it is possible to determine whether this object is indeed a vehicle.

How this method performs with respect to the requirements can be seen below.

Vehicles are detected in "vehicles-only" situations

No problem.

Only vehicles are detected in complex situations

Because not many moving objects other than vehicles have these symmetry properties it will be possible to distinguish between vehicles and non-vehicles.

Changes in ambient lighting conditions can be handled

As long as the overall intensity of the vehicle stays the same this will be no problem.

Partial occluded vehicles can be detected

This will fail because of the lost symmetry characteristics.

Multiple vehicles can be detected

Again this depends on whether the multiple vehicles can be segmented individually.

Movement of the camera can be dealt with

No effect.

1. Vehicles are detected in "vehicles-only" situations	+
2. Only vehicles are detected in complex situations	+
3. Changes in ambient lighting conditions can be handled	+
4. Partial occluded vehicles can be detected	-
5. Multiple vehicles can be detected	+/-
6. Movement of the camera can be dealt with	+

5.3 Edges

A typical feature of a regular vehicle is that it comprises a large number of horizontal structures. This is particularly true when the vehicles are observed from the front or from the rear. While it is true that other objects contain also many horizontal structures, but not that many moving objects contain horizontal structures.

Applying a horizontal edge detector to the image results in a cluster of edges at the area where a vehicle might be present. By analyzing the cluster of edges with an appropriate algorithm it can de determined whether it really is a vehicle. Such an algorithm needs to distinguish between very strong horizontal edges, such as from a vehicle, and weak horizontal edges, such as from a person.

In situations where vehicles are viewed from the front or rear and no other objects with many horizontal edges are to expected this method can be very suitable.

(a) (b)

Figure 18. (a) A normal vehicle that could be detected by finding horizontal edges; (b) other object which would be detected as a vehicle when only horizontal edges are considered.

Again, I give an evaluation of the requirements:

Vehicles are detected in "vehicles-only" situations

Definitely possible.

Only vehicles are detected in complex situations

As can be seen in Figure 18, there are also other objects that have many horizontal edges. Detecting just vehicles isn't therefore achievable.

Changes in ambient lighting conditions can be handled

Changes in ambient lighting won't have effect on the gradient image. Edges are therefore still extractable.

Partial occluded vehicles can be detected

It depends which part of the vehicle is occluded, but overall the amount of horizontal edges won't be enough to reliable state that a MVO is a vehicle.

Multiple vehicles can be detected

This depends on whether the multiple vehicles are be segmented individually.

Movement of the camera can be dealt with

No effect.

1. Vehicles are detected in "vehicles-only" situations	+
2. Only vehicles are detected in complex situations	-
3. Changes in ambient lighting conditions can be handled	+
4. Partial occluded vehicles can be detected	-
5. Multiple vehicles can be detected	+/-
6. Movement of the camera can be dealt with	+

5.4 Neural networks

The authors of [19] used a neural network to decide if the detected moving object is indeed a vehicle. They use an image region of a fixed size as the input to the neural network to decide whether the image region contains a vehicle. Such a neural network needs of course be trained to be of any use. So we need a large set of training images to create a neural network that can be used in various situations. The neural network could thus be useful when we have enough training samples.

Because neural networks tend to be slow in training and classification, the authors of [19] developed a hardware-implemented system for vehicle detection. The input to the neural network is a grey level image of the area where the vehicles are to be detected. This image is normalized and divided into a fixed number of tiles which are then passed to the input neurons (see Figure 19).

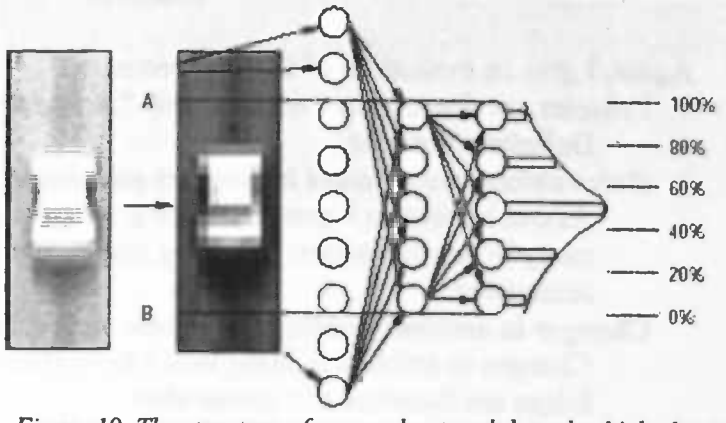


Figure 19. The structure of a neural network-based vehicle detector.

The authors researched several network architectures that were available on the used hardware device; Reduced Coulomb Energy (RCE), Probabilistic Reduced Coulomb Energy (PRCE) and Probabilistic Neural Networks (PNN).

The three architectures were compared to each other on several points; how image preprocessing (i.e. contrast normalization and tile size) affected the performance, how much training was needed and how good the networks detected vehicles in night images. Thru experiments done with these three architectures they found out that the PRCE network was the most suitable for vehicle detection.

For testing purposes they used a database of 10.000 images from six hours of videotape shot at different locations. One half consisted of daylight images and the other half consisted of night images. The vehicles varied in grey levels and size. The ambient lighting in the scenes was variable. For the results on the different contrast normalization and tile sizes I refer to the article itself. In Table 4 the performance of the PRCE and PNN network is shown. Where SR is the success rate, FAR the false alarm rate and x the number of prototypes stored during network training.

Table 4 Performance of the PRCE network and PNN for varying number of testing and training samples.

No. of Samples		PRCE Network			PNN	
Training	Testing	Proto(x)	SR(%)	FAR(%)	SR(%)	FAR(%)
50	1550	40	77	3.4	79	3.4
100	1500	72	84	1.0	84	1.0
200	1400	109	83	2.0	84	1.8
300	1300	183	90	1.2	91	1.5
400	1200	245	91	1.5	92	1.0
500	1100	92	93	1.3	91	2.6

The highest performance the researchers reached was with a PRCE network trained with 500 samples from the daylight samples in the database. Although the PNN was often slightly better than the PRCE network, the researchers noted that the PNN simply added a hidden unit for every training image which would lead to bad scalability.

To test the performance of the PRCE network during day as well as night situations a subset (5018 images) of the database was used. This set consisted of daylight images, nighttime images and transitions in-between. From this set 991 images were used for training. The resulting network performed a success rate of 92% and there were 39 false alarms. This shows that with this system a good detection rate can be achieved in similar situations as in the test setup.

How this technique performs in situations where other objects are to be expected is not mentioned but I think it can distinguish between vehicles and non-vehicles. The vehicles in the test setup were also going in the same direction (i.e. straight ahead) which questions the usability in situations where the vehicle direction isn't known beforehand.

Requirements evaluation:

Vehicles are detected in "vehicles-only" situations

According to the article this is no problem.

Only vehicles are detected in complex situations

Not mentioned in the article, but I think the neural network will be able to distinguish between vehicles and non-vehicles.

Changes in ambient lighting conditions can be handled

No problem according to the article.

Partial occluded vehicles can be detected

As long as the neural network isn't trained to detect partial occluded vehicles it will fail in doing so.

Multiple vehicles can be detected

The multiple vehicles must be individually analyzed by the neural network. So the MVO extraction algorithm must be able to segment the vehicle independently.

Movement of the camera can be dealt with

No effect.

1. Vehicles are detected in "vehicles-only" situations	+
2. Only vehicles are detected in complex situations	+
3. Changes in ambient lighting conditions can be handled	+
4. Partial occluded vehicles can be detected	-
5. Multiple vehicles can be detected	+/-
6. Movement of the camera can be dealt with	+

Chapter 6 Vehicle detection in still images

Techniques mentioned in the previous chapter are not robust at all when they need to detect vehicles and nothing else but vehicles. They rather detect objects which could be vehicles. We need more advanced techniques to determine if a segmented object is indeed a vehicle. The techniques mentioned in the previous chapter can be used as a preprocessing stage to mark a ROI. In subsequent stages the ROI can be analyzed to determine if it contains indeed a vehicle. If no preprocessing based on image sequences was done to find a ROI, we must analyze the complete image to find a vehicle.

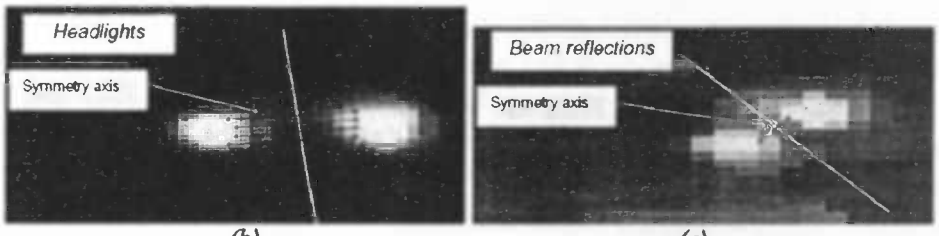
In this chapter I present several techniques that determine whether a vehicle is present in the image.

6.1 Head- and taillight detection

When we look at an image of vehicle we can notice some very distinctive features of the vehicle. On the front- and backside of a vehicle we can clearly see the head- and the taillights correspondingly. The lights are particularly distinctive when they are lit. In [20] the author detects pairs of headlights taking into account the symmetry between two headlights and their luminance. Two lights form a head- or taillight pair when they are roughly the same size and they are symmetric with respect to an axis pointing in the traffic direction.

By finding only these light pairs it is possible to distinguish between lights from a vehicle and other light sources such as reflections.

(a)



(b) (c)

Figure 20. (a) A typical motorway at night; (b) a pair of headlights and their symmetry relative to the traffic direction; (c) a pair of lights which show symmetry correspondences, but are clearly not a pair of headlights according to their distance and their symmetry axis.

Clearly this method is only useable when the lights of the vehicle are lit which isn't true most of the time. To be successful in distinguishing between lights from vehicles and other sources it must be known what the direction of the motion of the vehicle is, otherwise it is impossible to determine whether the symmetry axis is in line with the vehicle direction. Also, when there are many other lights in the scene, such as streetlights or many other vehicles it will be difficult to detect the single vehicles. Overall this method won't be practicable in normal situations.

The evaluation of the requirements of this vehicle detection method in still images:

Vehicles are detected in "vehicles-only" situations

As long as it the vehicles have their light lit and it is dark, this will be no problem.

Only vehicles are detected in complex situations

Because no other object has headlights like that of a vehicle it is possible to detect just vehicles.

Changes in ambient lighting conditions can be handled

This method will only work at night.

Partial occluded vehicles can be detected

This method depends on the symmetry between two headlights, so when only one headlight is visible the symmetry is lost.

Multiple vehicles can be detected

Will be possible.

Movement of the camera can be dealt with

No effect.

1. Vehicles are detected in "vehicles-only" situations	+
2. Only vehicles are detected in complex situations	+
3. Changes in ambient lighting conditions can be handled	-
4. Partial occluded vehicles can be detected	-
5. Multiple vehicles can be detected	+/-
6. Movement of the camera can be dealt with	+

6.2 3D Model based detection

Most of the previously mentioned approaches for detecting vehicles rely mostly on some sort of low-level reasoning about the possibility that an image feature or the grouping of different features extracted from the image represent a vehicle. This approach can be very effective in constrained scenes. On a motorway, for example, it is very unlikely that a rigid moving object approximately the size of a vehicle is in fact something completely different. To detect a vehicle in such a scene it is often sufficient to extract moving objects from the image by means of background subtraction. The extracted object will be most likely a vehicle.

But in an outdoor scene with many moving objects that don't belong to the class of vehicles it is obvious that many misclassifications will be made. Trying, for example, to find a region with relative many horizontal edges, which are characteristic for vehicles, is certainly useful in a region where only vehicles are to be expected such as on the motorway mentioned before. In a more complex scene this leads to many misclassifications. There are of course not much objects that are in motion and also have lots of horizontal edges, but it is possible that such objects exist.

Much research has been done in computer vision with respect to determining the three-dimensional location of objects from a single two-dimensional image. This is done by matching three-dimensional models (e.g. polygonal, polyhedral or points in 3D) of the objects to be located with the image containing the object. Various methods have been proposed by different authors ([21], [22], [1]) which all do basically the same. They determine the rotation and translation of the object with respect to the camera. Some of the research done completely focuses on determining the position and orientation of vehicles in outdoor images. All these methods have been proved to be successful in determining the pose of an object (or vehicle). Some methods work better in complex scenes than others while some may be faster than others.

The difference with the previously mentioned methods is that the model-based approach is top-down whereas the other methods are bottom-up. With the latter methods it is assumed there is an object in the scene and then the assumption is evaluated whereas in the former bottom-up approach image features are first collected resulting in a decision step in which it is determined whether those features belong to an object of interest.

Pose recovery can be used to identify whether an image contains a vehicle. When the algorithm calculates the pose of a vehicle model in the image and the resulting error is small enough it can be said with great certainty that the object is indeed a vehicle.

There are basically two ways to determine the pose of an object by using 3D models. The first is to find point-to-point correspondences between the model and the image and the second is to find line-to-line correspondences.

The first step in most algorithms is to start with in initial assumption of the correspondences between the points or lines and the pose of the object. This assumption is then evaluated and when necessary this assumption is altered iteratively until the correct pose is recovered or if a fixed number of iterations is reached. A final score says something about the quality of the recovered pose resulting in a decision made about the presence of an object and possibly the location of an object of interest.

Pose recovery – Lowe's method

Lowe [21] was one of the first who presented a simple and elegant method to compute the optimal 3D-transformation and rotation of an object model to align points from a 3D wire-frame model with corresponding points of an object in an image. He applies Newton[7]'s method to solve for the rotation and translation of the model to align the model with an object in the image.

Lowe starts with a simple equation to project a point in a three-dimensional model point **p** into a two-dimensional image point *(u, v)*. This is a simple pin-hole camera.

$$(x, y, z) = R(\mathbf{p} - \mathbf{t})$$

$$(u, v) = \left(\frac{fx}{z}, \frac{fy}{z} \right)$$

Where R is a rotation matrix and **t** is a translation vector which transforms a point **p** in model coordinates into a point in the camera-centered coordinate system. To project this new *(x,y,z)* point into an image point *(u,v)* the perspective projection is performed with a focal length f proportional to the real camera focal length.

The problem is now to solve for **t**, R and possibly f given a set of model points and their corresponding locations in the image. To apply Newton's method to this problem it is necessary to calculate the partial derivatives of u and v with respect to the unknown parameters. Because this is rather complicated for the previous projection equation Lowe has proposed another equation for projecting point **p** into image coordinates:

$$(x, y, z) = R\mathbf{p}$$

$$(u, v) = \left(\frac{fx}{z + D_z} + D_x, \frac{fy}{z + D_z} + D_y \right)$$

The parameters R and f stay the same whereas **t** has been replaced by the parameters D_x, D_y and D_z. Here D_x and D_y specify the location of the object on the image plane and D_z specifies the distance from the object to the camera.

---

[7] A root-finding algorithm which uses the first few terms of the Taylor series of a function $f(x)$ in the vicinity of a suspected root to zero in on the root.

This new parameterization facilitates the computation of the partial derivatives. The partial derivatives of x, y and z with respect to the rotation angles ϕ_x, ϕ_y and ϕ_z (rotation angles about the coordinate axes in radians) are given in the following table:

	x	y	z
ϕ_x	0	$-z$	y
ϕ_y	z	0	$-x$
ϕ_z	$-y$	x	0

Lowe states that it is now possible to calculate the partial derivatives of u and v with respect to all camera parameters. The following table shows these derivatives:

	u	v
D_x	1	0
D_y	0	1
D_z	$-fc^2x$	$-fc^2y$
ϕ_x	$-fc^2xy$	$-fc(z+cy^2)$
ϕ_y	$-fc(z+cx^2)$	fc^2xy
ϕ_z	$-fcy$	fcx
f	cx	cy

Where $c = \dfrac{1}{z + D_z}$.

In all consecutive iterations of Lowe's proposed algorithm the system needs to be solved for a vector of corrections:

$$\mathbf{h} = [\Delta D_x, \Delta D_y, \Delta D_z, \Delta\phi_x, \Delta\phi_y, \Delta\phi_z]$$

When also the focal length f is unknown then Δf will also be added to $\mathbf{h}$.

Lowe's algorithm now dictates that a model point is projected into the image using the current parameter estimates and then measuring the error with respect to the image point. The u and v components can used independently to create separate linear constraints. The errors E_u and E_v for the u and v components is expressed by the following equations:

$$E_u = \frac{\partial u}{\partial D_x}\Delta D_x + \frac{\partial u}{\partial D_y}\Delta D_y + \frac{\partial u}{\partial D_z}\Delta D_z + \frac{\partial u}{\partial \phi_x}\Delta\phi_x + \frac{\partial u}{\partial \phi_y}\Delta\phi_y + \frac{\partial u}{\partial \phi_z}\Delta\phi_z$$

$$E_v = \frac{\partial v}{\partial D_x}\Delta D_x + \frac{\partial v}{\partial D_y}\Delta D_y + \frac{\partial v}{\partial D_z}\Delta D_z + \frac{\partial v}{\partial \phi_x}\Delta\phi_x + \frac{\partial v}{\partial \phi_y}\Delta\phi_y + \frac{\partial v}{\partial \phi_z}\Delta\phi_z$$

So for every point-to-point correspondence two equations need to be solved.

With three correspondences we can derive six equations and produce a complete linear system which can be solved for all six parameters.
More compactly the above equations can be expressed as

$$\mathbf{J} \cdot \mathbf{h} = \mathbf{e}$$

where **J** is the Jacobian matrix containing the partial derivatives of u and v with respect to the six unknowns and **e** the vector of errors measured. Newton's method can then be used to minimize the error function.

A great improvement on this method, as proposed by Lowe himself, is to use line-to-line correspondences instead of point-to-point correspondences. Finding the transverse locations of lines is relatively easy whereas it is often very difficult to determine the exact location where a line starts and ends, also because lines can be partially occluded. By finding the equations of the detected lines in the image it is possible to determine the perpendicular distance between a point on a model line and a line in the image. This distance is used as an error measure that needs to be minimized. The perpendicular distance from the origin to a line with slope m is expressed as:

$$d = \frac{-m}{\sqrt{m^2+1}}u + \frac{1}{\sqrt{m^2+1}}v$$

When u and v are substituted with some point (u',v') we can calculate the perpendicular distance of this point to the origin. The perpendicular distance of this point to the line is then $d - d'$. This distance can then be used as the error measure.

Because we now have another error function we also need to find the derivatives of d with respect to the six unknowns. But because d is just a linear combination of u and v the derivatives of d are also just a linear combination of the partial derivatives of u and v. The following table shows the partial derivatives of d:

	d
D_x	A
D_y	B
D_z	$-Afc^2x - Bfc^2y$
ϕ_x	$-Afc^2xy - Bfc(z+cy^2)$
ϕ_y	$-Afc(z+cx^2) + Bfc^2xy$
ϕ_z	$-Afcy + Bfcx$
f	$Acx + Bcy$

Where $A = \frac{-m}{\sqrt{m^2+1}}$ and $B = \frac{1}{\sqrt{m^2+1}}$.

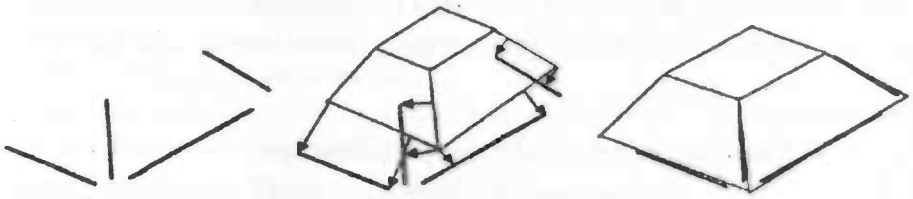
For each line-to-line correspondence there are two equations; one for each perpendicular distance between the endpoints of the model-line and the image-line.

The errors E_{p1} and E_{p2} for the perpendicular distances between the two endpoint *p1* and *p2* of the model edges to and image line is expressed by the following equations:

$$E_{p1} = \frac{\partial d}{\partial D_x}\Delta D_x + \frac{\partial d}{\partial D_y}\Delta D_y + \frac{\partial d}{\partial D_z}\Delta D_z + \frac{\partial d}{\partial \phi_x}\Delta \phi_x + \frac{\partial d}{\partial \phi_y}\Delta \phi_y + \frac{\partial d}{\partial \phi_z}\Delta \phi_z$$

$$E_{p2} = \frac{\partial d}{\partial D_x}\Delta D_x + \frac{\partial d}{\partial D_y}\Delta D_y + \frac{\partial d}{\partial D_z}\Delta D_z + \frac{\partial d}{\partial \phi_x}\Delta \phi_x + \frac{\partial d}{\partial \phi_y}\Delta \phi_y + \frac{\partial d}{\partial \phi_z}\Delta \phi_z$$

So there is the same amount of information as is in the point-to-point correspondences method. With three line-to-line correspondences the linear system of six equations and six variables can be solved.



(a) (b) (c)

Figure 21. Pose recovery from line correspondences; (a) extracted image lines, (b) initial guess of model pose with perpendicular distances between lines shown, (c) final recovered pose.

Pose recovery – Fully projective formulation for Lowe's method

Araújo *et al* (in [23]) noted that Lowe's method isn't quite correct. In Lowe's method it is assumed that D_x, D_y and D_z are constants that need to be found by an iterative algorithm. This is a false assumption because D_x, D_y and D_z depend on the location of the points (or lines). According to Araúja *et al* the correct full projective formulation would be:

$$[x', y', z']^T = \mathbf{R} \cdot \mathbf{p}$$

$$[d_x', d_y', d_z'] = -[\mathbf{r}_x \cdot \mathbf{t}, \mathbf{r}_y \cdot \mathbf{t}, \mathbf{r}_z \cdot \mathbf{t},]$$

$$[u, v] = f\left[\frac{x' + d_x'}{z' + d_z'}, \frac{y' + d_y'}{z' + d_z'}\right]$$

The partial derivatives of *u* and *v* are shown in the following table:

	u	v
d_x'	fc	0
d_y'	0	fc
d_z'	$-fac^2$	$-fbc^2$
ϕ_x	$-fac^2y'$	$-fc(z'+bcy')$
ϕ_y	$fc(z'+acx')$	fbc^2x'
ϕ_z	$-fcy'$	fcx'
f	ac	bc

Where $[a,b,c]=\left[x'+d_x', y'+d_y', \frac{1}{z'+d_z'}\right]$.

Similar to Lowe's original method, this new formulation can also be applied to line-to-line correspondences just as how Lowe described it.

Problem: Initial correspondences

For all pose recovery algorithms based on line-to-line (or point-to-point) correspondences it is obvious that it must be known which line in the image corresponds to which line of the model. Of course one can try to calculate the pose of the object using all the possible line-to-line correspondences. This method has a very high impact on the computational load and it is practically impossible to compute in reasonable time.

Thus for pose recovery to be useful it is necessary that some preceding algorithm chooses the most likely line-to-line correspondences to use in the pose calculation. This can be done by first extracting the vehicle from a sequence of images as is done in [24] or by first searching particular perceptual groupings of lines as Lowe did in [25].

Even with such an algorithm it is most likely that still many possible correspondences remain, but the pose can probably be calculated in reasonable time.

Using pose recovery is, in my opinion, not practical for mere vehicle detection. It is too complex to find the proper correspondences in still images. A more suitable use for pose recovery is found in applications that need the exact location of a vehicle in a three-dimensional world. The speed of a vehicle can for example be determined if its position at two different moments is known. Most applications found are using pose recovery for tracking vehicles rather than for the detection.

Requirements evaluation:

Vehicles are detected in "vehicles-only" situations

When a good implementation of a 3D pose recovery algorithm is available it will be possible to detect vehicles of which a wire frame model is available.

Only vehicles are detected in complex situations

A distinction can be made between objects of which a 3D model is available and of which no model is available.

Changes in ambient lighting conditions can be handled

Changes in ambient lighting conditions won't have an effect on the gradient image from which the lines are extracted.

Partial occluded vehicles can be detected

Partial occluded vehicles lack many lines that are needed to recover the pose, so this will be difficult. But this highly depends on the amount of occlusion.

Multiple vehicles can be detected

Multiple vehicles result in more lines in the image, so the problem of find the correct correspondences will be very complex.

Movement of the camera can be dealt with

No effect.

1. Vehicles are detected in "vehicles-only" situations	+
2. Only vehicles are detected in complex situations	+
3. Changes in ambient lighting conditions can be handled	+
4. Partial occluded vehicles can be detected	+/-
5. Multiple vehicles can be detected	+/-
6. Movement of the camera can be dealt with	+

6.3 Template matching

Template matching is a well-known approach that can detect and locate known objects in images. The basic idea behind template matching is that a template of the object that needs to be detected is known in advance, this template is then used to locate the object in a target image. There are more or less two types of template matching; area-based matching and feature-based matching.

Area-based matching uses small image patches composed of gray values as templates. The matching stage tries to find a match between the small image patch and the target image solely based on the values of the pixels. The matching criterion used is the difference between the pixel values in the template and those in the target image. Matching based on finding differences is very sensitive to changes in lighting and to be useful we need templates that are very similar to the appearance of the object in the image.

The other type of template matching is feature-based matching; this type uses higher-level features derived from the image. These features can for example be the edges of objects. Matching is now not based on calculating the difference between the gray values in the template image and the target image, but on calculating the differences between the features. This type of template matching is considered more effective than area-based template matching. In this section I will describe several template matching strategies, area-based as well as feature-based strategies.

Area-based matching

In [26] the authors used the most simple form of template matching; they match gray value images to the target image. The purpose of their research was to classify the vehicles into three different types, i.e. sedan, van or pickup. The matching process is preceded by a segmentation step based on background subtraction. The matching score is calculated as:

$$S = 1 - \frac{\sum_{x=0}^{X-1}\sum_{y=0}^{Y-1}|I(x,y) - T(x,y)|}{L \cdot X \cdot Y}$$

Where L is the maximum gray level, X and Y are the width and height of the template image respectively, I is the target image and T is the template image. The score S varies between 0 and 1 where 1 indicates a perfect match.

As in all template matching methods there needs to be a set of templates to match against the target image. These templates need to cover all the vehicles that are to be expected. In experiments in [26] were done with a set of twelve template images covering three types of vehicles, i.e. sedan, van and pickup. The vehicles were always shown from the side, so all templates are also side view images of vehicles. The similarity between each template and two unknown vehicles is show in Figure 22.

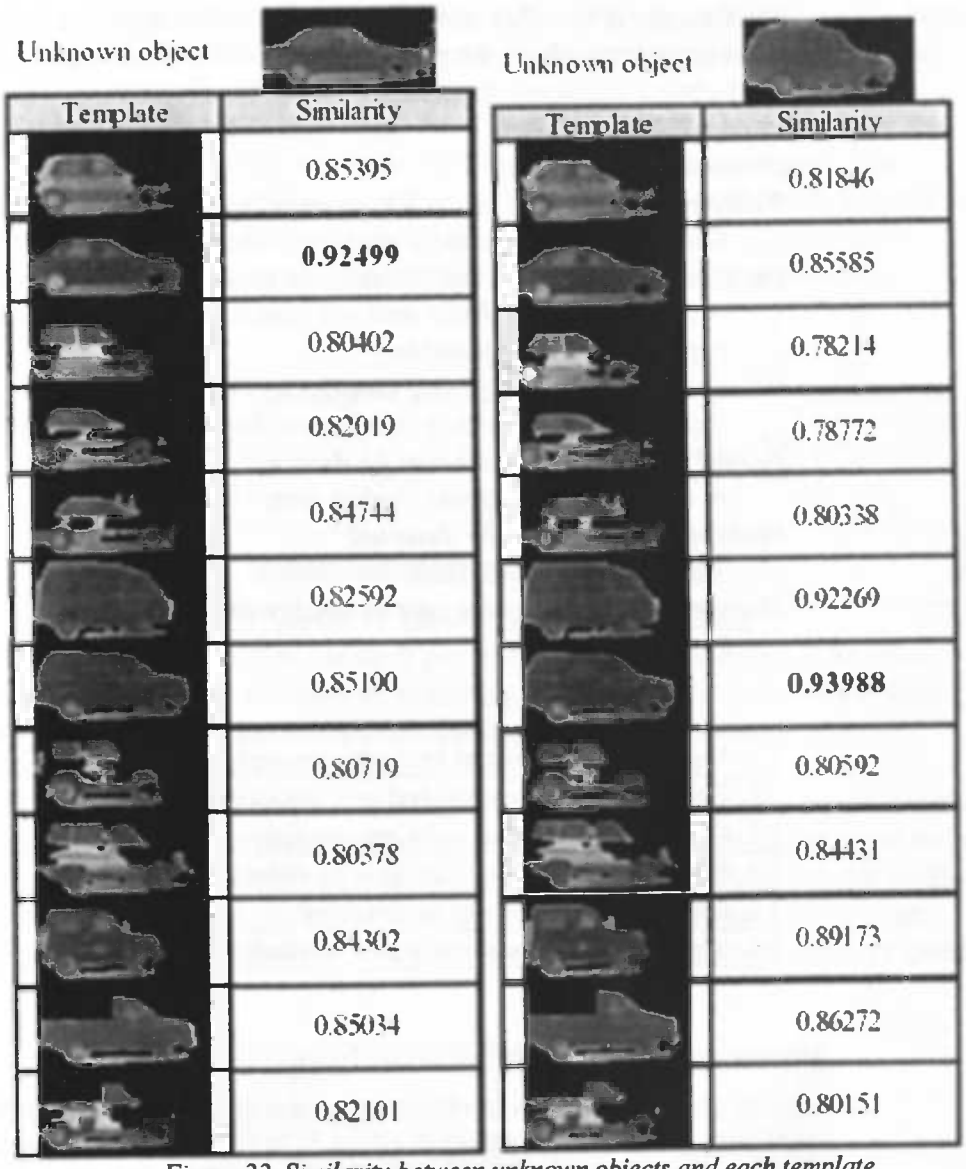


Figure 22. Similarity between unknown objects and each template.

The conclusion made was that this method was very good in classifying the different types of vehicles in the test situation. How this method works in situations where not only vehicles are present but also other objects such as people is not researched. But it can be expected that the similarity between a person and the templates is not very high.

There are a couple of requirements that need to be met for this method to be successful.

- Vehicles need always be viewed from the same viewpoint
- The size of the vehicles in the image need all be almost the same
- The intensity of the pixels in the template image and those in the target image need to be similar.

My thought is that this method is too limited to apply in unconstrained outdoor situations where the pose of the vehicles is not known in advance and where lighting changes are great.

Requirements evaluation:

Vehicles are detected in "vehicles-only" situations

Only if there are enough templates to cover all possible vehicles.

Only vehicles are detected in complex situations

A template of a vehicle will not match an image of, for example, a person. So this is possible.

Changes in ambient lighting conditions can be handled

According to the article this is possible.

Partial occluded vehicles can be detected

A vehicle that is partial visible won't match with a template.

Multiple vehicles can be detected

This won't be a problem.

Movement of the camera can be dealt with

No effect.

1. Vehicles are detected in "vehicles-only" situations	+
2. Only vehicles are detected in complex situations	+/-
3. Changes in ambient lighting conditions can be handled	+/-
4. Partial occluded vehicles can be detected	-
5. Multiple vehicles can be detected	+
6. Movement of the camera can be dealt with	+

Feature-based template matching

The greatest problem in object recognition and particularly in template matching is the uncertainty in the data. It is impossible to have an exact template for every possible vehicle that needs to be detected. So we need a method that is able to cope with these uncertainties. It needs to be able to detect vehicles of which no exact template available is. One option to deal with the uncertainties in pixel intensities is to use higher-level features extracted form the target image. A template in this case is a binary image where a pixel is unequal to zero when at that location a feature is to be expected and zero otherwise.

The authors of [27] used a template matching approach to track vehicles from another vehicle. In their application they used the salient edges of a vehicle as the features to be matched. They used the shape of the sides of a vehicle as the template. The matching technique they applied is called Chamfer matching; the basis of this algorithm is matching shapes (i.e. edges) using distance maps. A distance map (or distance transform) is a map where at

each location the distance to the nearest edge in the target image is calculated. The templates that are matched to the target image are binary (black/white) images covering the shape of the object.

To find the object in the image the algorithm shifts the template over the distance transform (DT) and sums the distances at those locations in the template where the value is 1, i.e. the correlation between the template and the DT is calculated. When the correlation at a certain position is less than a predefined threshold it can be concluded that a vehicle is present at that location.

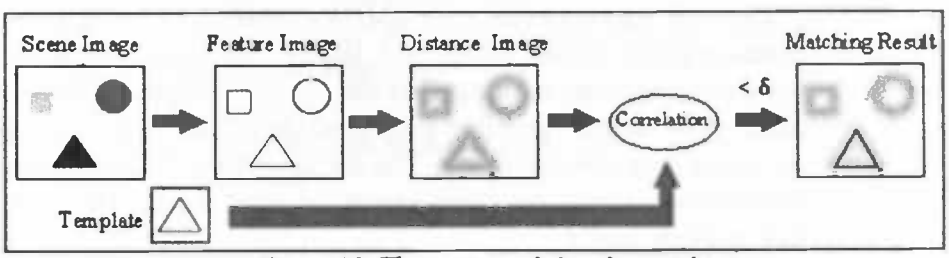
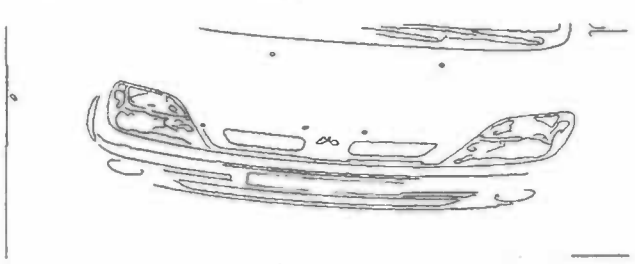


Figure 23. The process of chamfer matching.

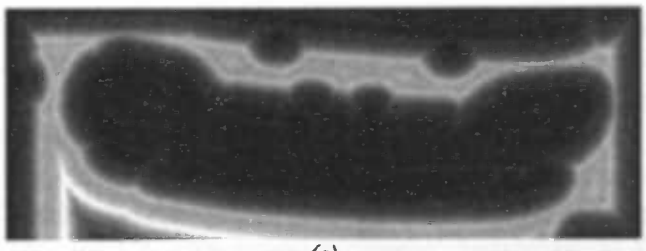
This method seemed to me very promising to detect vehicles. It handles imperfect data and it is based on matching shapes of objects. So when there are appropriate templates available that cover the shape of most types of vehicles it should perform very well.

The first step in Chamfer matching is finding the edges in the target image. An edge detector that is able to extract all edges and no non-edges is yet to be discovered. Nevertheless many algorithms exist that come close to the perfect edge detector. One of the algorithms that could be used is the Canny edge detector [28]. This algorithm extracts the edges in the target image very good.

(a)



(b)



(c)

Figure 24. (a) The target image; (b) edges extracted with a Canny edge detector; (c) distance image (blue=short distance, red=long distance).

The next step is to calculate at every position in the feature image the distance to the nearest edge pixel. In [27] the authors noted that the computational load to determine the exact Euclidean distance to the edge pixels is too high. Therefore they used the Chamfer distance transform, hence the name Chamfer matching. The computation of the Chamfer distance involves two kernel operations over the edge image. This can be done very fast and efficiently.

The next step is the actual matching; at every possible location in the DT and for every available template the correlation is calculated. According to the score at a location it can be said whether a vehicle is present.

In [27] the authors didn't use templates that cover the complete shape of a vehicle, instead they used only the shape of the sides of a vehicle. For the application the authors used Chamfer matching these simple templates proved to be good enough to track vehicles in front of another vehicle. Also the vehicle was already segmented from the rest of the image. From begin to end the process can be visualized as in Figure 25.

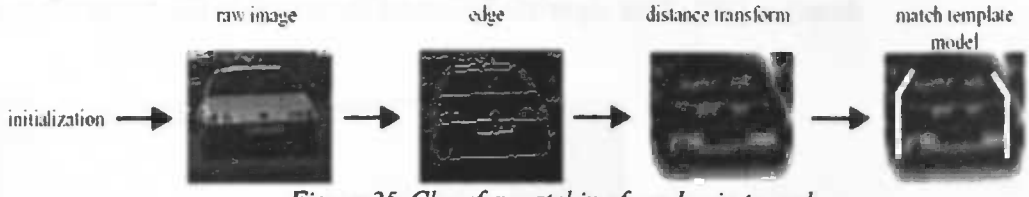


Figure 25. Chamfer matching from begin to end.

In Chapter 8 I will show the results of my own experiments I did on Chamfer matching. Making representative templates is one of the biggest problems I encountered in Chamfer matching.

How this method performs with respect to the requirements is shown next.

Vehicles are detected in "vehicles-only" situations

This depends highly on the used templates. But when a representative set of templates is available then detecting vehicles will be no problem.

Only vehicles are detected in complex situations

The templates will match only on true vehicles.

Changes in ambient lighting conditions can be handled

The ability to extract edges from an image isn't effected by changes in lighting.

Partial occluded vehicles can be detected

Partial occluded vehicles won't be matched by any template.

Multiple vehicles can be detected

This will be no problem.

Movement of the camera can be dealt with

No effect.

1. Vehicles are detected in "vehicles-only" situations	+
2. Only vehicles are detected in complex situations	+
3. Changes in ambient lighting conditions can be handled	+
4. Partial occluded vehicles can be detected	-
5. Multiple vehicles can be detected	+
6. Movement of the camera can be dealt with	+

Chapter 7 Conclusion

It seems that there is no single method that meets up to all the requirements. Most systems that were considered performing very well according to an article often make certain assumptions about the situation where the vehicles need to be detected. Some of these assumptions are:

- Assume that the direction of all the vehicles is the same, i.e. like on a highway.
- Take the angle at which the vehicles are viewed as a constant value.
- Assume no non-vehicles are to be expected.
- Look and at daylight images.

By making these assumptions it is possible to develop a system that detects vehicles with a good performance. Without making any assumption it is very difficult to develop a visual based vehicle detection system.

Also, by combining several methods into one system it is possible to develop a fairly robust vehicle detection system. Which combination to make depends completely on the situation where the vehicles need to be detected.

Chapter 8 Experiments

In this chapter I will present a few results from experiments I did. I've done more experiments to test the several methods described in this thesis. So the experiments described here are just a subset.

8.1 Background Estimation

In Figure 26 a few images from a longer sequence of images is shown. Of this sequence the average is taken to estimate the background. The estimation of the background at each frame of Figure 26 is shown in Figure 27.

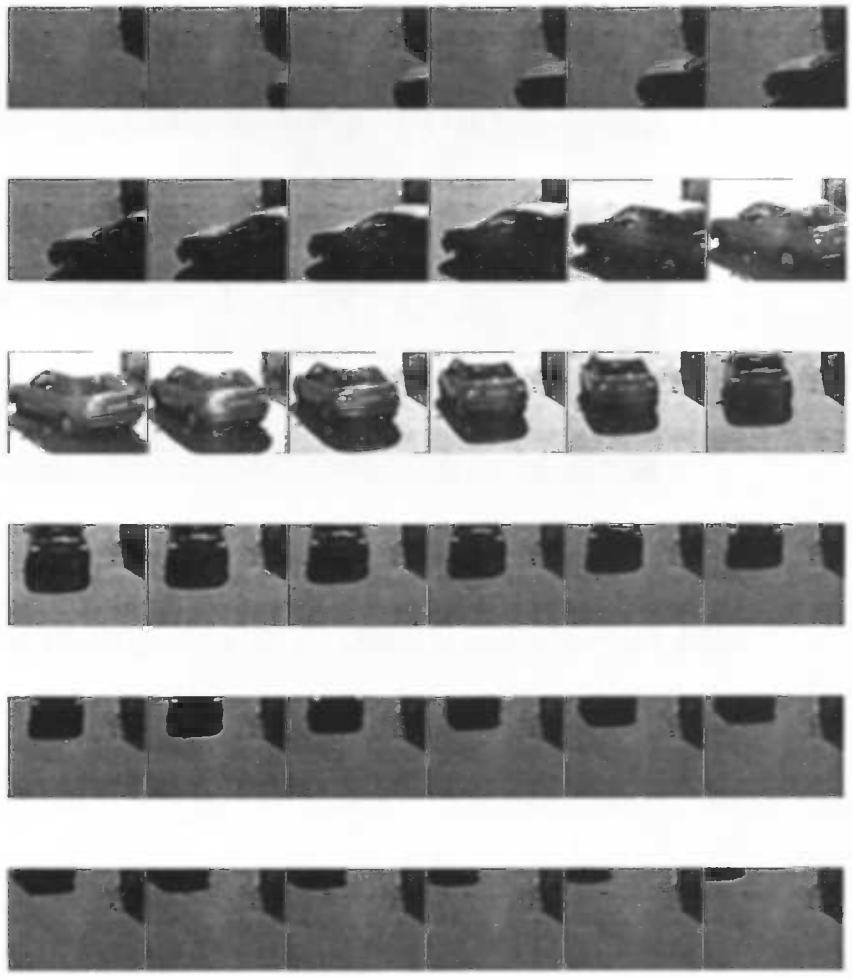


Figure 26. Sequence of video images.

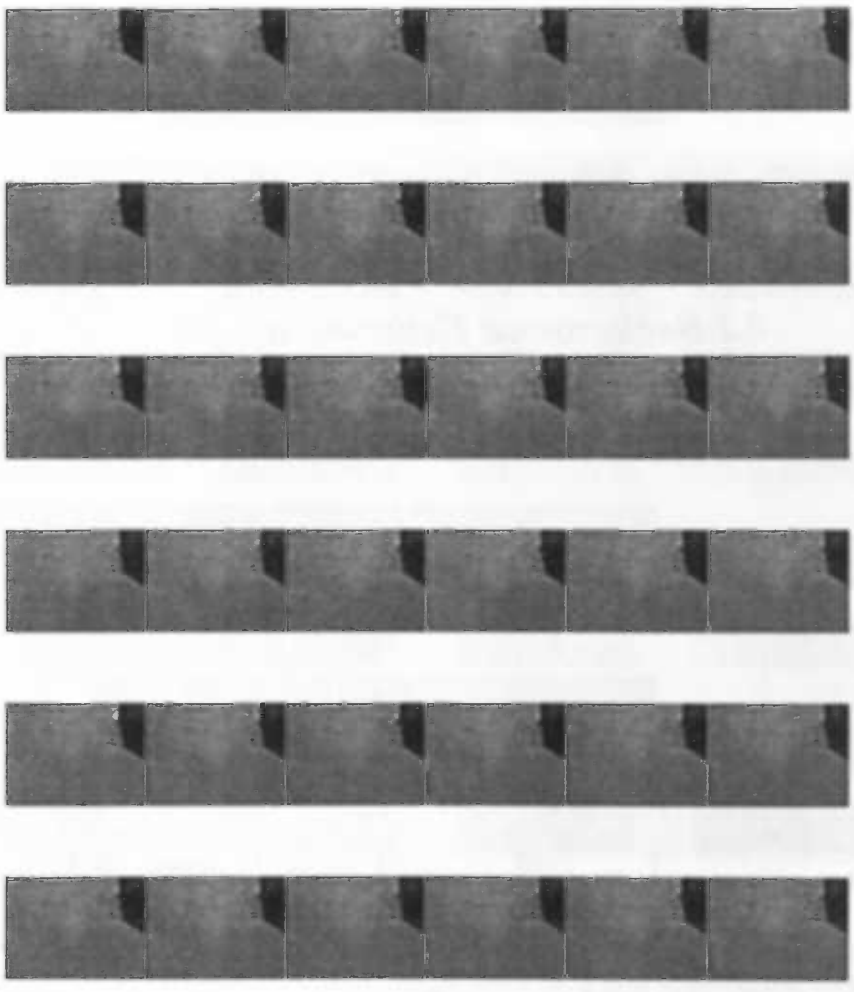


Figure 27. Background estimation at each frame.

Taking a simple averaging technique performs quite good. But the speed at which the background is updated is chosen empirically for this situation.

8.2 3D pose recovery

In Matlab I implemented several functions to test the 3D pose recovery algorithm. I used the fully projective formulation of Araújo to find the minimal perpendicular distance between 3D model-lines and image-lines that were extracted with Burns line extraction algorithm. Because searching for the best line-to-line correspondences was to complex I manually chose the line-to-line correspondences. The only thing that was to be done was to find the minimal distance. The image used can be seen in Figure 28. The lines that are used to recover the 3D pose are superimposed on this image. These lines are manually selected subset of all the lines found with Burns line extraction algorithm. The 3D model used is shown in Figure 29, this model is a general vehicle model as described by Koller in [42].

Figure 28. Image of a car with lines superimposed on them.

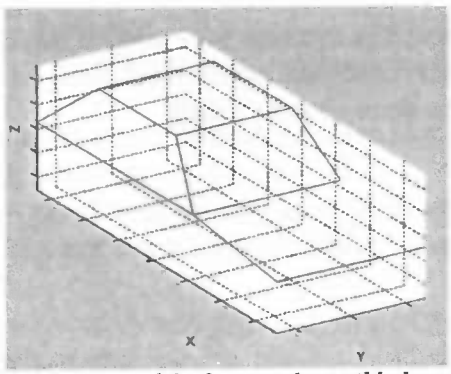


Figure 29. 3D wire frame model of a car; the visible lines are manually selected.

When applying the iterative pose recovery algorithm I get the result shown in Figure 30. Each sub-image shows the pose of the model at an iteration. Top-left is the first iteration and from left to right are the next iterations. As can be seen the correct pose of the model is recovered.

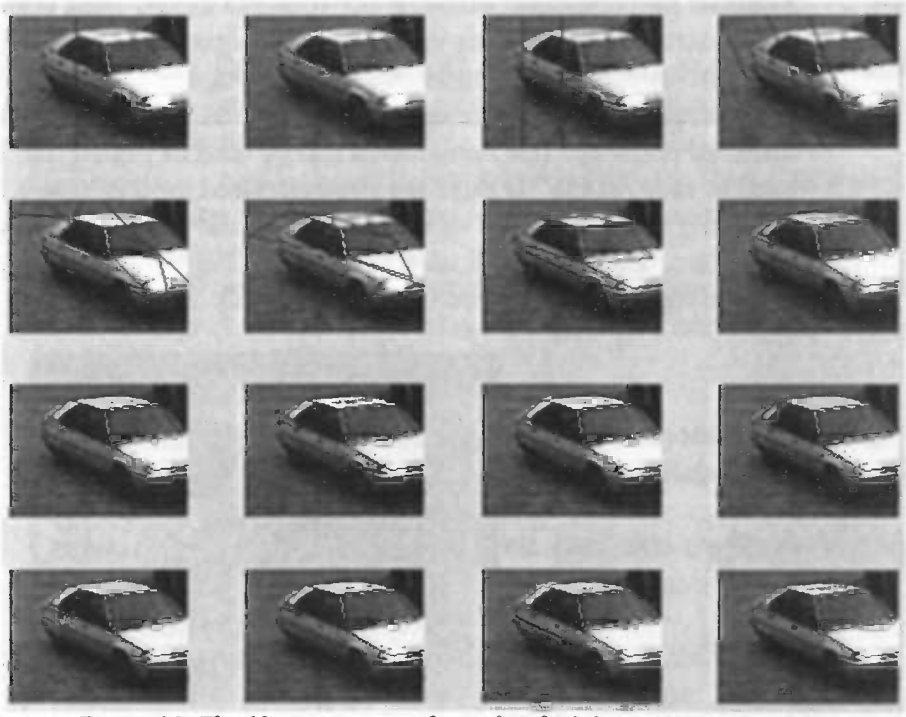


Figure 30. The 12 iterations performed to find the correct pose of the model.

Of course this test was very simplified. It was know in advance which model-lines corresponded to which image-lines. In a real application this needs to be done automatically.

8.3 Chamfer matching

I implemented a Chamfer matching algorithm in C (under Linux) for testing its ability to detect vehicles in images. For a test set of vehicles shown from the front as in Figure 31 I used the templates in Figure 32, these templates are only used as they are shown, so they are not rotated or scaled. These are representations of the front bumper of a vehicle.

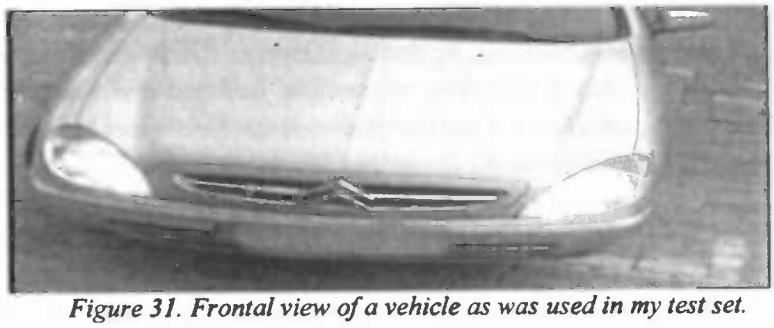


Figure 31. Frontal view of a vehicle as was used in my test set.

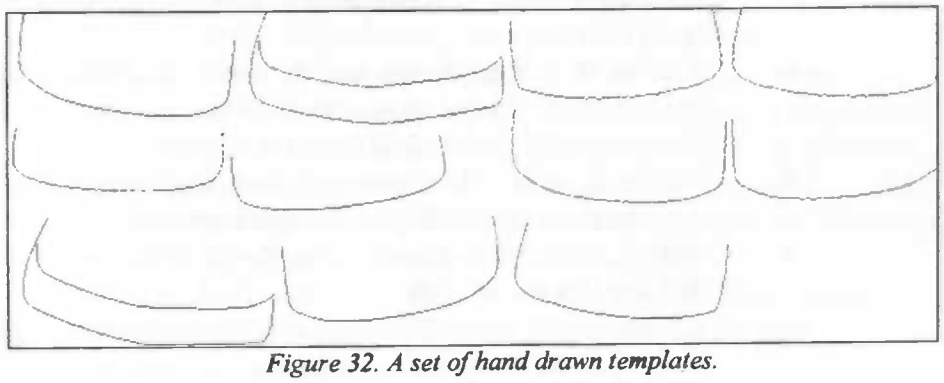


Figure 32. A set of hand drawn templates.

In a test set of 82 images which contain vehicles like in Figure 31 and no vehicles at all the test results are:

- In 54 images no vehicle was detected, in 47 of these the bumper of the vehicle was not completely visible or there was no vehicle present.
- In 28 images a vehicle was detected, in 2 of these the bumper was not completely visible.

So with a limited amount of templates a good performance can be achieved.

Chapter 9 References

1. Koller, D., K. Daniilidis, and H.H. Nagel, *Model-based object tracking in monocular image sequences of road traffic scenes.* International Journal of computer vision, 1993. **10**(3): p. 257-281.
2. Di Stefano, L. and E. Viarani. *Vehicle detection and tracking using the block matching algorithm.* in *3rd IEEE Conference on Circuits, Systems, Communications and Computer*. 1999.
3. Sullivan, G.D., et al., *Model-based vehicle detection and classification using orthographic approximations.* Image and Vision Computing, 1997. **15**: p. 649-654.
4. Dacolian, *Dacolian BV.*
5. Coifman, B., et al., *A real-time computer vision system for vehicle tracking and traffic surveillance.* Transportation Research, 1998. **C6**: p. 271-288.
6. Ferrier, N.J., S.M. Rowe, and A. Blake. *Real-time traffic monitoring.* in *2nd IEEE Workshop on Applications of Computer Vision*. 1994.
7. Traficon, *VIP/I - Incident Monitor.*
8. Gupte, S., et al., *Detection and classification of vehicles.* IEEE Transactions on Intelligent Transportation Systems, 2002. **3**(1).
9. Eikvil, L. and R.B. Huseby, *Traffic surveillance in real-time using hidden Markov models.*
10. Cucchiara, R., et al. *Real-time detection of moving vehicles.* in *10th International Conference on Image Analysis and Processing*. 1999. Venice.
11. Fathy, M. and M.Y. Siyal, *An image detection technique based on morphological edge detection and background differencing for real-time traffic analysis.* Pattern Recognition Letters, 1995. **16**: p. 1321-1330.
12. Hoose, N., *IMPACT: an image analysis tool for motorway analysis and surveillance.* Traffic Engineering Control Journal, 1992: p. 140-147.
13. Cucchiara, R., et al. *Detecting objects, shadows and ghosts in video streams by exploiting color and motion information.* in *11th IEEE Conference on Image Analysis and Processing*. 2001.
14. Karmann, K. and A. von Brandt, *Moving object recognition using an adaptive background memory.* Time-varying image processing and moving object recognition, 1990. **2**: p. 297-281.
15. Ridder, C., O. Munkelt, and H. Kirchner. *Adaptive background estimation and foreground detection using Kalman-filtering.* in *International Conference of Recent Advances in Mechatronics*. 1995.

16. Kilger, M. *A shadow handler in video-based real-time traffic monitoring system.* in *WACV'92*. 1992.
17. Lipton, A., *Local application of optic flow to analyse rigid versus non-rigid motion.* ICCV Workshop on Frame-Rate Vision, 1999.
18. Bertozzi, M., et al. *Stereo vision-based vehicle detection.* in *IEEE Intelligent Vehicles Symposium*. 2000.
19. Mantri, S., D. Bullock, and J. Garrett, *Vehicle detection using a hardware-implemented neural net.* IEEE Expert, 1997. **12**(1): p. 15-21.
20. Cucchiara, R. and M. Piccardi. *Vehicle detection under day and night illumination.* in *3rd International ICSC Symposium on Intelligent Industrial Automation*. 1999.
21. Lowe, D.G., *Fitting parameterized three-dimensional models to images.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 1991. **5**(13): p. 441-450.
22. Sullivan, G.D., A.D. Worrall, and J.M. Ferryman, *Visual object recognition using deformable models of vehicles.* IEEE Press, 1995.
23. Araújo, H., R.L. Carceroni, and C.M. Brown, *A fully projective formulation to Lowe's tracking algorithm*, in *Technical report 641*. 1996, University of Rochester, Computer Science Dept.
24. Koller, D., *Model-based object tracking in road traffic scenes*.
25. Lowe, D.G., *Three-dimensional object recognition from single two-dimensional images.* Artifical Intelligence, 1987. **3**(31): p. 353-395.
26. Thiang, A.T. Guntoro, and R. Lim. *Type of vehicle recognition using template matching method.* in *International Conference on Electrical, Electronics, Communication and Information*. 2001.
27. Leuven, J., M.B. Leeuwen, and F.C.A. Groen. *Real-time vehicle tracking in image sequences.* in *IEEE Instrumentation and Measurement Technology*. 2001.
28. Sonka, M., V. Hlavac, and R. Buyle, *Image processing, analysis, and machine vision*. 2 ed. 1998: ITP.
29. Beveridge, J.R., C. Graves, and C. Lesher, *Some lessons learned from coding the Burns line extraction algorithm in the DARPA image understanding environment*. 1996, Colorado State University.
30. Beveridge, J.R. and E.M. Riseman. *Hybrid weak-perspective and full-perspective matching.* in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 1992.
31. Borgefors, G., *Hierarchical Chamfer matching: a parametric edge matching algorithm.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 1988. **10**(6): p. 849-865.
32. Carceroni, R.L. and C.M. Brown, *Numerical methods for model-based pose recovery*. 1997, University of Rochester, Computer Science Dept.
33. Cucchiara, R., et al. *Statistic and knowledge-based moving object detection in traffic scenes.* in *International Conference on Intelligent Transportation Systems*. 2000.

34. Di Stefano, L. and E. Viarani. *Vehicle detection in traffic images.* in *1st International Conference on Enterprise Information Systems*. 1999.
35. Ferryman, J.M., A.D. Worrall, and S.J. Maybank. *Learning enhanced 3D models for vehicle tracking.* in *Proceedings of British Machine Vision Conference*. 1998.
36. Ferryman, J.M., et al., *A generic deformable model for vehicle recognition.* BMVC, 1995. **1**: p. 127-136.
37. Friedman, N. and S. Russell. *Image segmentation in video sequences: A probabilistic approach.* in *13th Conference on Uncertainty in Artificial Intelligence*. 1997.
38. Gavrila, D.M. and V. Philomin. *Real-time object detection for "smart" vehicles.* in *Proc. of IEEE Conference on Computer Vision*. 1999.
39. Haag, M. and H.H. Nagel, *Combination of edge element and optical flow estimates for 3D-model-based vehicles tracking in traffic image sequences.* International Journal of computer vision, 1999. **3**(35): p. 295-319.
40. Haralick, R.M. and L.G. Shapiro, *Computer and robot vision.* Vol. 1 & 2. 1992: Addison-Wesley Publishing Company.
41. Jain, A.K., *Object matching using deformable templates.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 1996. **18**(3): p. 267-278.
42. Koller, D. *Moving object recognition and classification based on recursive shape parameter estimation.* in *Proc. of the 12th Israeli Conf. on Artificial Intelligence, Computer Vision, and Neural Networks*. 1993.
43. Koller, D., et al. *Towards robust automatic traffic scene analysis in real-time.* in *International Conference on Pattern Recognition*. 1994.
44. Kollnig, H. and H.H. Nagel, *3D pose estimation by fitting image gradients directly to polyhedral models.* Proceedings of the 5th International Conference on Computer Vision, 1995.
45. Kollnig, H. and H.H. Nagel, *3D pose estimation by directly matching polyhedral models to gray value gradients.* International Journal of computer vision, 1997. **3**(23).
46. Mantri, S. and D. Bullock, *Analysis of feedforward-backpropagation neural networks used in vehicle detection.* Transportation Research, 1995. **3**: p. 161-174.
47. Matthews, N.D., et al., *Vehicle detection and recognition in greyscale imagery.* Control Engineering Practice, 1996. **4**(4): p. 473-479.
48. Mikic, I., et al. *Moving shadow and object detection in traffic scenes.* in *Intl'l Conference on Pattern Recognition*. 2000.
49. Prati, A., et al. *Analysis and detection of shadows in video streams: a comparitive evaluation.* in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2001.

50. Prati, A., et al. *Shadow detection algortihms for traffic flow analysis: a comparative study.* in *Proc. of IEEE Int'l Conference on Intelligent Transportation Systems*. 2001.
51. Sullivan, G.D. *Visual interpretation of known objects in constrained scenes.* in *Royal Society discussion meeting on natural and artificial low level seeing systems*. 1992.
52. Taktak, R., et al., *Analysis and inspection of road traffic using image processing.* Mathematics and Computers in Simulation, 1996. **41**: p. 273-283.
53. Tan, T.N. and K.D. Baker, *Efficient image gradient based vehicle localization.* IEEE Transactions on Image Processing, 2000. **9**(8).
54. Tan, T.N., G.D. Sullivan, and K.D. Baker, *Model-based localisation and recognition of road vehicles.* International Journal of computer vision, 1998. **1**(27): p. 5-25.