Towards a Non-Intrusive Traffic Surveillance System Using Digital Image Processing

by

Berino Lorio B.Eng. (Stell.)

A thesis submitted to the Faculty of Engineering
in partial fulfilment of the requirements for the degree
Master of Engineering (Civil Engineering: Transportation)
University of Stellenbosch

September 2001

Supervisors

Prof B.M. Herbst
Prof C.J. Bester

Stellenbosch University http://scholar.sun.ac.za

Declaration

I, the undersigned, do hereby declare that the work contained in this thesis is my own original work, unless stated otherwise, and has not been submitted in its entirety or partially to any other university for the purpose of obtaining a degree.

Stellenbosch

Stellenbosch University http://scholar.sun.ac.za

Abstract

With the increased focus on the use of innovative and state-of-the-art technology in Intelligent Transport Systems (ITS), the need for more accurate and more detailed road traffic flow data has become apparent. Data obtained from vehicle detector loops, which merely act as vehicle presence sensors, is neither reliable nor accurate enough anymore. This type of sensor poses the problem that it has to be inserted into the road surface; temporarily obstructing traffic flows, and has to be replaced after pavement reconstruction. One of the solutions to this problem is to develop a traffic surveillance system that uses video image processing.

In cities where Intelligent Transport Systems are used extensively, roadways are monitored through Closed Circuit Television Cameras (CCTV) that are closely watched by traffic control centre personnel. These cameras are mounted on posts on the roadside. These cameras can serve a dual purpose, being used for both human monitoring and as inputs to Video Image Processing Systems.

In this study some of the digital image processing techniques that could be used in a traffic surveillance system were investigated. This report leads the reader through the various steps in the processing of a scene by a traffic surveillance system based on feature tracking, and discusses the pitfalls and problems that are experienced.

The tracker was tested using three image sequences and the results are presented in the final chapter of this report.

Keywords: Computer Vision, Digital Image Processing, Video Image Processing, Feature Tracking, Traffic Surveillance Systems.

Stellenbosch University http://scholar.sun.ac.za

Opsomming

Met die toenemende fokus op die gebruik van innoverende oplossings en gevorderde tegnologie in Intelligente Vervoerstelsels, het die noodsaaklikheid van akkurater en meer gedetailleerde padverkeer vloeidata duidelik geword. Data wat verkry word d.m.v. voertuig deteksie lusse, wat alleenlik voertuig teenwoordigheid/afwesigheid meet, is nie meer akkuraat of betroubaar genoeg nie. Hierdie tipe sensors het egter die nadeel dat dit in die plaveisel ingesny moet word, dus vloei tydelik kan belemmer, en moet vervang word elke keer as plaveisel rekonstruksie gedoen word. Een van die oplossings vir hierdie probleem is om 'n verkeers waarnemingstelsel te ontwikkel wat van video-beeldverwerking gebruik maak.

In stede waar van uitgebreide intelligente verkeerstelsels gebruik gemaak word, word paaie gemonitor d.m.v. geslote baan televisiekameras wat op pale langs die paaie aangebring is. Personeellede van die verkeers beheer sentrum hou dan die inkomende televisiebeelde dop. Hierdie kameras kan 'n dubelle rol vervul deurdat dit vir beide menslike waarneming en as invoer in 'n video-beeldverwerking stelsel gebruik kan word.

In hierdie studie was verskeie digitale beeldverwerking tegnieke wat gebruik kan word in 'n verkeers waarnemingstelsel ondersoek. Hierdie verslag lei die leser deur die verskeie stappe in die verwerking van 'n toneel deur 'n verkeers waarneming stelsel wat gebaseer is op die volg van kenmerke. Die verslag beskryf ook die slaggate en probleme wat ondervind word.

Die voertuig volger was getoets deur van drie reekse beelde gebruik te maak en die resultate word weergegee in die finale hoofdstuk van hierdie verslag.

Sleutelwoorde: Rekenaar Visie, Digitale Beeldverwerking, Video-Beeldverwerking, Kenmerk Navolging, Verkeers Waarnemingstelsels.

Stellenbosch University http://scholar.sun.ac.za

Acknowledgements

I would like to thank the following people and organisations for their help and support, without which this project would not have been possible:

- My Heavenly Father, for his His grace in giving me the health and ability to perform this study.
- Prof B.M. Herbst for acting as supervisor and for his valuable advice, suggestions and supervision.
- Prof F. Hugo for the opportunity to study via the ITT internship.
- Prof C.J. Bester for his support and guidance through the course of my studies.

Stellenbosch University http://scholar.sun.ac.za

Contents

Stellenbosch University http://scholar.sun.ac.za

Stellenbosch University http://scholar.sun.ac.za

Stellenbosch University http://scholar.sun.ac.za

List of Figures

Stellenbosch University http://scholar.sun.ac.za

List of Tables

Stellenbosch University http://scholar.sun.ac.za

List of Abbreviations

NDoT	National Department of Transport
ITS	Intelligent Transport Systems
VMS	Variable Message Sign
CCTV	Closed Circuit Television
LOS	Level of Service
VIPS	Video Image Processing System
CCD	Charge Coupled Device
PAL	Phase Alternate Line
PC	Personal Computer
SSD	Sum of Squared Deviations

Stellenbosch University http://scholar.sun.ac.za

Chapter 1
Origin and objective of the study

1.1 Vision for SA transport in 2020

Moving South Africa is the National Department of Transport's (NDoT) 20-year strategic framework for the transport sector. *Moving South Africa – Action Agenda* provides the transition from the *White Paper on National Transport Policy* of 1996, which provides the guidelines on where to focus limited resources by guiding the actions of the different transport sector role players, to the eventual delivery of a seamless transportation system.

According to the vision statement in the *Action Agenda*, released in 1999 - "*By 2020, transport in South Africa will meet the needs of freight and passenger customers for accessible, affordable, safe, frequent, high quality, reliable, efficient and seamless transport operations and infrastructure. It will do so in a constantly upgrading, innovative, flexible and economically and environmentally sustainable manner*", [12].

There are however significant challenges that will have to be overcome in order to achieve the goals as set out by the *Moving South Africa* strategic framework.
It is these challenges that have caused more focus to be placed on the use of new and innovative technologies. The South African transport authorities have since placed more emphasis on the import and development of such technologies. The technologies that have been developed in order to improve the operation of transport systems worldwide have been grouped under the collective name "Intelligent Transport Systems", hereafter simply to be referred to as ITS.

Stellenbosch University http://scholar.sun.ac.za

1.2 What is ITS?

ITS has been defined by various organisations but according to the information document of the South African Society for ITS [13]: "*ITS involves the collection, processing, integration and supply of information through the application of computer, control and communications technologies to enable authorities, operators and individual customers to make better informed, i.e. more "intelligent" transport decisions. The overarching function of ITS is to improve the operations of transport systems which in turn will support the general transport objectives of mobility, safety, reliability, effectiveness, efficiency and environmental quality*"

Several ITS user service categories have been defined of which the most common are:

- Advanced Traffic Management
- Advanced Traveller Information
- Advanced Vehicle Control Systems
- Commercial Vehicle Operations
- Emergency Management
- Electronic Payment
- Public Transport
- Safety

In this study it is the advanced traffic management category that is of interest. The role that a traffic surveillance system should fulfil in a traffic management system will be discussed in the following section.

1.3 ITS and control systems theory

According to Ogata in *Modern Control Engineering* [1], "A system is a combination of components that act together and perform a certain objective". A system can however be static or dynamic; a static system has a fixed output whereas a dynamic system's output

Stellenbosch University http://scholar.sun.ac.za

is dependent on one or more inputs, and only once the system's output is dependent on one or more inputs can it be considered a Control System.

The term "intelligent" in Intelligent Transport Systems refer to the ability of the system to act on an input, causing an effect on the output of the system. For accurate control the output of the control system should be fed back as an input, this is called a feedback control system. In this context the study undertaken focussed on a method of obtaining an input or feedback for the control mechanisms in an Intelligent Transport System.

Using the analogy of a feedback control system, a typical traffic management system could be represented as shown in Figure 1.1.

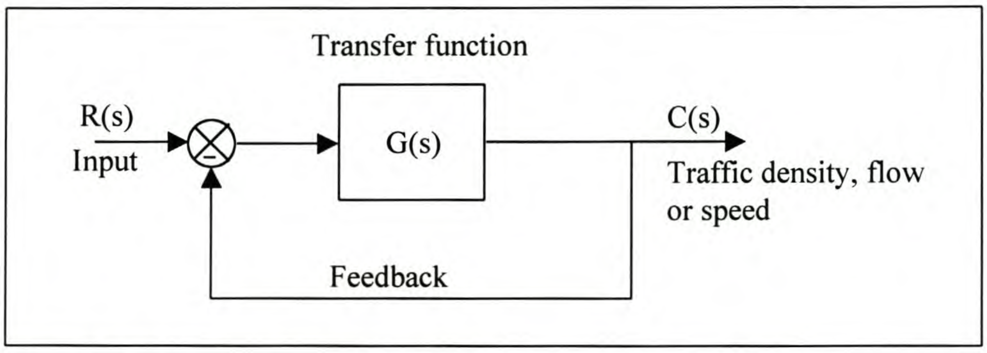


Figure 1.1: A traffic management feedback control system

In Figure 1.1 a simple feedback control system is presented. The left hand side shows an input, this could typically be a desired speed, density or flow. The input is then passed through a system with a certain transfer function .The transfer function, *G(s)*, is a mathematical function that determines the output signal from the input received and is often expressed in terms of the Laplace (frequency) domain. Note that the concept of a transfer function is being used as an analogy and that the transfer function of a real traffic management system probably would not be expressible in terms of the Laplace domain. The system could typically be in the form of a VMS (Variable Message Sign), a ramp metering system (controlling the access onto a freeway) or one or more traffic lights. The traffic stream should thus respond to the actions dictated by the traffic management system. The traffic stream however, may respond in an unpredictable manner to the

Stellenbosch University http://scholar.sun.ac.za

actions performed by the traffic management system, necessitating a feedback system. The traffic conditions are fed back as an input to the control system in order to change the output continuously and bring the traffic flow closer to the desired condition. Thus this system functions dynamically to control the traffic stream.
One of the problems of a system as described above is that constant and accurate information of traffic conditions are required as feedback for the control system.

The ideal traffic surveillance system will observe the traffic flow and as soon as the flow is impeded or reaches high density the surveillance system will warn of possible problems. In which case appropriate action is taken. This can include informing the emergency services in the case of an accident or breakdown, or the changing of the message displayed on a variable message sign (VMS) informing motorists of lane closures or blockages.

1.4 Objective of the study

Currently traffic counters are of the loop detector kind, which is merely a presence detector. This technology represents a "blind" type of detection where only the presence/absence of a vehicle can be assessed with high accuracy. Traffic parameters such as speed, density and vehicle classification have to be derived from presence/passage data and require multiple loops. Furthermore the loops are not reliable, and installation poses problems because traffic has to be redirected and loops have to be replaced after pavement reconstruction.

With the increase in deployment of ITS systems, the need for more detailed and more accurate information has arisen. This has lead to some new and innovative ideas on the collection of data.

The rapid increase in computer vision technologies and applications has led to the idea of using computer vision for traffic data collection. Since closed circuit television (CCTV)

Stellenbosch University http://scholar.sun.ac.za

cameras are used for manual freeway surveillance, it would be economical to use these same cameras for traffic data collection.

The objective of the study undertaken was therefore to investigate the use of image processing as a technique to monitor traffic flows and to extract relevant information from the images.

Stellenbosch University http://scholar.sun.ac.za

Chapter 2

Requirements for a traffic surveillance system and contributions so far

2.1 Requirements for a traffic surveillance system

The challenge lies in developing a single traffic surveillance system that will be able to perform the functions as outlined below:

- The traffic surveillance system must be able to assess traffic conditions accurately. In order to do so the three traffic parameters: speed, flow and density, need to be reported accurately.
- The system should be able to function under a wide range of traffic conditions – from light, fast moving traffic to slow moving, congested flows.
- The system should be able to operate under a wide variety of lighting and weather conditions.
- The traffic surveillance system should be able to operate in real-time.
- The traffic composition (percentage of heavy vehicles in traffic stream) also needs to be measured since heavy vehicles impede traffic flow. Alternatively stated, a heavy vehicle is equivalent to more than one light passenger vehicle.
- Advanced features could also include the detection of queues and queue lengths.
- The surveillance system could include individual traffic flow reports for each of the lanes on a multi-lane freeway, and include lane change counts, which could indicate lane blockages.

The main prerequisite for such a system is the ability to track the movement of the individual vehicles over several video frames. Since it is of primary importance, it forms the main focus of the study.

Stellenbosch University http://scholar.sun.ac.za

2.1.1 The traffic flow parameters

The three parameters can be defined as follows:

Density (K): The number of vehicles per unit distance normally expressed as vehicles per kilometre.

Speed (U): The speed of the traffic stream, normally expressed in km/hr.

Flow (Q): The number of vehicles that passes a line on the road surface perpendicular to the direction of travel, per time unit, normally expressed as vehicles per hour.

There are two different methods of calculating average speed. Papacostas [11] defines these as follows:

Time Mean Speed (also average spot speed or microscopic speed)

$$U_t = \frac{1}{N}\sum_{i=1}^{N} U_i = \frac{1}{N}\frac{\sum_{i=1}^{N}\Delta x_i}{\Delta t} \tag{2.1}$$

where N is the number of vehicles and U_i is the speed of the i^{th} vehicle. Δx_i refers to the distance the ith vehicle travelled and Δt to the time taken to travel the distance.

Space Mean Speed (also macroscopic speed)

$$U_s = \frac{1}{\frac{1}{N}\sum_{i=1}^{N}\frac{1}{U_i}} = \frac{N\Delta x}{\sum_{i=1}^{N}\Delta t_i} \tag{2.2}$$

where t_i is the time it took the ith vehicle to traverse the distance Δx.

There are various other ways to measure speeds, but generally the space mean speed is the proper stream average speed. The relationship between speed, flow and density has been obtained by using a macroscopic or fluid continuum (hydrodynamic analogy) approach. For the macroscopic approach the traffic flow has to be observed over a length of road (i.e. vehicles are observed in a space-time domain), which makes the space mean speed the correct measure of average speed to use in this case.

Stellenbosch University http://scholar.sun.ac.za

There are also ways to convert time mean speed to space mean speed, but that is not discussed here.

The relationship between the parameters is $Q = K \times U$, thus the flow is equal to the speed multiplied by the density. Figure 2.1 shows the relationship between the three parameters

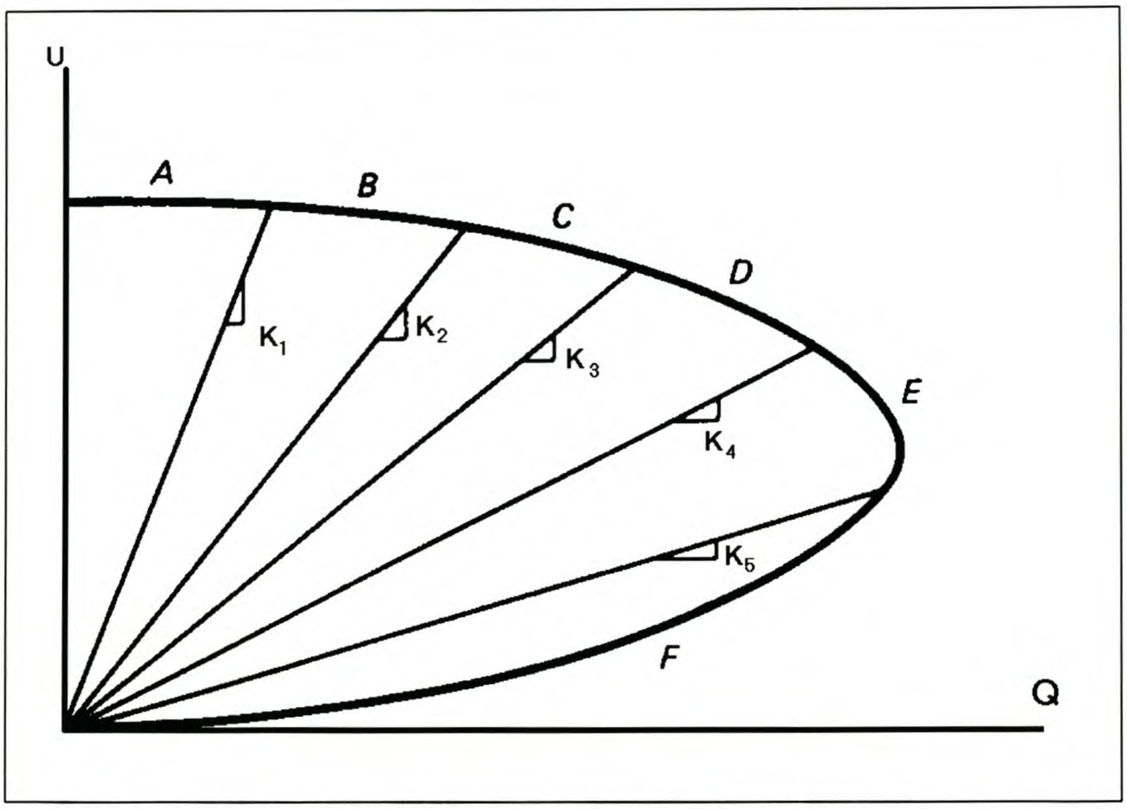


Figure 2.1: The relationship between the speed, flow and density of a traffic stream

The horizontal axis in the figure represents flow, the vertical axis represents speed and the inverse of the slope of a line from the origin of the graph to a point on the curve represents the density.

As can be seen from the graph, if the density is low or approaches zero, the vehicles travel at high speeds. If the density increases, the speed decreases slightly, and the flow increases. This trend continues until the optimum flow is reached. This is the rightmost section of the curve. If the density is increased further the speed continues to decrease

Stellenbosch University http://scholar.sun.ac.za

and the flow also starts to decrease. Further increases in the density will eventually cause traffic to come to a standstill in bumper-to-bumper type traffic.
The range of densities has also been classified into 6 Levels of Service (LOS), as can be seen in the above figure. The Levels of Service range from LOS A which represents free flow, to LOS F that defines breakdown flow.
It is very important that a traffic surveillance system is able to supply these three parameters in order to assess where on the speed/flow curve the traffic conditions are and what the Level of Service is, so that the appropriate action can be taken by the traffic management system.

2.2 Contributions

According to Coiffman [2] most of the commercial VIPS (Video Image Processing Systems) available today are *tripwire* systems (e.g. AUTOSCOPE, CCATS, TAS, IMPACTS and TraffiCam), which imitate the operation of loop detectors. These systems typically allow a user to specify several detection regions in the video image and then detects image intensity changes in the detection regions to indicate the presence of a vehicle. Some of these (e.g. IMPACTS) use a large number of detection zones to indicate the passage of vehicles, but they neither identify vehicles nor track them. The primary advantages of these systems are that they are not computationally intensive, and that it is easy to place/replace detector zones because there is no need to cut the pavement.

Commercial vehicle tracking systems include CMS Mobilizer, Eliop EVA, PEEK VideoTrak, NestorTrafficVision, and Sumitomo IDET. Generally these systems use region based tracking where the vehicles are segmented based on movement, but vehicles that are close to each other and thus partially obscured by each other, or linked by their shadows will merge into one object.

UNIVERSITEIT STELLENBOSCH
BIBLIOTEEK

Stellenbosch University http://scholar.sun.ac.za

Independent tests of commercial VIPS suggest that the systems have problems with congestion, high flow, occlusion, camera vibration due to wind, lighting transitions, and long shadows linking vehicles together [2].

Since then various research efforts were aimed at developing a traffic surveillance system that will be accurate under all conditions. This has led to the investigation of vehicle tracking by use of features, including colour features, to identify and track vehicles.

2.2.1 Vehicle tracking strategies

Multi-object tracking has received considerable attention in the computer vision field and much of the background work has been in non-transportation problems. From computer vision literature, the different tracking approaches for video data can be classified as follows:

3D Model based tracking

According to Coiffman [2], several research groups have previously investigated three-dimensional model-based vehicle tracking systems. The most serious weakness of these systems according to them is the reliance on detailed geometric object models. In practice it is not possible to obtain detailed models for all vehicles that could be found on the roadway.

Three-dimensional reconstruction, as explained in Costeira [3] is another option that can be explored but the drawback is that fairly large amounts of very accurate feature point locations are necessary.

Region based tracking

In this approach a 'blob' (connected region) is identified. This connected region is then associated with a vehicle and each vehicle is then tracked over successive frames using a cross-correlation measure. This process typically employs a background subtraction

Stellenbosch University http://scholar.sun.ac.za

technique. The background is repeatedly updated to accommodate changes in the weather and lighting conditions. The image is then segmented by subtracting the incoming image from the background image, and looking for pixels where the difference is larger than a specified threshold. This works fairly well under uncongested traffic conditions. However under congested conditions vehicles may partially occlude one another causing 'blobs' to become merged and thus merging two vehicles into one.

Active contour based tracking

A dual to the region based approach is tracking based on active contour models, or *snakes*. In this approach a representation of the bounding contour of the object is obtained and dynamically updated.
The problem with this approach is that initialisation is difficult and if a vehicle enters the detection region partially occluded then the system will group the vehicle with the occluding vehicle, and this will result in measurement errors.

Feature based tracking

An alternative to tracking whole objects is tracking easily identifiable features of the object. These features include points, corners and lines. The advantage of this approach is that even under partial occlusion, some features of a vehicle may remain visible and can thus be tracked. Feature tracking will be discussed in detail in Chapter 4.

Stellenbosch University http://scholar.sun.ac.za

2.3 Common problems and inadequacies

As mentioned before the problems with the current state of VIPS are:

Long shadows that link vehicles
Partial or full occlusion of vehicles
Camera vibration
Lighting transitions

Congestion and high flow (mentioned earlier) are not included here since they are considered the cause of occlusion and shadow linkages.

In order to develop a vehicle tracking method that is robust, these problems should be regarded carefully so that they can ultimately be overcome.

2.4 Hardware issues

A VIPS roughly comprises of the following hardware:

1) A CCD (Charge Coupled Device) camera. This device uses a light sensitive two-dimensional array of sensors to sample the image projected onto it at a rate of approximately 50 frames per second. This image is then coded into an electronic signal that can be decoded by televisions (e.g. the PAL standard).
2) A Video Image Digitiser. This piece of hardware decodes the PAL television signal into individual frames. In the case of this study only every second frame (25 frames per second) was used. The digitised image can then be resized; it does not have to be the same size as a standard television set picture.
3) One or more processors. The processors can range from Personal Computers to Dedicated Microprocessors implemented on a custom designed PC board.

Stellenbosch University http://scholar.sun.ac.za

In this study the footage was initially recorded onto videotape and then digitised into a large number of frames. These images were processed later using the digitised inter-frame time for the necessary calculations. In doing so a process that should operate in real-time can be simulated at a lower speed. All image processing was performed on greyscale images of size 240×320 with 256 intensity levels.

Stellenbosch University http://scholar.sun.ac.za

Chapter 3

Image segmentation

Image segmentation refers to the process of locating or isolating objects of interest. In order to track the desired objects it is necessary to remove all unwanted objects or information from the image. In this study the segmentation was done on the assumption that the objects to be tracked are the only moving objects in the frame sequences. This chapter discusses how this was accomplished.

3.1 Comparing images

Initially the effect of merely subtracting each of the frames from its predecessor and displaying the absolute value was investigated. An example of a scene is shown in Figure 3.1 and the image resulting from the subtraction of consecutive frames is shown in Figure 3.2. The resulting image is black with light strips along the moving edges.

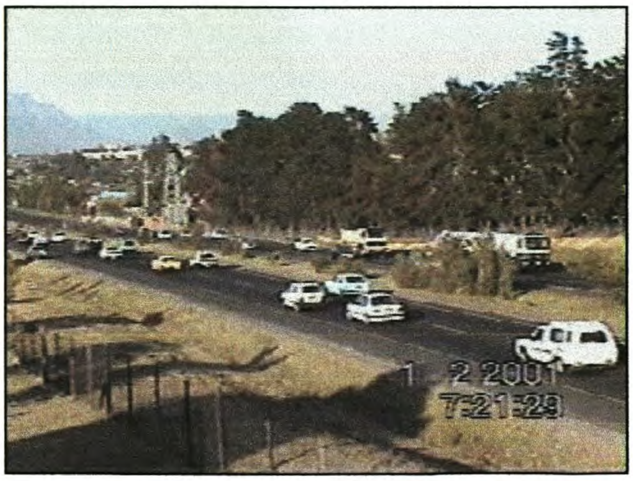


Figure 3.1: Frame used for image subtraction

Stellenbosch University http://scholar.sun.ac.za

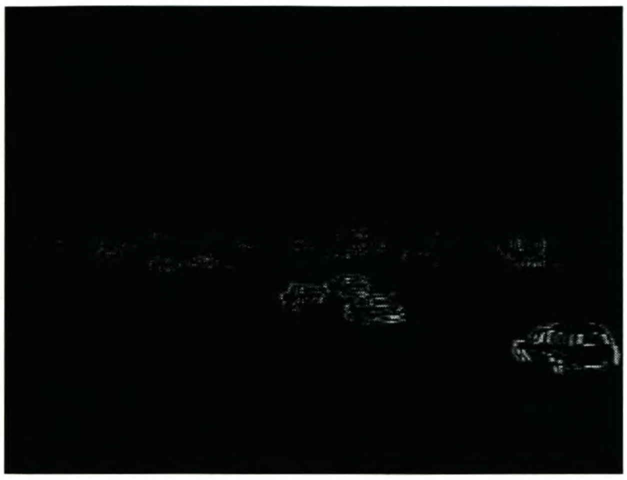


Figure 3.2: Resulting image from sequential image subtraction

The following step was to subtract the images from a background without vehicles, so that the entire vehicle would be displayed when an image from the sequence was subtracted from the background image. The calculation of the background image is described in section 3.2.

Figure 3.3 shows the resulting image when an image from the sequence was subtracted from the background image and the absolute value displayed.

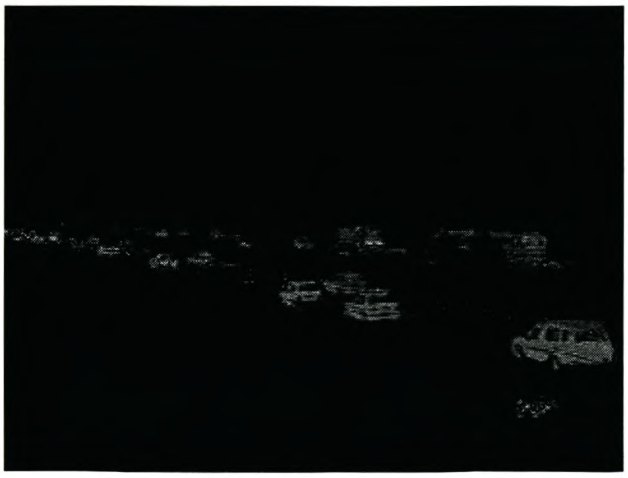


Figure 3.3: Result of subtracting the image from its background image

Stellenbosch University http://scholar.sun.ac.za

As can be seen in Figure 3.3 if the two images are subtracted from each other, the vehicles seem transparent and features from the background are still apparent through the vehicles.

The final step was to compare the background and sequence images, if the absolute value of the difference between the input image pixel value and the background pixel value was larger than a specified threshold, the values of the input image was kept, otherwise the value was set to 0.
This resulted in an image that was black except for the vehicles that remained their original grey colour. An example of such an image is shown in Figure 3.4.
The problem with comparing the image as just described is that they are prone to visual noise. If the intensity of the noise is slightly higher than the threshold used for the comparison, then the noise is more apparent against the all black background (appears as a spike or spot) than it otherwise would be.

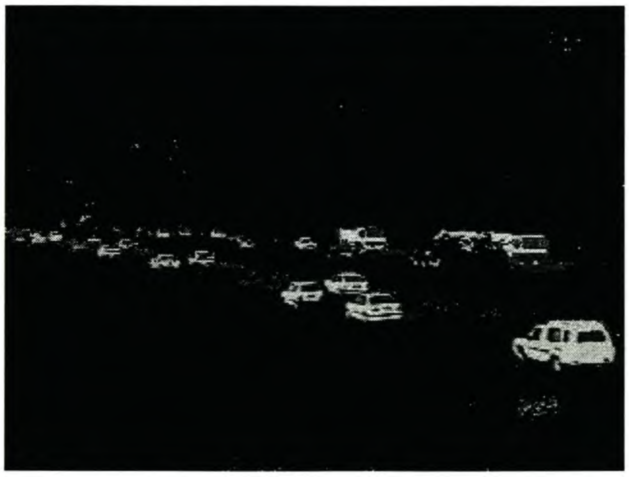


Figure 3.4: The result of comparing the original image to the background image

3.2 Computation of background image

Since the background image had to appear in the same lighting and weather conditions as the frame sequence, and since the background image had to be regularly updated it would not be possible to wait for an opportunity when there is no traffic on the road to

Stellenbosch University http://scholar.sun.ac.za

photograph or film the background image. For this reason a method had to be devised to obtain the background image from a sequence of images of the road, while there were vehicles present.

This method was based on the principle that if the intensity values at the same pixel position were recorded for several frames, these observations would have a high or peak concentration of values at a specific intensity level. Figure 3.5 indicates how the values are obtained and shows a typical histogram of the intensity values.

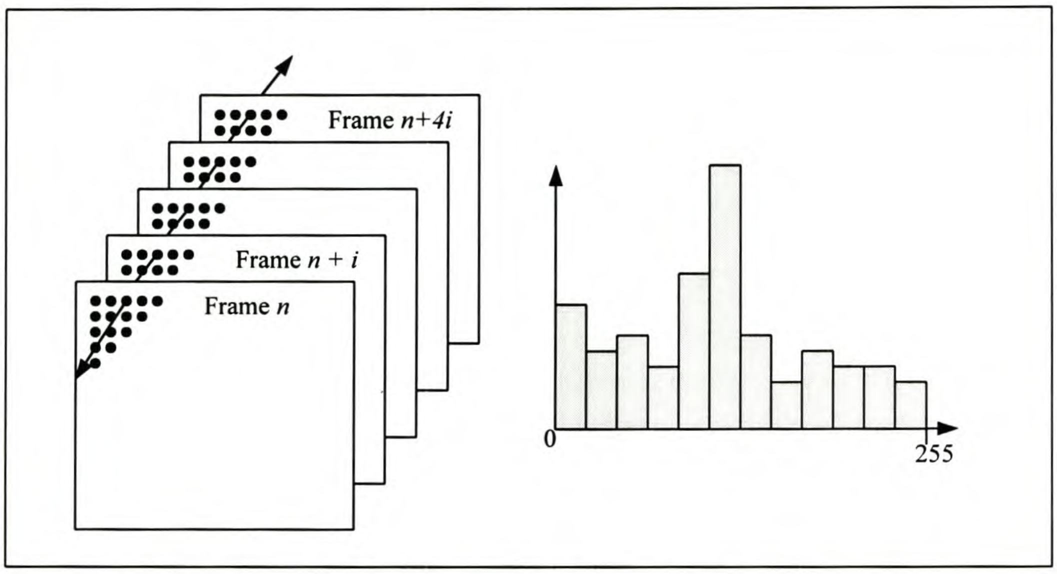


Figure 3.5: Collection of intensity values and a typical histogram

The left hand side shows how for instance the values for the pixel in the top row, third column are collected. The values would be collected at a base frame n and at each i^{th} frame thereafter until enough intensity values were recorded. The right side of figure 3.5 shows a hypothetical histogram of the recorded intensity values and shows a definite peak. The position of the peak will be different for different pixel locations. The form of the histogram can be explained in the following way. If a vehicle does not obscure the view onto the road surface or any other stationary object, the pixel value will remain relatively constant (the intensity of light from the road surface). If however a moving vehicle passes in front of the object, the pixel value will change, and since the vehicle can

Stellenbosch University http://scholar.sun.ac.za

have any colour and usually consists of various colours, the pixel value will be varying. Once the vehicle has passed the pixel value will return to the value of the stationary object or road. Thus it is expected that the intensity value of the background pixel will be at the intensity level where the highest concentration of recorded values for that pixel are found.

The reason for using every i^{th} frame (or every 20th frame in the case of thus study) is to reduce the possibility of having one vehicle obscure the road surface for the entire duration of the pixel value acquisition. This is possible since only 20 frames were used in this study and using sequential frames would result in a video clip of 0.8 seconds to be used. It is not uncommon for a single vehicle to be present over a single pixel for such a short time period.

Unfortunately, even if the pixel is located on a stationary object, its value will not be perfectly stationary and may vary by several intensity levels out of a greyscale of 256 levels. This is due to noise in the image acquisition system. For this reason an interval of 11 greyscale levels were used. Starting at the interval from 0 to 10, the number of values falling within this interval was noted and the interval shifted one pixel value up (interval becomes from 1 to 11). This process was repeated until the higher end of the interval reached 255. The middle value of the interval with the largest number of observations was then used as the background intensity value for that specific pixel. The bin size was found on a trial and error basis, and a bin size of 11 was found to perform satisfactory. If the bin size is too small, it is possible that the algorithm will not detect the correct location of the region of highest concentration and this will result in a speckled background. If the bin size is too large on the other hand, the region of highest concentration of values could lie anywhere within the large interval and the background intensity levels could be incorrect.
A background image of the same section as seen in Figures 3.1 to 3.3 is shown in Figure 3.6.

Stellenbosch University http://scholar.sun.ac.za

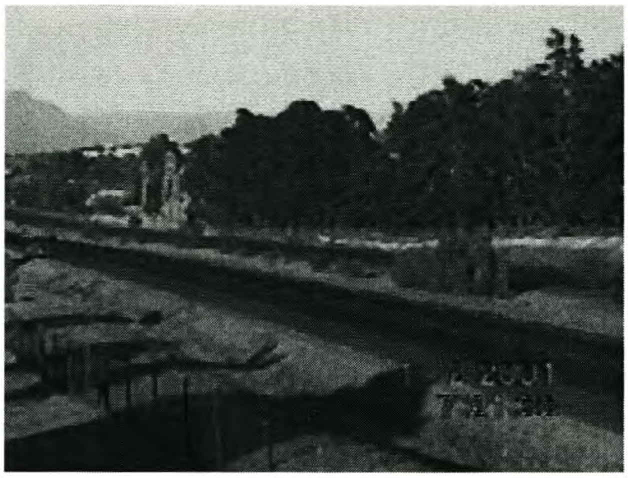


Figure 3.6: Calculated background image

3.3 Image enhancement

Imperfections in images due to noise in the image acquisition process, moving "stationary" objects such as trees swaying in the wind, and imperfect background images used for image comparisons, are unavoidable. An example of an image that has been compared with the background image as described in the last paragraph of section 3.1 is shown in Figure 3.4.

The noise (little light spots) in these images can be reduced by two techniques:

- Increasing the comparison threshold. With this procedure the noise with magnitude slightly larger than the threshold can be removed. The danger however exists that objects (vehicles) with colour intensity close to that of the surrounding area will be ignored or partially deleted.
- Median Filtering. This procedure uses a region with specific size and form (mask) around each pixel. It uses all the pixel values in this region and ranks them. It then uses the centre value of the ranked series (or average between the two centre values if an even number of values are used) as the pixel value in the centre of the region. See Gonzalez [10] for a more detailed description of the procedure. This procedure is very effective in eliminating small spikes or grains from an image.

Stellenbosch University http://scholar.sun.ac.za

In this study the segmented image obtained from the comparison of the input image with the background image was used to define a region in which to select features. This eliminated the possibility of tracking non-vehicles.

A reasonable comparison threshold (15 to 20, depending on the image sequence) was chosen to eliminate most of the noise. The remaining noise was removed by median filtering with a 3×3 window. A typical resulting image is shown in Figure 3.7.

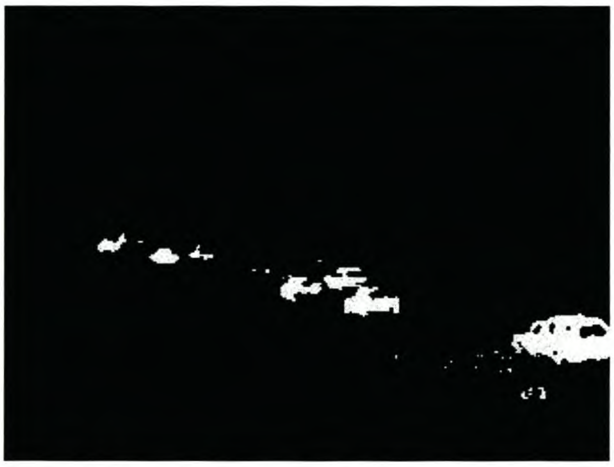


Figure 3.7: An image showing the segmented objects

3.4) Conclusion

The above sections have explained how the input image was segmented and processed in order to define a region within which features could be selected. Unfortunately, even after all of the above procedures have been employed to eliminate noisy errors, noise or some other disturbance may still cause regions outside the region of interest (road) to be segmented. An extra image that defines the region of interest has been included as part of the image segmentation process to curb this problem. Figure 3.8 shows an example of such an image. Any remaining objects falling outside this region is ignored.
The region of interest image is drawn manually as part of the calibration for each scene; however further research may allow the process to be automated.

Stellenbosch University http://scholar.sun.ac.za

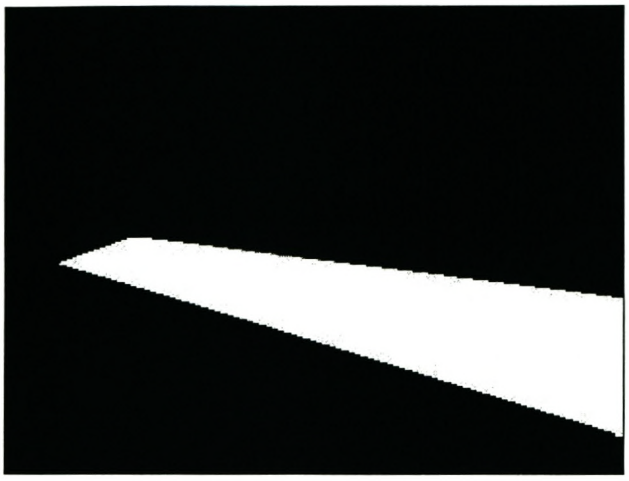


Figure 3.8: An image containing the location of the travelled way

All the above methods are used to improve the feature selection process, but the features themselves are tracked on the unprocessed greyscale input images, i.e. the segments provide a region of interest (in this case a moving vehicle) where features are selected.

Stellenbosch University http://scholar.sun.ac.za

Chapter 4

Feature tracking

This chapter will discuss feature selection, tracking and rejection in detail. Even though these processes were performed on the vehicle tracker it will be discussed in terms of general applications. The discussion is not confined to tracking vehicles.

4.1) Discussion of various optical flow techniques

A fundamental problem in the processing of image sequences is the measurement of optical flow or image velocity. Many methods for computing optical flow have been proposed, and more continue to be developed.

Baron et al [4] have evaluated some of the more common or well-known optical flow techniques. Following is a short description of each of the classes of optical flow techniques.

Differential techniques

Differential techniques compute velocity from spatiotemporal derivatives of image intensity or filtered versions of the image. The initial instances used first order derivatives and were based on image translation, i.e. given image I (or part thereof), then

$$I(\vec{x},t) = I(\vec{x} - \vec{v}t,0) \qquad (4.1)$$

where $\vec{v} = (u,v)^T$. This meant that after time t an object at position $\vec{x}$ would have moved to position $\vec{x} - \vec{v}t$ where $\vec{v}$ represents the optical flow velocity.

The Lucas and Kanade method, which will be discussed in detail in the next section, is a differential technique.

Stellenbosch University http://scholar.sun.ac.za

Differential techniques developed include those of:

- Horn and Schunck
- Lucas and Kanade
- Nagel
- Uras, Girosi, Verri and Torre

Consult Barron et al [4] for references.

Region based matching

Accurate numerical differentiation may be impractical due to noise, because a small number of frames exist or because of aliasing in the image acquisition process. In these cases differential approaches may be inappropriate and it is natural to turn to region based matching. Such approaches define velocity $\vec{v}$ as the shift $\vec{d} = (d_x, d_y)$ that yields the best fit between image regions at different times. Finding the best match amounts to maximizing a similarity measure (over $\vec{d}$), such as the normalised cross-correlation or minimizing a distance measure such as the sum-of-squared difference (SSD) with respect to $\vec{d}$:

$$\begin{aligned} SSD(\vec{x};\vec{d}) &= \sum_{j=-n}^{n} \sum_{i=-n}^{n} W(i,j)\left[I_1(\vec{x}+(i,j)) - I_2(\vec{x}+\vec{d}+(i,j))\right]^2 \\ &= W(\vec{x}) * \left[I_1(\vec{x}) - I_2(\vec{x}+\vec{d})\right]^2 \qquad (4.2) \end{aligned}$$

where W denotes a discrete 2-d window function, and $\vec{d} = (d_x, d_y)$ take on integer values. The * is the convolution operator.

There is a close relationship between the SSD distance measure, the cross-correlation similarity measure, and the differential techniques. Minimising the SSD distance amounts to maximising the integral of product term $I_1(\vec{x})I_2(\vec{x}+\vec{d})$. The difference in (4.2) can be seen as a window-weighted average of a first-order approximation to the temporal derivative of $I(\vec{x},t)$.

Stellenbosch University http://scholar.sun.ac.za

Region Based Matching Techniques include those of:

- Anandan
- Singh

Consult Barron et al [4] for references.

Energy based methods

The third class of optical flow techniques is based on the output energy of velocity-tuned filters. These are also called frequency-based methods owing to the design of velocity-tuned filters in the Fourier domain. The Fourier transform of a translating 2-d pattern as in equation (4.1) is

$$\hat{I}(\vec{k},\omega) = \hat{I}_0(\vec{k})\delta(\omega + \vec{v}^T\vec{k}), \qquad (4.3)$$

where $\hat{I}_0(\vec{k})$ is the Fourier transform of $I(\vec{x},0)$, $\delta(\vec{k})$ is a Dirac delta (or impulse) function, ω denotes temporal frequency and $\vec{k} = (k_x, k_y)$ denotes spatial frequency.

This shows that all non-zero power associated with a translating 2-d pattern lies on a plane through the origin in frequency space. Interestingly, according to [4], it has now been shown that certain energy-based methods are equivalent to correlation-based methods and to the gradient-based approach of Lucas and Kanade. Studies where both the Lucas-Kanade and Energy based methods have been used, reported similar results.

The technique developed by Heeger, as mentioned in Barron et al [4], is an example of an energy-based method.

Phase-based techniques

This class of methods is not discussed in this report and the reader is referred to Barron et al [4] for reference to the method of Fleet and Jepson.

Stellenbosch University http://scholar.sun.ac.za

Barron et al [4] evaluated all of the aforementioned techniques, and according to them "*Of these different techniques on the sequences we tested, we found that the most reliable were the first order, local differential method of Lucas and Kanade*"

Based on the statement above the decision was made to use the Lucas Kanade optical flow technique for the vehicle tracker.

4.2 The Lucas Kanade feature tracker

4.2.1 Theoretical derivations

As described in the previous section an image sequence can be represented by a function of three variables $I(x, y, t)$ where x and y are the spatial coordinates and t the temporal coordinate. Images taken at short time intervals are usually strongly related to each other because the images are taken from slightly different viewpoints. This correlation or moving patterns in the image stream can be expressed by:

$$I(x, y, t+\tau) = I(x-\xi, y-\eta, t) \tag{4.4}$$

This means that an image taken at time $t+\tau$ (or part thereof) can be obtained by moving every point or pixel $\vec{x} = (x, y)$ in the current image, taken at time t, by a displacement amount $\vec{d} = (\xi, \eta)$. The displacement between time instants t and $t+\tau$ is generally a function of x, y, t and τ.

The property described by (4.4) is unfortunately not satisfied in practice. Even when the image is taken of a rigid object, (4.4) is not true at occluding boundaries (since points in the image appears/disappears at occluding boundaries), or if the object's reflectivity changes as the angle with the camera changes.

The problem in finding $\vec{d}$ is that a single pixel cannot be followed from one frame to the next unless it has a distinct brightness compared to its surroundings. A picture element is

Stellenbosch University http://scholar.sun.ac.za

also subject to noise and can be confused with neighbouring pixels of similar brightness. Another problem is that a small point or feature does not necessarily move in such a way that it is located on a single pixel (does not displace in integer pixel values).

It is because of the above difficulties that windows or frames are used to track features. It is important that the window that is used contains sufficient information to be tracked. The criteria for window selection are discussed in detail in section 4.4.

Another problem that arises is the question of whether the same feature is being tracked from frame to frame, or whether the feature is occluded at some stage. Criteria for feature rejection are discussed in section 4.5.

Redefining $J(\vec{x}) = I(x, y, t+\tau)$ and $I(\vec{x}-\vec{d}) = I(x-\xi, y-\eta, t)$ in order to drop the time variable the image model is changed to

$$J(\vec{x}) = I(\vec{x}-\vec{d}) + \eta(\vec{x}) \tag{4.5}$$

where η is noise.

The displacement vector $\vec{d}$ is then defined as the displacement that will minimise the residual error defined by the following double integral over the window W:

$$\varepsilon = \int_W \left[I(\vec{x}-\vec{d}) - J(\vec{x})\right]^2 \omega d\vec{x}\,. \tag{4.6}$$

In this expression, ω is a weighting function. In the simplest case, ω could be set to 1. Otherwise ω could be a Gaussian-like function emphasising the central area of the window.

Several ways have been proposed in the literature to minimise the residual. According to Shi and Tomasi, 1991 [5], when the displacement $\vec{d}$ is much smaller than the window size, linearisation is the most efficient method. Thus if the inter-frame displacement is sufficiently small, the displacement vector is replaced by its linearisation:

$$I(\vec{x}-\vec{d}) = I(\vec{x}) - \vec{g}\cdot\vec{d}\,, \tag{4.7}$$

The residual is therefore rewritten as

Stellenbosch University http://scholar.sun.ac.za

$$\varepsilon = \int_W \left[I(\vec{x}) - \vec{g}\cdot\vec{d} - J(\vec{x}) \right]^2 \omega d\vec{x} = \int_W (h - \vec{g}\cdot\vec{d})^2 \omega d\vec{x} \qquad (4.8)$$

where $h = I(\vec{x}) - J(\vec{x})$.

This residual is a quadratic function of the displacement $\vec{d}$ that should have a local minimum close to the true displacement. Differentiating the last residual expression with respect to $\vec{d}$ and setting the result equal to zero yields the following vector equation:

$$\int_W (h - \vec{g}\cdot\vec{d})\vec{g}\omega dA = 0. \qquad (4.9)$$

Furthermore $(\vec{g}\cdot\vec{d})\vec{g} = (\vec{g}\vec{g}^T)\vec{d}$, and $\vec{d}$ is assumed to be constant within W.
This yields:

$$\left(\int_W \vec{g}\vec{g}^T \omega dA \right)\vec{d} = \int_W h\vec{g}\omega dA. \qquad (4.10)$$

This is a system of two equations that can be rewritten as

$$G\vec{d} = \vec{e} \qquad (4.11)$$

where the coefficient matrix is the 2×2 matrix

$$G = \int_W \vec{g}\vec{g}^T \omega dA \qquad (4.12)$$

and the right side is the two dimensional vector

$$\vec{e} = \int_W (I - J)\vec{g}\omega dA \qquad (4.13)$$

Equation (4.11) is the basic step of the tracking procedure. For every successive pair of frames, the matrix *G* is computed from one frame, by estimating gradients and computing their second order moments.
The vector $\vec{e}$ is computed from the difference between the two frames, along with the gradient computed above. The displacement $\vec{d}$ is then the solution to the system.

The solution to $\vec{d}$ is an approximation. It is however straightforward to further refine the estimate for $\vec{d}$ by using a Newton Raphson-like iterative technique. The process is iterated until the change in displacement is less than a specified threshold.

Stellenbosch University http://scholar.sun.ac.za

It is also important to notice that to solve for $\vec{d}$, G needs to be invertible which is equivalent to saying that I and J contains gradient information in both the x and y directions in the window W. We return to this question when we discuss the selection of appropriate features to track.

4.2.2 Physical interpretation

In order to understand the meaning of the above solution expression (4.9) can be derived in a more intuitive way. Consider the intensity function window, W, as in Figure 4.1. If a copy is made of it and placed on the first, the two will fit perfectly onto each other and there will be no space between the two surfaces. If however the one is slightly moved horizontally, a space between the two intensity patches will develop. The width of this gap measured vertically (the difference between the values of the two intensity profiles) is a function of the horizontal displacement between the two patches. It can further be shown that if the horizontal displacement is small, the vertical gap is related to the horizontal displacement through the image gradient at that point.

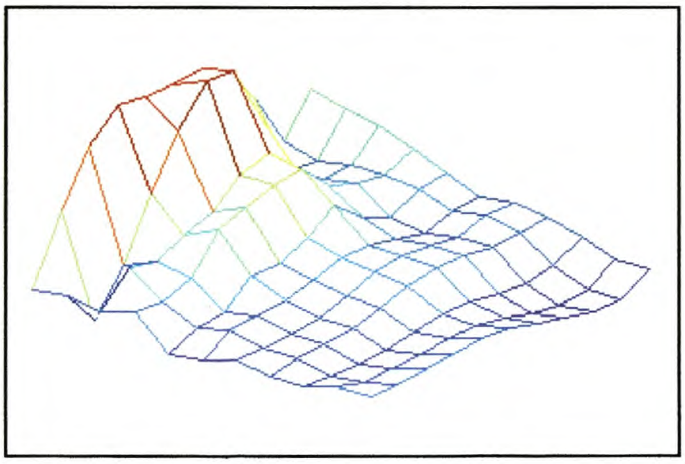


Figure 4.1: Example of an image intensity function in a window

Stellenbosch University http://scholar.sun.ac.za

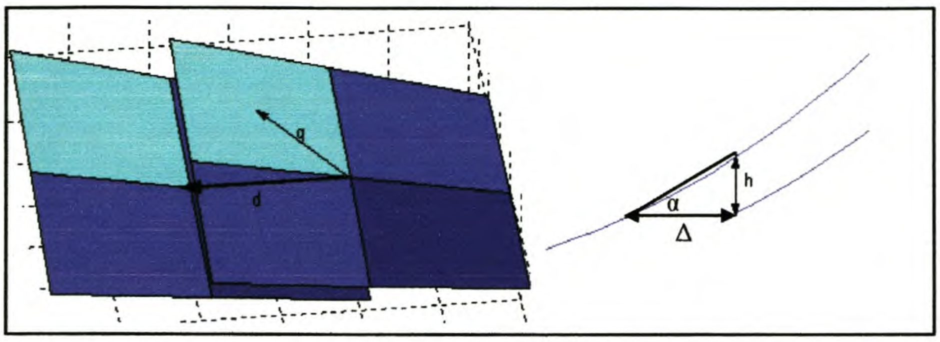


Figure 4.2: Example of an image patch and its translated counterpart showing the displacement and the gradient vectors (left), and a section through the same patches along the direction of the gradient showing the horizontal gap (right)

Figure 4.2 shows a small patch of the intensity function *I(x)* and the corresponding translated patch. The Figure also shows a cross section of the two patches along the direction of the image gradient.

The image gradient $\vec{g} = \left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y} \right)$, can be expressed as $\vec{g} = g\vec{u}$, where g is the magnitude of $\vec{g}$ and $\vec{u}$ is a unit vector.

The distance Δ measured along the gradient direction is the projection of $\vec{d}$ onto $\vec{u}$:

$$\Delta = \vec{d} \cdot \vec{u} .$$

From the right part of figure 4.2, it can be seen that the vertical gap width $h = I - J$ is

$$h = \Delta \tan\alpha ,$$

where α is the maximum slope of the patch. Since the tangent of α is equal to the magnitude g of the gradient, the following can be written:

$$h = \Delta g = \vec{d} \cdot \vec{u} g = \vec{d} \cdot \vec{g} .$$

By equating the first and last term, the equation relating the image gradient $\vec{g}$, the inter-frame displacement $\vec{d}$, and the difference h between image intensities are found:

$$\vec{g} \cdot \vec{d} = h \quad . \tag{4.14}$$

Stellenbosch University http://scholar.sun.ac.za

This is a scalar equation in the two-dimensional unknown $\vec{d}$. The image gradient $\vec{g}$ can be estimated from one image, while the difference h is easily computed from both.

The fact that the number of unknowns in equation (4.14) exceeds the number of constraints is called the *aperture problem* in the literature. If we only look at a small patch as in the last figure, at most one component of the displacement $\vec{d}$ can be determined.
If the whole window W is considered, different patches may allow the different components of the displacement vector to be calculated since $\vec{d}$ is assumed to be constant within W.

Measurements can now be combined. By observing that if the displacement $\vec{d}$ is assigned a wrong value equation 4.14 will not be true. The best value for $\vec{d}$ can be chosen as the one that will minimise the square of the differences (from equation (4.14)), integrated over the whole window. Thus the weighted residual

$$\int_W (h - \vec{g} \cdot \vec{d})^2 \omega dA$$

is minimised with respect to $\vec{d}$. This residual is equal to the residual as defined in equation (4.9).

4.2.3 Practical implementation

Up to now the optical flow algorithms have only been discussed in terms of a continuous image intensity function. In practice however the method has to be implemented on an image that consists of regularly spaced samples of image intensity i.e. discrete pixels.

In order to compare two windows from two consecutive images I and J, two smaller images, A and B, which contains the respective images from the windows, are defined as follows:

Let $W_{A[p_x,p_y]} = [p_x - v_x - 1, p_x + v_x + 1] \times [p_y - v_y - 1, p_y + v_y + 1]$

Stellenbosch University http://scholar.sun.ac.za

and $W_{B[p_x,p_y]} = [p_x - v_x, p_x + v_x] \times [p_y - v_y, p_y + v_y]$,

then

$$A = \{(x,y) | A \subseteq I ; (x,y) \in W_{A[p_x,p_y]}\} ,$$

$$B = \{(x,y) | B \subseteq J ; (x,y) \in W_{B[p_x,p_y]}\} ,$$

Here (p_x, p_y) is the point which is to be tracked and the centre of the images *A* and *B*, and $\vec{r} = (r_x, r_y)$ refers to an estimate of the displacement vector and $\vec{v} = (v_x, v_y)$ is half the horizontal and vertical window sizes. The use of the estimated displacement vector will be explained in the section dealing with the pyramidal implementation of the flow computation.

Also note that the domains of *A(x,y)* and *B(x,y)* are slightly different, *A(x,y)* is defined over a window size of (2v+3)×(2v+3) while *B(x,y)* is defined over a window size of (2v+1)×(2v+1).

This difference in size is due to the extra sample points necessary for the computation of the spatial derivatives that is performed on A. In order to compute the spatial derivatives (using the Sobel operator) on the edge of the window of size (2v+1)×(2v+1) requires that an extra row or column of data is available on each of the four sides of the image.

The displacement vector is changed in this section to $\vec{e} = [e_x, e_y]^T$ in order to avoid confusion later.

Equation (4.6) can now be rewritten as

$$\varepsilon(\vec{e}) = \sum_{x=p_x-v_x}^{p_x+v_x} \sum_{y=p_y-v_y}^{p_y+v_y} (A(x,y) - B(x+e_x, y+e_y))^2 . \qquad (4.15)$$

Just as before the derivative of this residual with relation to the displacement vector is obtained and equated to zero since

$$\left. \frac{\partial \varepsilon(\vec{e})}{\partial \vec{e}} \right|_{\vec{e}=\vec{e}_{opt}} = [0;0] . \qquad (4.16)$$

After linearisation the above equation becomes:

Stellenbosch University http://scholar.sun.ac.za

$$\frac{\partial\varepsilon(\vec{e})}{\partial(\vec{e})} = -2\sum_{x=p_x-v_x}^{p_x+v_x}\sum_{y=p_y-v_y}^{p_y+v_y}(A(x,y)-B(x+e_x,y+e_y))\cdot\begin{bmatrix}\frac{\partial B}{\partial x} & \frac{\partial B}{\partial y}\end{bmatrix} \tag{4.17}$$

which results in the following after the first order Taylor expansion about the point $\vec{e} = [0\ \ 0]^T$,

$$\frac{\partial\varepsilon(\vec{e})}{\partial(\vec{e})} = -2\sum_{x=p_x-v_x}^{p_x+v_x}\sum_{y=p_y-v_y}^{p_y+v_y}(A(x,y)-B(x,y)-\begin{bmatrix}\frac{\partial B}{\partial x} & \frac{\partial B}{\partial y}\end{bmatrix}\vec{e})\cdot\begin{bmatrix}\frac{\partial B}{\partial x} & \frac{\partial B}{\partial y}\end{bmatrix}. \tag{4.18}$$

The quantity *A(x,y)-B(x,y)* can be interpreted as the temporal derivative at the points $(x,y):\forall(x,y)\in W_{B[p_x,p_y]}$,

$$\delta I(x,y) = A(x,y)-B(x,y).$$

The matrix $\begin{bmatrix}\frac{\partial B}{\partial x} & \frac{\partial B}{\partial y}\end{bmatrix}$ is the gradient vector.

A slight change of notation is now introduced and the gradient vector becomes

$$\nabla I = \begin{bmatrix} I_x \\ I_y \end{bmatrix} = \begin{bmatrix}\frac{\partial B}{\partial x} & \frac{\partial B}{\partial y}\end{bmatrix}^T$$

Since it is assumed that the entire window is only translated, the image derivatives I_x and I_y can be computed from the first image *A(x,y)* in the (2v+1)×(2v+1) neighbourhood of point $\vec{p}$. The importance of this will become clear when the iterative technique is described.

In this study the Sobel operators for calculating derivatives are used. These are as follows:

$\forall(x,y)\in W_{B[p_x,p_y]}$

$$I_x(x,y) = \frac{A(x+1,y+1)+2A(x+1,y)+A(x+1,y-1)-A(x-1,y+1)-2A(x-1,y)-A(x-1,y-1)}{8}$$

$$I_y(x,y) = \frac{A(x+1,y+1)+2A(x,y+1)+A(x-1,y+1)-A(x+1,y-1)-2A(x,y-1)-A(x-1,y-1)}{8}$$

Equation (4.18) can now be written as:

$$\frac{1}{2}\frac{\partial\varepsilon(\vec{e})}{\partial(\vec{e})} \approx \sum_{x=p_x-v_x}^{p_x+v_x}\sum_{y=p_y-v_y}^{p_y+v_y}(\nabla I^T\vec{e}-\delta I)\nabla I^T, \tag{4.19}$$

Stellenbosch University http://scholar.sun.ac.za

$$\frac{1}{2}\frac{\partial\varepsilon(\vec{e})}{\partial(\vec{e})} \approx \sum_{x=p_x-v_x}^{p_x+v_x}\sum_{y=p_y-v_y}^{p_y+v_y}\left(\begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix}\vec{e} - \begin{bmatrix} \delta I\, I_x \\ \delta I\, I_y \end{bmatrix}\right). \tag{4.20}$$

Denote

$$G = \sum_{x=p_x-v_x}^{p_x+v_x}\sum_{y=p_y-v_y}^{p_y+v_y}\begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix} \text{ and } \vec{b} = \sum_{x=p_x-v_x}^{p_x+v_x}\sum_{y=p_y-v_y}^{p_y+v_y}\begin{bmatrix} \delta I\, I_x \\ \delta I\, I_y \end{bmatrix}$$

Then equation (4.20) can be written as

$$\frac{1}{2}\left[\frac{\partial\varepsilon(\vec{e})}{\partial\vec{e}}\right]^T \approx G\vec{e} - \vec{b}$$

The displacement vector is therefore given by:

$$\vec{e} = G^{-1}\vec{b}\,. \tag{4.21}$$

This agrees once again with equation (4.11) derived earlier.

This solution is however only valid for very small displacement because of the first order Taylor approximation that was used. To get a more accurate solution the method must be iterated.

Let k be the iterative index. At some iteration $k \geq 1$, assume that an initial estimate of the displacement $\vec{e}^{\,k-1} = \begin{bmatrix} e_x^{k-1} & e_y^{k-1} \end{bmatrix}$ has been obtained from previous iterations 1,2,…,k-1. Let B_k be the new translated image (translated by $\vec{e}^{\,k-1}$) according to the initial estimate.

$\forall(x,y) \in [p_x - v_x, p_x + v_x] \times [p_y - v_y, p_y + v_y]$

$$B_k(x,y) = B(x + e_x^{k-1}, + e_y^{k-1})$$

The goal is then to compute the residual motion vector $\vec{\eta}^{\,k} = \begin{bmatrix} \eta_x^k & \eta_y^k \end{bmatrix}$ that minimises the error function

$$\varepsilon(\vec{\eta}) = \varepsilon(\eta_x^k, \eta_y^k) = \sum_{x=p_x-v_x}^{p_x+v_x}\sum_{y=p_y-v_y}^{p_y+v_y}(A(x,y) - B_k(x+\eta_x^k, y+\eta_y^k))^2\,.$$

This solution can be obtained by the Lucas-Kanade optical flow computation

$$\vec{\eta}^{\,k} = G^{-1}\vec{b}_k$$

Stellenbosch University http://scholar.sun.ac.za

where the 2×1 vector b$_k$ (also called the image mismatch vector) is defined as follows:

$$\vec{b}_k = \sum_{x=p_x-v_x}^{p_x+v_x} \sum_{y=p_y-v_y}^{p_y+v_y} \begin{bmatrix} \delta I_k(x,y)\, I_x(x,y) \\ \delta I_k(x,y)\, I_y(x,y) \end{bmatrix}$$

and the image difference δI_k is defined as:

$\forall (x,y) \in [p_x - v_x, p_x + v_x] \times [p_y - v_y, p_y + v_y]$,

$$\delta I_k(x,y) = A(x,y) - B_k(x,y).$$

Note that it is only necessary for the spatial derivatives to be computed once at the beginning of the iterations. The 2×2 matrix G thus remains constant through the iterations. It is only the image mismatch vector b$_k$ that has to be recomputed at each iteration. Once the residual optical flow $\vec{\eta}^k$ is computed the displacement estimate is updated for the next iteration

$$\vec{e}^{\,k} = \vec{e}^{\,k-1} + \vec{\eta}^k.$$

The above process is iterated until the magnitude of the computed optical flow vector is lower than a threshold (say 0.03 pixels), or the iterations exceed a maximum number of iterations (say 20). The initial estimate of the displacement vector (k=1) is set to 0 ($\vec{e}^{\,0} = [0\ \ 0]^T$).

4.2.4 Subpixel computation

It is essential that computations be done at subpixel accuracy. It is therefore necessary to calculate image intensity values at non-integer coordinates (between pixels). To obtain these intensity values image intensity has to be interpolated using surrounding pixel values. In this study bilinear interpolation was used for this purpose.

Assume that the image intensity values are needed at *I(x,y)* where *x* and *y* are not integers. Let x_0 and y_0 be the integer parts of *x* and *y*, and let α_x and α_y be the two remainder values (between 0 and 1) such that:

Stellenbosch University http://scholar.sun.ac.za

$$x = x_0 + \alpha_x$$

$$y = y_0 + \alpha_y$$

then *I(x,y)* my be computed by bilinear interpolation from the original intensity values by use of:

$$I(x,y) = (1-\alpha_x)(1-\alpha_y)I(x_0,y_0) + \alpha_x(1-\alpha_y)I(x_0+1,y_0) + \\ (1\text{-}\alpha_x)\alpha_y I(x_0,y_0+1) + \alpha_x\alpha_y I(x_0+1,y_0+1).$$

It is important to note that when computing the image derivatives $I_x(x,y)$ and $I_y(x,y)$ in the neighbourhood $W = \{(x,y) \in [p_x - v_x, p_x + v_x] \times [p_y - v_y, p_y + v_y]$ it is necessary to use the set of original intensity values (from image *I(x,y)* and not from derived windows) in the integer grid patch $(x,y) \in [p_x - v_x - 1, p_x + v_x + 1] \times [p_y - v_y - 1, p_y + v_y + 1]$. But $\vec{p} = [p_x \;\; p_y]$ are not guaranteed to be integer values. If $\vec{p}$ is broken into its integer and non-integer parts

$$p_x = p_{x_0} + p_{x_\alpha}$$
$$p_y = p_{y_0} + p_{y_\alpha}$$

where p_{x_0} and p_{y_0} are the integer parts and, p_{x_α} and p_{y_α} are the remainder values, then it can be seen that the presence of remainder values and subsequent interpolation will necessitate the use of the original intensity values from the integer grid patch

$W = \{(x,y) \in [p_{x_0} - v_x - 1, p_{x_0} + v_x + 2] \times [p_{y_0} - v_y - 1, p_{y_0} + v_y + 2]$

to compute the image patch *I(x,y)* in the neighbourhood

$W = \{(x,y) \in [p_x - v_x - 1, p_x + v_x + 1] \times [p_y - v_y - 1, p_y + v_y + 1]\}$

so that the spatial derivatives can be calculated.

A similar situation occurs when computing the image difference. $\delta I_k(x,y)$ in the neighbourhood $W = \{(x,y) \in [p_x - v_x, p_x + v_x] \times [p_y - v_y, p_y + v_y]$. In order to compute $\delta I_k(x,y)$ it is necessary to know the values: $J(x + r_x + e_x^{k-1}, y + r_y + e_y^{k-1})$ for all $(x,y) \in [p_x - v_x, p_x + v_x] \times [p_y - v_y, p_y + v_y]$, or said differently $J(\alpha, \beta)$ has to be

Stellenbosch University http://scholar.sun.ac.za

known for all

$$(\alpha,\beta) \in \left[p_x + r_x + e_x^{k-1} - v_x, p_x + r_x + e_x^{k-1} + v_x\right] \times \left[p_y + r_y + e_y^{k-1} - v_y, p_y + r_y + e_y^{k-1} + v_y\right].$$

Since $p_x + r_x + e_x^{\;k-1}$ and $p_y + r_y + e_y^{\;k-1}$ are not necessarily integers, let q_{x_0} and q_{y_0} again be the integer parts, and q_{x_α} and q_{y_α} the remainder values (between 0 and 1) of q_x and q_y so that:

$$p_x + r_x + e_x = q_{x_0} + q_{x_\alpha}$$
$$p_y + r_y + e_y = q_{y_0} + q_{y_\alpha}$$

Then in order to compute the image patch $J(\alpha,\beta)$ in the neighbourhood

$$W = \left\{(\alpha,\beta) \in \left[p_x + r_x + e_x^{k-1} - v_x, p_x + r_x + e_x^{k-1} + v_x\right] \times \left[p_y + r_y + e_y^{k-1} - v_y, p_y + r_y + e_y^{k-1} + v_y\right]\right\}$$

it is necessary to use the intensity values from the original image $J(\alpha,\beta)$ in the integer grid patch $(\alpha,\beta) \in \left[q_{x_0} - v_x, q_{x_0} + v_x + 1\right] \times \left[q_{y_0} - v_y, q_{y_0} + v_y + 1\right]$.]

4.3 Pyramidal implementation

Two of the key features of a feature tracker are accuracy and robustness. Accuracy refers to the ability of the tracker to track features very closely. This means that difference between the measured and actual displacement must be very small, typically smaller than a single pixel. A small integration window (small values of v_x and v_y) seems preferable so that the detail in the window would not be “smoothed out” as would be the case in a large tracking window. Furthermore a small tracking window would stand less chance of straddling a border where two patches on the image move at different speeds (such as at occluding boundaries).

Robustness refers to the ability of the tracker to operate and give accurate results under various environments such as different lighting conditions and different magnitudes of image velocity. For robustness a larger tracking window is preferable since larger image motions can be accommodated. There is thus a trade-off in the tracking window size between accuracy and robustness. One solution to this problem is a pyramidal

Stellenbosch University http://scholar.sun.ac.za

implementation. The pyramidal implementation on the other hand requires more calculations and causes processing delays.

The pyramid representation of image I of size $n_x \times n_y$ is defined as follows:
Let I^0 be the initial image. This image is the highest resolution or largest image. The image size is $n^0_x \times n^0_y$, which is the size of the original image. The pyramid representation is then built in a recursive fashion where I^{b+1} is computed from I^b.

Let L be the image level index so that I^L, n_x^L and n_y^L refer to the image level L, the image width and image height respectively. Image I^L is then defined as follows:

$$\begin{aligned} I^L(x,y) = {} & \frac{1}{4} I^{L-1}(2x,2y) + \\ & \frac{1}{8}(I^{L-1}(2x-1,2y) + I^{L-1}(2x+1,2y) + I^{L-1}(2x,2y-1) + I^{L-1}(2x,2y+1)) + \\ & \frac{1}{16}(I^{L-1}(2x-1,2y-1) + I^{L-1}(2x+1,2y-1) + I^{L-1}(2x-1,2y+1) + I^{L-1}(2x+1,2y+1)) \end{aligned} \tag{4.22}$$

The result is that the pyramid is built up of images containing the same scene, but each level is a quarter the size of the previous. An example of the images making up a pyramid with 3 levels (four images including the base image) is shown below in Figure 4.3.

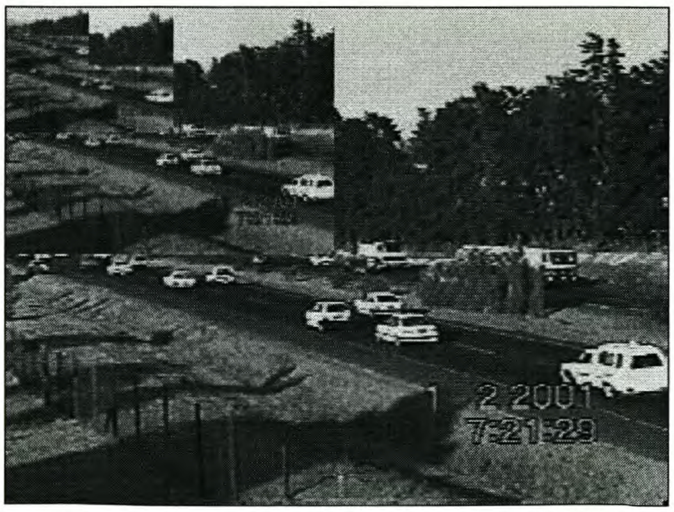


Figure 4.3: Images contained in an image pyramid

Stellenbosch University http://scholar.sun.ac.za

Depending on whether n_x is an even or odd value and whether the numbering of the pixels starts at 0 or 1, it might be necessary to add dummy pixels on one or both sides of the image. The same holds for n_y. These dummy pixel values were computed as follows in this study:

$$\begin{gathered}
I^{L-1}(-1,-1) = I^{L-1}(0,0), \\
I^{L-1}(-1,y) = I^{L-1}(0,y), \\
I^{L-1}(x,-1) = I^{L-1}(x,0), \\
I^{L-1}(n_x^{L-1},y) = I^{L-1}(n_x^{L-1}-1,y), \\
I^{L-1}(x,n_y^{L-1}) = I^{L-1}(x,n_y^{L-1}-1), \\
I^{L-1}(n_x^{L-1},n_y^{L-1}) = I^{L-1}(n_x^{L-1}-1,n_y^{L-1}-1).
\end{gathered}$$

Then $I^L(x,y)$ as defined above is only defined for values of x and y such that $0 \le 2x \le n_x^{L-1}-1$ and $0 \le 2y \le n_y^{L-1}-1$. Therefore n_x^L and n_y^L are the largest integer values that satisfy

$$n_x^L \le \frac{n_x^{L-1}+1}{2} \tag{4.23}$$

$$n_y^L \le \frac{n_y^{L-1}+1}{2} \tag{4.24}$$

Equations (4.22) to (4.24) are used to construct the pyramid up to level m. Practical values of m are 2,3,4. For typical images it is excessive to go higher than level 4. For example, for an image of size 640×480 the images I^1, I^2, I^3 and I^4 are of respective sizes 320×240, 160×120, 80×60 and 40×30. Considering the implications of tracking features close to image boundaries (discussed in the following section), using pyramids with large numbers of levels, causes a large part of the image to become useless for tracking features.

4.3.1 Tracking features close to image boundaries

It is essential that the entire image window used for tracking a feature fall entirely within the boundaries of the image. If part of the tracking window falls outside the image boundaries the feature cannot be tracked and must be declared lost. There is thus a region

Stellenbosch University http://scholar.sun.ac.za

in an image in which a feature cannot be tracked. Considering that the entire $(2\nu_x+1)\times(2\nu_y+1)$ window has to be within the image, there is a band of width ν_x (and ν_y) around the original image *I* that cannot be used. If a pyramidal implementation of height *m* is used this band is increased to a width of $2^m\nu_x$ (and $2^m\nu_y$) around the original image *I*. For small values of ν_x, ν_y and *m* with large images, this might not be a significant limitation, but it may become a problem for large values of ν_x, ν_y and *m* with small images. For instance $\nu_x=\nu_y$=5 pixels and m =3, causes a band of 40 pixels around the original image that cannot be used.

4.3.2 Summary of the pyramidal tracking algorithm

The algorithm in a pseudo code is taken from [14].

Goal: Let $\vec{u}$ be a point on image I. Find its corresponding location $\vec{v}$ on image J.

```
Build pyramid representations of I and J:   $\{I^L\}_{L=0,\dots,m}$ and $\{J^L\}_{L=0,\dots,m}$
Initialisation of pyramidal guess:   $\vec{r}^m = [r_x^m \;\; r_y^m]^T = [0 \;\; 0]^T$
for $L = m$ down to 0 with step of $-1$
    Location of point $\vec{u}$ on image $I^L$:   $\vec{u}^L = [p_x \;\; p_y]^T = \vec{u}/2^L$
    Derivative of $I^L$ with respect to x:
$$I_x(x,y) = \frac{A(x+1,y+1)+2A(x+1,y)+A(x+1,y-1)-A(x-1,y+1)-2A(x-1,y)-A(x-1,y-1)}{8}$$
    Derivative of $I^L$ with respect to y:
$$I_y(x,y) = \frac{A(x+1,y+1)+2A(x,y+1)+A(x-1,y+1)-A(x+1,y-1)-2A(x,y-1)-A(x-1,y-1)}{8}$$
    Spatial gradient matrix:   $G = \sum_{x=p_x-\nu_x}^{p_x+\nu_x} \sum_{y=p_y-\nu_y}^{p_y+\nu_y} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$
    Initialisation of iterative Lucas-Kanade:   $\vec{e}^0 = [0 \;\; 0]^T$
    for $k$=1 to $K$ with step of 1 (or until $\|\vec{\eta}^k\|$< accuracy threshold)
```

Stellenbosch University http://scholar.sun.ac.za

```
                Image difference:
        $\delta I_k(x,y) = A(x,y) - B_k(x,y)$
                Image mismatch vector:
        $\vec{b}_k = \sum_{x=p_x-v_x}^{p_x+v_x} \sum_{y=p_y-v_y}^{p_y+v_y} \begin{bmatrix} \delta I_k(x,y)\, I_x(x,y) \\ \delta I_k(x,y)\, I_y(x,y) \end{bmatrix}$
                Optical flow (Lucas-Kanade):        $\vec{\eta}^k = G^{-1}\vec{b}_k$
                Guess for next iteration:           $\vec{e}^k = \vec{e}^{k-1} + \vec{\eta}^k$
        **end of for-loop on** *k*
        Final optical flow at level L:              $\vec{d}^L = \vec{e}^K$
        Guess for next level L-1:
        $\vec{r}^{L-1} = \left[ r_x^{L-1} \;\; r_y^{L-1} \right]^T = 2\left(\vec{r}^L + \vec{d}^L\right)$
**end of for-loop on** *L*
Final optical flow vector                           $\vec{d} = \vec{r}^0 + d^0$
Location of point on J:                             $\vec{v} = \vec{u} + \vec{d}$
```

Solution: The corresponding point is at location $\vec{v}$ on image *J*

Stellenbosch University http://scholar.sun.ac.za

4.4 Choice of tracking window size and pyramid height

Appendix A contains figures of tracking lines for various tracking window sizes and various pyramid heights. The figures contained in Appendix A is similar to Figure 4.4 shown below with the exception that the background is not present (only the tracking lines are shown) and the lines are not distorted due to the pixel representation.

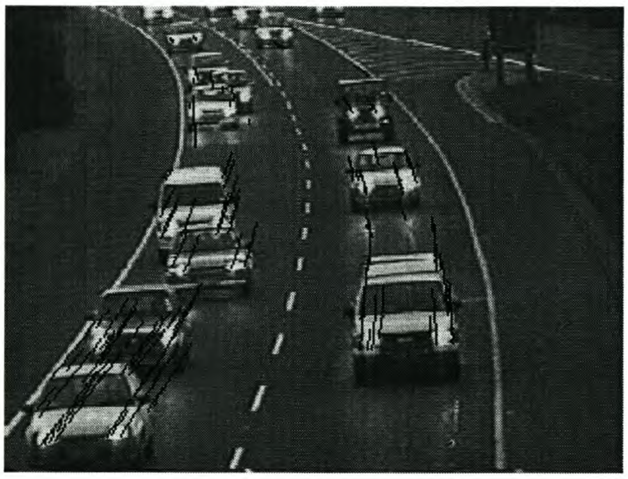


Figure 4.4: A frame from one of the image sequences showing the superimposed tracking lines

The tracking lines in Appendix A have been obtained by tracking features in the same sequence as Figure 4.4 above.

The figures in Appendix A are used to show the effect of the tracking window size and pyramid height on the accuracy or variability of the tracking process.

It can be seen that the larger the tracking window becomes the smoother the lines become. This seems to indicate that the larger the tracking window size, the more accurate the tracker becomes. An explanation for this is that more information is contained in larger tracking windows and noisy spikes in the tracking window do not have such a pronounced effect.

A problem that arises as the tracking window is enlarged is that the tracking window might contain parts of the surrounding environment instead of containing only a part of

Stellenbosch University http://scholar.sun.ac.za

the object that is to be tracked. The surrounding environment will then influence the tracking process and cause the tracking to be inaccurate. There is thus a trade-off to obtain the best tracking window size. This window size can be different for different camera positions.

From Figures 4.6 and 4.7 it can be seen that the best tracking window size for Figure 4.6 will likely be larger than that for Figure 4.7 simply because the vehicles or moving objects are larger on the first image (in relation to the size of the image).

A representation of some tracking window sizes have been included in the bottom of Figures 4.6 and 4.7, starting from left to right these are 5×5, 7×7, 9×9, 11×11, 13×13, 15×15 and 17×17.

Considering the height of the pyramidal representation, it can be seen from Figure A4 as well as Figures A8 to A10 in Appendix A that the tracking lines do become smoother and more accurate as the pyramid height is increased. The negative effect of the forbidden border around the image that cannot be used for tracking (see section 4.3.1) is very noticeable when the pyramid height is increased to 3 as shown below on the right hand side of Figure 4.5 and Figure A10 in Appendix A. Figure 4.5 shows up to where features were tracked before they were discarded with no pyramid on the left hand side and a pyramid of level 3 on the right hand side.

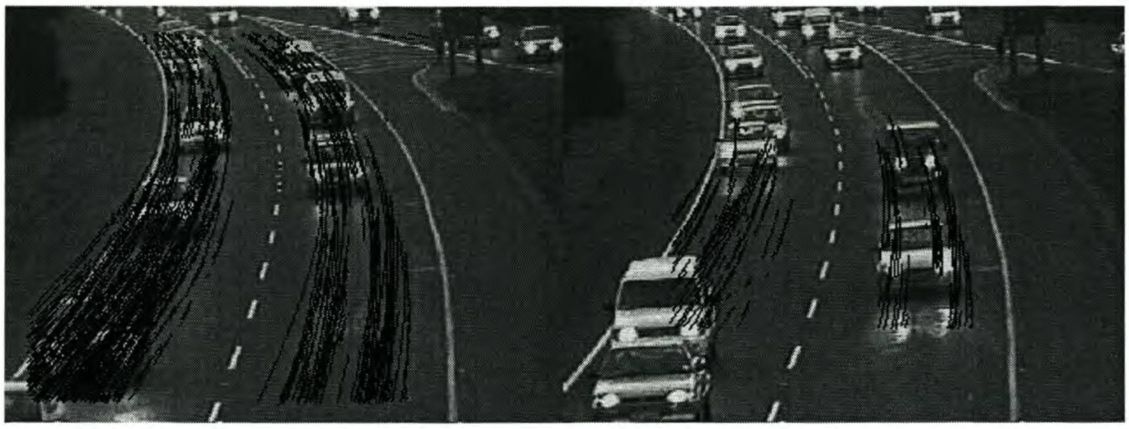


Figure 4.5: Superimposed tracking lines showing the effect of borders for no pyramid (left) and pyramid height of 3 (right).

In Figure A10 it can be seen that a noticeably smaller amount of features could be tracked through the full 30 frames of the sequence without being discarded due to their proximity

Stellenbosch University http://scholar.sun.ac.za

to the image boundary. In this study it was found that a feature tracking window of 11×11 and a pyramid height of 1 (base image and one level up) was sufficient for tracking in most of the sequences used.

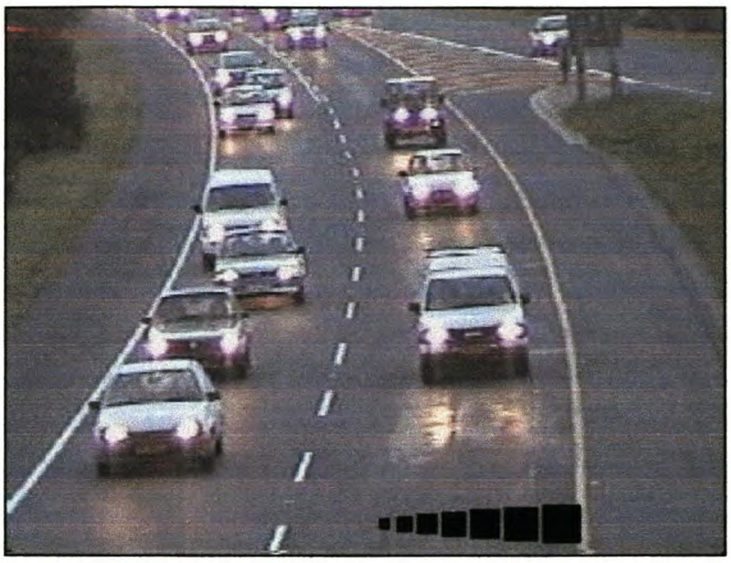


Figure 4.6: Longer focal length image showing tracking window sizes

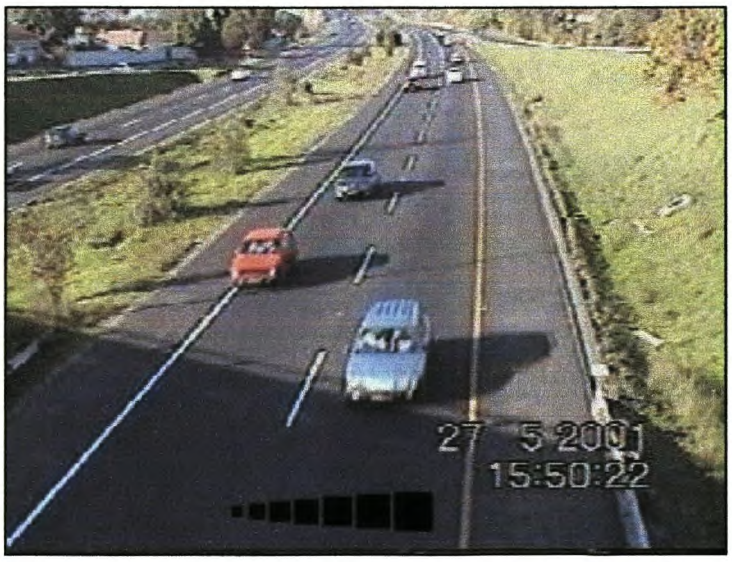


Figure 4.7: Shorter focal length image showing tracking window sizes

Stellenbosch University http://scholar.sun.ac.za

4.5 Feature selection

Not all parts of an image contain motion information, for example regions where the image intensity is constant over the whole tracking window. Some parts of image, such as straight edges, only contain motion information in the direction orthogonal to the edge. For this reason it is necessary that features be selected in such a way that they are tracked easily, reliably and accurately.

Various criteria have been advised for the selection of features or feature windows. These included the tracking of corners, windows with high spatial frequency content, or regions where some mix of second order derivatives are sufficiently high. These definitions all yield trackable features, but use preconceived ideas of what constitutes a good tracking window. In this study features were selected as described in [14]. In this method features are selected based on the Lucas-Kanade Tracker. It involves finding features that are best suited to the specific algorithm.
A good tracking window is a window of which the coefficient matrix (the 2×2 matrix G) is above the noise level and well conditioned. The noise requirement means that both the eigenvalues of *G* must be large, and the well-conditioned requirement means that the eigenvalues should be of the same order of magnitude.

Two small eigenvalues mean a roughly constant or flat intensity profile in the window, one large eigenvalue means there is a unidirectional pattern or line, and two large eigenvalues means that there is gradient information in both directions and the window can thus be tracked reliably.

The method is as follows:

1) Compute the matrix *G* and its minimum eigenvalue λ_m at every pixel in the image *I*.
2) Call λ_{max} the largest of λ_m over the whole image.

Stellenbosch University http://scholar.sun.ac.za

3) Retain the image pixels that have a λ_m larger than a percentage of λ_{max}. This threshold is typically chosen to be about 10%.
4) From those pixels only the local maximum in a specific size neighbourhood is kept. (Keep the pixel with the largest λ_m in a neighbourhood of 3×3 pixels).

These features are the ones that will be tracked by the feature tracker. It should be noted that the window for feature selection could be smaller than the window used for tracking (the window used to compute the G matrix). For feature selection a 3×3 window is typically large enough, but would be too small for tracking purposes.

4.6 Feature monitoring

4.6.1 Calculating residuals using affine deformation

Once a feature has been selected for tracking it has to be monitored through the tracking process to ensure that it is still tracking the feature it was initially selected to track. This means that a feature-tracking window has to be compared to its counterpart in a previous frame to determine whether the contents are still the same. The measure of dissimilarity most often used is the residual as defined in equation (4.6). This approach would be accurate if the movement in the image stream was purely translational. Unfortunately, the movement in an image stream often contains rotation and scaling. Rotation and scaling are affine deformations. If there is rotation and scaling present in an image stream then the displacement of a tracking window is not constant throughout the window. For this reason an *affine motion* field is a more accurate representation:

$$\delta = D\vec{x} + \vec{d}$$

where

$$D = \begin{bmatrix} d_{xx} & d_{xy} \\ d_{yx} & d_{yy} \end{bmatrix}$$

Stellenbosch University http://scholar.sun.ac.za

is a deformation matrix, and $\vec{d}$ is the translation of the feature window's centre and $\vec{x}$ is measured with respect to the window's centre. Thus a point $\vec{x}$ in the first image I moves to point $A\vec{x}+\vec{d}$ in the second image J, where $A=E+D$ and E is the 2×2 identity matrix:

$$J(A\vec{x}+\vec{d}) = I(\vec{x}). \tag{4.25}$$

Tracking now means that six parameters (the deformation matrix and the displacement vector) has to be determined. The quality of the estimate is dependant on the size of the tracking window; if the window is small, the matrix D is harder to estimate. The quality of the estimate is also dependant on the texture of the tracking window and the displacement between frames.

Because smaller tracking windows are preferable for tracking (as discussed in section 4.3), a pure translational model as described in the previous sections is preferable for tracking between sequential frames. The affine deformation model on the other hand can be used to compare the content of tracking windows over a large number of frames. It has been shown in Shi et al [5] that the pure translational model is more accurate for calculating inter-frame displacements.

Following is the definition of the affine deformation tracker.

As before the problem of determining the motion parameters is finding the A and $\vec{d}$ that minimises the dissimilarity

$$\varepsilon = \iint_W \left[J(A\vec{x}+\vec{d}) - I(\vec{x})\right]^2 \omega(\vec{x})d\vec{x} \tag{4.26}$$

where W is the given feature window and $\omega(\vec{x})$ is a weighting function. In the case of pure translational deformation A is an identity matrix. To minimise the residual in (4.26) it is differentiated with respect to the unknown entries of the deformation matrix D and the displacement vector $\vec{d}$ and the result equated to zero. The result is then linearised by use of a truncated Taylor expansion

$$J(A\vec{x}+\vec{d}) = J(\vec{x}) + g^T(\vec{u}). \tag{4.27}$$

This yields the following 6×6 system:

$$T\vec{z} = \vec{a} \tag{4.28}$$

Stellenbosch University http://scholar.sun.ac.za

where $\vec{z}^T = [d_{xx} \;\; d_{yx} \;\; d_{xy} \;\; d_{yy} \;\; d_x \;\; d_y]$ contains the entries of the deformation D and displacement $\vec{d}$.

The error vector

$$\vec{a} = \iint_W [I(\vec{x}) - J(\vec{x})] \begin{bmatrix} xg_x \\ xg_y \\ yg_x \\ yg_y \\ g_x \\ g_y \end{bmatrix} \omega d\vec{x}$$

depends on the difference between the two images, and the 6×6 matrix T, which can be computed from one image, can be written as

$$T = \iint_W \begin{bmatrix} U & V \\ V^T & Z \end{bmatrix} \omega \, d\vec{x}$$

where

$$U = \begin{bmatrix} x^2 g_x^2 & x^2 g_x g_y & xyg_x^2 & xyg_x g_y \\ x^2 g_x g_y & x^2 g_y^2 & xyg_x g_y & xyg_y^2 \\ xyg_x^2 & xyg_x g_y & y^2 g_x^2 & y^2 g_x g_y \\ xyg_x g_y & xyg_y^2 & y^2 g_x g_y & y^2 g_y^2 \end{bmatrix}$$

$$V^T = \begin{bmatrix} xg_x^2 & xg_x g_y & yg_x^2 & yg_x g_y \\ xg_x g_y & xg_y^2 & yg_x g_y & yg_y^2 \end{bmatrix}$$

$$Z = \begin{bmatrix} g_x^2 & g_g g_y \\ g_g g_y & g_y^2 \end{bmatrix}$$

Even when affine motion is a good model equation (4.28) is only approximately satisfied due to the linearisation of the system. The correct affine deformation can however be found by performing several iterations until the change in parameters between one step and the next is sufficiently small.

In this study features are tracked using the translational model as described in section 4.2 and 4.3. Then the features are monitored by solving for the affine deformation parameters

Stellenbosch University http://scholar.sun.ac.za

using a current image and comparing it to a base image (e.g. first or every 10th frame thereafter). Once the deformation parameters were obtained the features' residuals as defined in equation (4.26) are computed. The analysis of the residual values is discussed in the next section.

Since the displacement vector is known before the feature residuals are checked, the system in equation (4.28) can be changed slightly so that only four instead of six parameters have to be determined. Using $\omega(\vec{x}) = 1$, the system then looks as follows:

$$\iint_W \begin{bmatrix} U \\ V^T \end{bmatrix} d\vec{x} \times \begin{bmatrix} d_{xx} \\ d_{yx} \\ d_{xy} \\ d_{yy} \end{bmatrix} = \vec{a} - \iint_W \begin{bmatrix} V \\ Z \end{bmatrix} d\vec{x} \times \begin{bmatrix} d_x \\ d_y \end{bmatrix}. \tag{4.29}$$

The above system is an over constrained 6×4 system of equations, and its solution is calculated in a least squares sense.

4.6.1.1 Photometric normalisation

In order to make the implementation of the Lucas-Kanade Tracker as robust as possible, the calculation of the residuals incorporated a normalisation technique taken from Tomasini [6]. This normalisation limits the effect of intensity changes between frames by subtracting the average grey level in each of the two regions considered. Thus the residual calculation then becomes

$$\varepsilon = \sum_W [(J(A\vec{x} + \vec{d}) - \bar{J}) - (I(\vec{x}) - \bar{I})]^2 , \tag{4.30}$$

where $J(\cdot) = I(\cdot, t+1), I(\cdot) = I(\cdot, t)$. Note that $\bar{I}$ and $\bar{J}$ are the average grey levels in the two regions considered, but $\vec{x}$ and $\vec{d}$ are vectors.

4.6.2 Feature rejection using residuals

After the residuals have been calculated they need to be checked for outliers (abnormally high values). A high value indicates that the feature-tracking window in the current image does not resemble the corresponding window in the base image. This is due to non-affine

Stellenbosch University http://scholar.sun.ac.za

warping which can be caused by occlusions, perspective distortions and strong intensity changes. The problem is then to determine some threshold above which a feature's residual is declared too high and the feature rejected.

4.6.2.1 Distribution of residuals

According to Tomasini et al [6] the distribution of residuals are as follows.
When the residuals of good features are observed, it is assumed that the intensity $I(\delta(\vec{x}),t)$ of each pixel in the current frame's tracking window is equal to the intensity of the corresponding pixel (taking the affine transformation into account) in the first or base frame $I(\vec{x},0)$ plus some Gaussian noise $n \cong \eta(0,1)$. Hence

$$I(\delta(\vec{x}),t) - I(\vec{x},0) \cong \eta(0,1)\,.$$

Since the square of a Gaussian random variable has a chi-square distribution, the following expression is obtained

$$\left[I(\delta(\vec{x}),t) - I(\vec{x},0)\right]^2 \cong \chi^2(1)$$

The sum of n chi-square random variables with one degree of freedom is distributed as a chi-square with n degrees of freedom. Therefore the residual computed according to (4.29) over a $N \times N$ window W is distributed as a chi-square with N^2 degrees of freedom:

$$\varepsilon = \sum_W [I(\delta(\vec{x}),t) - I(\vec{x},0)]^2 \cong \chi^2(N^2)\,.$$

As the number of degrees of freedom increases the chi-square distribution approaches the Gaussian distribution. The Gaussian is often used to approximate a chi-square with more than 30 degrees of freedom. Thus if the tracking window W is at least 7×7 it can safely be assumed that the residual is distributed in a Gaussian fashion:

$$\varepsilon \cong \eta(N^2, 2N^2)\,.$$

4.6.2.2 The X84 rejection rule

If a feature has been occluded or has gone bad (warped by a non-affine transformation), the two regions over which the residual is computed will differ by more than mere Gaussian noise. The residual is then an outlier and should be rejected. Thus the detection

Stellenbosch University http://scholar.sun.ac.za

and rejection of bad features becomes an issue of determining the mean and variance of a Gaussian distribution from a dataset that contains possible outliers.
To do this Tommasini et al [6] employed a rejection rule called the X84 rejection rule taken from Hampel [7].
The X84 rejection rule achieves robustness by the use of the median (estimated mean) and the median deviation. This rule prescribes that features be rejected when their residuals are more than *k* Median Absolute Deviations (MADs) away from the median:

$$MAD = \underset{i}{med}\left\{\left|\varepsilon_i - \underset{j}{med}\,\varepsilon_j\right|\right\}$$

In this case, ε_i is the tracking residual value between the i^{th} feature in the last frame, and the same feature in the base frame. A value of k=5.2, under the hypothesis of Gaussian distribution is adequate in practice since it corresponds to about 3.5 standard deviations containing more than 99.9% of a Gaussian distribution, [7].
Note that the X84 breaks down if there are too many outliers, in which case a larger median residual value is assumed and outliers are accepted that should have been rejected.

4.7 Conclusions

In this chapter the Lucas-Kanade Feature tracker and some variations on its application were discussed. The effects of different tracking window sizes and pyramid heights have been discussed. It is now possible to select, track and monitor features, as well as reject bad features. From here on it is necessary to obtain the 3D coordinates to group these features in order to deduce meaning from the feature tracks. In the following section the clustering or grouping of features are discussed in detail.

Stellenbosch University http://scholar.sun.ac.za

Chapter 5

Feature clustering

According to Anderberg [8] "*Cluster analysis encompasses many diverse techniques for discovering structure within complex bodies of data*".
Following the previous section on Feature Tracking it now becomes necessary to derive meaning from the feature trajectories as obtained by the tracking algorithm. In this section the clustering of the points into meaningful entities is discussed.

5.1 Similarity value computation

Before clustering methods can be used it is necessary to define some type of association between the entities that are to be clustered.
In this study, the only attribute of the features that is available on the commencement of the clustering procedure is the position in 2-dimensional space (as tracked on the images) at a series of time instants. These have to be used to determine a measure of association between the features. The two dimensional points can either be used in their raw form or they can be transformed (using for example the geometric data of the environment) using a perspective transformation. An example of this is using the geometric data of a freeway and projecting the points in the image onto the freeway to obtain real world coordinates. Perspective transformations are discussed in the next section.

In this study the similarity value is obtained by calculating the distances between features over a number of frames. The standard deviation of these distances is used as the similarity value. Thus if $\vec{x}_i = \begin{bmatrix} x_{i,1} & x_{i,2} & \cdots & x_{i,m} \\ y_{i,1} & y_{i,2} & \cdots & y_{i,m} \end{bmatrix}$ denotes the coordinates for the i^{th} of n features for the last m frames then the distance between feature i and j at frame k of m frames is $\gamma_{i,j,k} = \sqrt{(x_{i,k} - x_{j,k})^2 + (y_{i,k} - y_{j,k})^2}$. This definition can also easily be adapted for the 3-dimensional case.

Stellenbosch University http://scholar.sun.ac.za

The similarity matrix entries are then calculated using the sample standard deviation equation:

$$s_{i,j} = \sqrt{\frac{\sum_{k=1}^{m} \left(\gamma_{i,j,k} - \bar{\gamma}_{i,j}\right)^2}{(n-1)}} . \tag{5.1}$$

It is important to note that $s_{i,j} = s_{j,i}$ and for that reason the similarity matrix only needs to be an upper or lower triangular matrix.

The reason for using standard deviation as a similarity value is the following:

- A small value corresponds to features that remain a fixed distance from each other. It is thus expected that a small similarity value will indicate that the features belong to the same cluster or object (the same vehicle in this case).
- The magnitude of the standard deviation is dependent on the size of the distance between points. This will cause points that are far from each other to have larger similarity values. This makes sense since it is less likely that points that are far removed from each other belong to the same object.
- Another derived measure, the angle between features, could have been used for the clustering process. This will however not work if the movement of the object is of such a nature that it can cause the angle to change and has for this reason not been used as a measure for clustering.

5.2 Clustering algorithms

Broadly speaking there are two types of clustering algorithms, these being hierarchical and non-hierarchical clustering methods.

Hierarchical methods use a similarity matrix to construct a tree depicting the relationships among entities. This is shown in Figure 5.1 below.

Stellenbosch University http://scholar.sun.ac.za

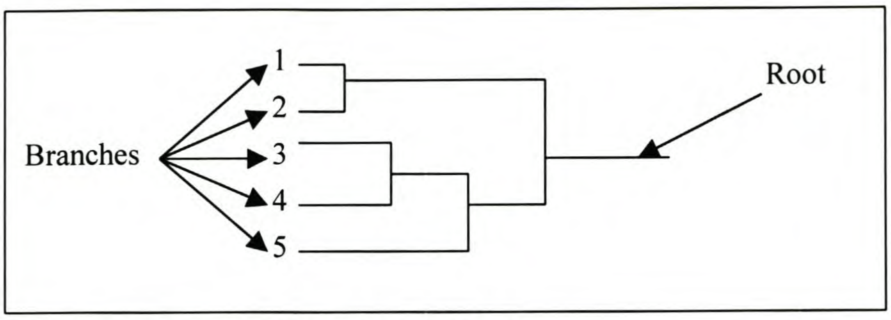


Figure 5.1: Tree for hierarchical clustering

The branches on the left each represent one entity (or in this study a feature). Moving from the branches toward the root, the clusters with the largest similarity are grouped together. These are called agglomerative methods because the clusters are increasingly aggregated. There are also divisive methods that begin at the root and then divide into the branches, but these are less common. Once a tree has been constructed from say *n* entities the analyst may choose from as many as *n* sets of clusters. The set of clusters for Figure 5.1 is shown below in Table 5.1.

Table 5.1: Clusters for each step of the hierarchical clustering example

Number of clusters	Clusters
5	(1),(2),(3),(4),(5)
4	(1,2),(3),(4),(5)
3	(1,2),(3,4),(5)
2	(1,2),(3,4,5)
1	(1,2,3,4,5)

It is important to note that the clusters are nested. Once two entities are merged they are joined permanently and become a building block for later merges. The same is true for the splitting of features in the divisive methods. The last statement contains both the strengths and weaknesses of the hierarchical methods. By grouping entities or clusters permanently, the number of possible cluster groupings that remains are reduced greatly compared to investigating all the possible combinations. The weakness is however, that if

Stellenbosch University http://scholar.sun.ac.za

an incorrect grouping is made early on in the clustering process, it cannot be rectified and the clustering is built upon this faulty foundation.
There are several hierarchical clustering methods, including the Ward method that was used in this study.

The central idea in non-hierarchical clustering methods is to choose an initial partition of the data units and then alter the cluster membership in order to obtain a better clustering. The concept of these methods are often close to the steepest descent algorithms as used for unconstrained optimisation in non-linear programming.
The disadvantage of these methods are that the number of clusters have to be specified beforehand or user specified threshold values have to be introduced in order to have the method choose the number of clusters such as MacQueen's *k*-means method with coarsening and refining parameters, from Anderberg [8].

In the case of traffic surveillance systems the number of clusters that can be expected can vary quite considerably. Furthermore, the computational complexity of the clustering method should be as small as possible. For the above reasons a self-regulating hierarchical method, the Ward method, was chosen for use in this study.

5.2.1 The Ward clustering method

Ward described a very general hierarchical clustering method in 1963, in which the merges at each stage are chosen so as to optimise an objective function. Ward however illustrated his method with an error function (sum of square errors) and the example became better known than the general procedure.
The method (following Andenberg [8]) is described in detail below.

The following quantities are defined:
x_{ijk} : score on the *i*th of *n* variables for the *j*th of m_k data units in the *k*th of *h* clusters,

Stellenbosch University http://scholar.sun.ac.za

$\bar{x}_{ik} = \sum_{j=1}^{j=m_k} x_{ijk} / m_k$: mean on the *i*th variable for data units in the *k*th cluster.

$$E_k = \sum_{i=1}^{i=n} \sum_{j=1}^{j=m_k} (x_{ijk} - \bar{x}_{ik})^2 = \sum_{i=1}^{i=n} \sum_{j=1}^{j=m_k} x_{ijk}^2 - m_k \sum_{i=1}^{i=n} \bar{x}_{ik}^2$$

:error sum of squares for cluster *k*; sum of Euclidian distances from each data point in cluster *k* to the mean vector of cluster *k*; within group squared deviations about the mean for cluster *k*,

$E = \sum_{k=1}^{k=h} E_k$: total within group error sum of squares for the collection of clusters.

At the start there are m data units, each its own cluster; thus the membership and the mean of each cluster coincide so that $E_k = 0$ for all clusters. The Ward objective is to find at each stage the two clusters whose merger will result in the minimum increase in the total within error *E*. Suppose clusters *p* and *q* are chosen to be merged and the resulting cluster is denoted as *t*. The increase in *E* is then:

$$\Delta E_{pq} = E_t - E_p - E_q$$

$$= \left[\sum_{i=1}^{i=n} \sum_{j=1}^{j=m_t} x_{ijt}^2 - m_t \sum_{i=1}^{i=n} \bar{x}_{it}^2 \right] - \left[\sum_{i=1}^{i=n} \sum_{j=1}^{j=m_p} x_{ijp}^2 - m_p \sum_{i=1}^{i=n} \bar{x}_{ip}^2 \right] - \left[\sum_{i=1}^{i=n} \sum_{j=1}^{j=m_q} x_{ijq}^2 - m_q \sum_{i=1}^{i=n} \bar{x}_{iq}^2 \right]$$

$$= m_p \sum_{i=1}^{i=n} \bar{x}_{ip}^2 + m_q \sum_{i=1}^{i=n} \bar{x}_{iq}^2 - m_t \sum_{i=1}^{i=n} \bar{x}_{it}^2 \qquad (5.2)$$

The mean on the *i*th variable for the new cluster is calculated using the relation $m_t \bar{x}_{it} = m_p \bar{x}_{ip} + m_q \bar{x}_{iq}$; squaring both sides of the equation results in:

$$m_t^2 \bar{x}_{it}^2 = m_p^2 \bar{x}_{ip}^2 + m_t^2 \bar{x}_{ip}^2 + 2 m_p m_q \bar{x}_{ip} \bar{x}_{iq} .$$

The product of the means can be rewritten as

$$2 \bar{x}_{ip} \bar{x}_{iq} = \bar{x}_{ip}^2 + \bar{x}_{iq}^2 - (\bar{x}_{ip} - \bar{x}_{iq})^2 ,$$

so that

$$m_t^2 \bar{x}_{it}^2 = m_p (m_p + m_q) \bar{x}_{ip}^2 + m_q (m_p + m_q) \bar{x}_{iq}^2 - m_p m_q (\bar{x}_{ip} - \bar{x}_{iq})^2 .$$

Noting that $m_t = m_p + m_q$ and dividing both sides of the last equation by m_t^2,

Stellenbosch University http://scholar.sun.ac.za

$$\bar{x}_{it} = \frac{m_p}{m_t}\bar{x}_{ip}^2 + \frac{m_q}{m_t}\bar{x}_{iq}^2 - \frac{m_p m_q}{m_t^2}(\bar{x}_{ip} - \bar{x}_{iq})^2 . \tag{5.3}$$

Substituting equation (5.3) into equation (5.2) then gives

$$\Delta E_{pq} = \frac{m_p m_q}{m_p + m_q}\sum_{i=1}^{i=n}(\bar{x}_{ip} - \bar{x}_{iq})^2 . \tag{5.4}$$

Thus the minimum increase in the error sum of squares is proportional to the squared Euclidian distance between the centroids of the merged clusters. This method then also involves the weighting of the distance between centroids when computing the distances. The error, *E*, is non-decreasing implying that the method does not allow a grouping to be reversed.

From the above derivation the increase in the value of *E* is now known. When using the stored similarity matrix approach, an update equation that calculates the new similarity values between the newly merged cluster and the rest of the clusters is necessary.

Let t denote the result of merging clusters p and q, and let r denote any other cluster. The increase in *E* that would result from the potential merger of clusters r and t is obtained from equation (5.4) as

$$\Delta E_{rt} = \frac{m_r m_t}{m_r + m_t}\sum_{i=1}^{i=n}(\bar{x}_{ir} - \bar{x}_{it})^2 . \tag{5.5}$$

By substituting $\bar{x}_{it} = (m_p\bar{x}_{ip} + m_q\bar{x}_{iq})/m_t$, $m_t = m_p + m_q$, and using equation (5.3), the squared terms of equation (5.5) may be written as

$$(\bar{x}_{ir} - \bar{x}_{it})^2 = \frac{m_p}{m_t}(\bar{x}_{ir} - \bar{x}_{ip})^2 + \frac{m_q}{m_t}(\bar{x}_{ir} - \bar{x}_{iq})^2 - \frac{m_p m_q}{m_t^2}(\bar{x}_{ip} - \bar{x}_{iq})^2 .$$

Substituting this into equation (5.5) and gathering terms

$$\Delta E_{rt} = \frac{1}{m_r + m_t}\sum_{i=1}^{i=n}\left[m_r m_p(\bar{x}_{ir} - \bar{x}_{ip})^2 + m_r m_q(\bar{x}_{ir} - \bar{x}_{iq})^2 - \frac{m_r m_p m_q}{m_p + m_q}(\bar{x}_{ip} - \bar{x}_{iq})^2\right].$$

Using equation (5.4) again yields

$$\Delta E_{rt} = \frac{1}{m_r + m_t}\left[(m_r + m_p)\Delta E_{rp} + (m_r + m_q)\Delta E_{rq} - m_r\Delta E_{pq}\right]. \tag{5.6}$$

Stellenbosch University http://scholar.sun.ac.za

Equation (5.4) shows that the initial entries in the similarity matrix are $E_{ij} = \frac{1}{2}d_{ij}^2$, half the Euclidian distance between data units *i* and *j*. It is not necessary to divide the entries in the initial matrix by 2; the updates of equation (5.6) may be carried out using the squared distances and when the increment to the error sum of squares function is added, use half the computed value.

The algorithm now involves the following steps:

1. Store the lower triangular matrix of squared distances. Treat s_{ij} as E_{ij}.
2. At each stage merge the two clusters which as a pair give the smallest increment to the error sum of squares. Let these clusters be labelled p and q with the resulting cluster t.
3. Update entries in the similarity matrix according to equation (5.6). Increment the error using
$$E = E + \frac{1}{2}\Delta E_{pq}\,.$$
4. Repeat steps 2 and 3 for m-1 stages where m is the number of data units.

5.2.2 Remarks on the implementation of the Ward clustering algorithm

As described in section 5.1, the similarity values were calculated as the standard deviation of the distance between pairs of points for a number of time instants in the past. In section 5.2.1 the references to the Euclidian distance refers to the similarity measure and not to the actual distance between feature points.

The clustering algorithm has now been described in detail. The next issue is how many clusters should there be before the grouping process is stopped.

If the error is plotted it will exhibit a trend like Figure 5.2.

Stellenbosch University http://scholar.sun.ac.za

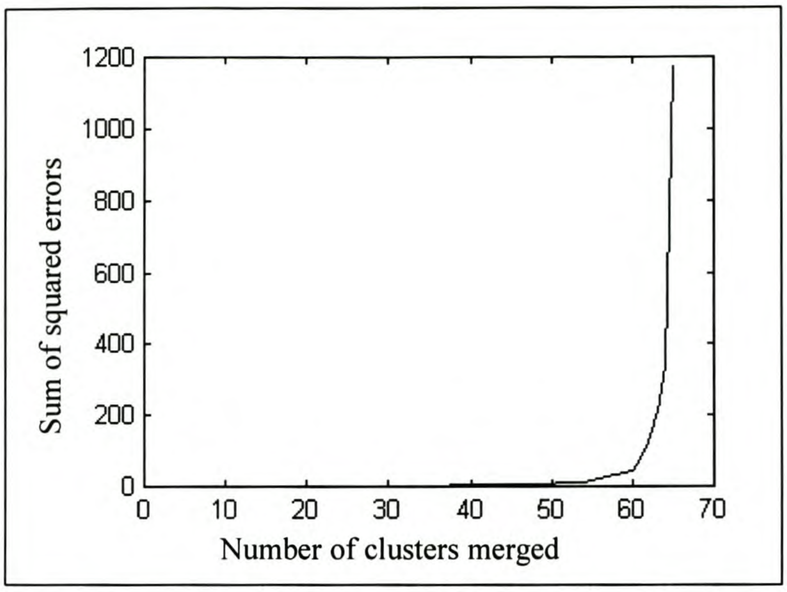


Figure 5.2: A typical plot of an error for complete clustering

The slope of the function will increase as the clustering continues. This is expected from the clustering method since the groupings that result in the minimum increment of the error are performed first.

From the graph it can also be seen that the error remains relatively low initially. Then at some point the function value starts increasing substantially. This indicates that features that do not belong to the same object are being grouped. Once all the features have been grouped into one cluster the error should be equal to the total error sum of squares.

The clustering is stopped at the point where the slope between successive values of the error sum of squares function exceeds the average slope of the curve over its full course (until only one cluster remains).

To do this the error is subtracted from a straight line from the origin of the curve to the maximum point of the full error sum of squares plot. This graph is shown in Figure 5.3. The minimum point of this plot is selected as the point at which clustering should be stopped.

Stellenbosch University http://scholar.sun.ac.za

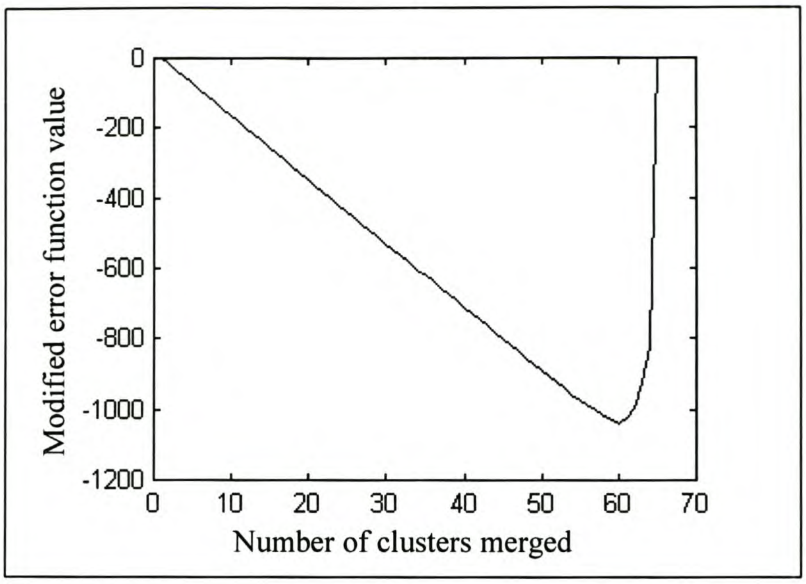


Figure 5.3: Plot of a typical altered error sum of squares function

From Figure 5.3 it can be seen that there are 65 features available, and in this case the clustering procedure is stopped after 60 groupings, leaving 5 clusters. Therefore the traffic surveillance system has identified 5 vehicles.

5.3 Conclusions

The Ward clustering method, being a hierarchical clustering method, does not guarantee an optimal solution, however Andenberg [8] stated "*the Ward solution is usually very good even if it is not optimal on this criterion*".

A clustering method such as the *k*-means method can refine the result of a hierarchical clustering method; however if the tracking and perspective transformations are accurate the results from the Ward clustering method should be accurate and refining should not be necessary.

One problem has, however, been identified with the approach as described in this chapter; if only one vehicle is present the clustering algorithm will still attempt to group the features belonging to the vehicle into more than one cluster (relating this scenario to Figure 5.3, the minimum point on the altered error sum of squares function will always

Stellenbosch University http://scholar.sun.ac.za

be reached before the final grouping is made simply because there is slight variation in the calculated distances between features even when they belong to the same vehicle). One solution to the problem is the introduction of a threshold below which the minimum point on the modified error of the clustering algorithm (as shown in Figure 5.3) will not be used to stop the clustering. This will ensure that it is possible to group features into a single group. The problem is that the feature positions are too variable and until a more accurate perspective transformation is obtained (these problems are discussed in section 6.4) it would not be worthwhile to search for an appropriate threshold.

Thus the problems encountered with the clustering algorithm can be attributed to problems with perspective distortions and transformations as described in the following chapter.

Stellenbosch University http://scholar.sun.ac.za

Chapter 6

Perspective transformation

6.1 Objective

When features are tracked through a sequence of images the coordinates are located on a two-dimensional plane and measured in units of pixels. This plane represents a 3-dimensional real world. The coordinates as obtained by the feature tracker are of little use if it cannot be converted into coordinates that represent the real world. The perspective distortions found in images (the illusion that the further an objects is away from the vantage point, the smaller it becomes) are especially problematic. Due to the above reasons perspective transformations are essential. In this chapter, perspective transformations will be discussed.

6.2 Assumptions

When a scene is filmed with a camera, a line of points is projected onto one point on the image, the so-called projection line. This implies that only the closest point on the line can be seen. In order to restore the 3D coordinates from 2D projections some assumptions or additional information is required. A common assumption is that the points projected onto the image plane come from one or more planes in 3D space, thus if the parameters defining the plane in 3D space is known, the real world coordinates can be found by projecting the points back from the image onto the real world plane.
For the above reasons, the section of road that is surveyed should be as flat as possible so that the assumption of a plane in 3D space is not violated.
The inaccuracies associated with such assumptions can be overcome by using a stereo setup. In a stereo setup two cameras are used and subsequently the depth of the point

Stellenbosch University http://scholar.sun.ac.za

(distance from camera)can be calculated. A stereo setup was however not attempted in this study.

6.3 The transformation

There are various transformations available. One of these is described in section 2.5.2 of Gonzalez/Woods [10]. This transformation includes the calibration of a full camera model. Even though calibration is relatively straightforward, the transformation has the disadvantage that it only handles the projection of points from 3D space onto the image plane and not the opposite (projection of points from the image back into 3D space). Another transformation discussed in Russ [9] stretches the image so that the observed trapezoidal distortion obtained by photographing a tilted surface, is rectified. This method however does not incorporate enough parameters and is only an approximate method. The coordinates obtained are therefore not sufficiently accurate.

For the purpose of the project, the perspective transformation used was custom defined in order to facilitate the calibration of the model.
Figure 6.1 below shows how the model looks in 3-D space

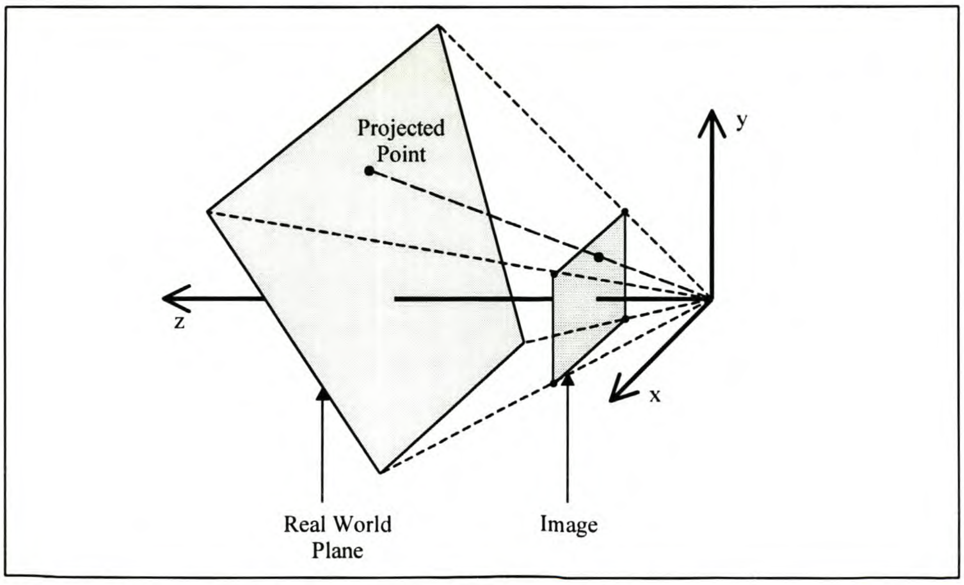


Figure 6.1: The perspective transformation model used

Stellenbosch University http://scholar.sun.ac.za

Figure 6.1 depicts two planes that are set in a Cartesian coordinate system. The smaller shaded plane is the image plane, and the larger one is part of the "real world" plane that is filmed by the camera. The real world plane is drawn skew to show that it is not, and does not have to be, parallel to the image plane.
In a camera the lens would be located at the origin of this Cartesian coordinate system with an inverted image forming on the image plane behind the origin (negative z). In this case the non-inverted image plane was placed on the same side of the origin as the real world plane. In order to find the real world coordinates of a point on the image plane, a line has to be drawn from the origin through the point on the image plane and the intersection of this line with the real world plane would provide the real world coordinates.

The broken lines in Figure 6.1 shows the projection of the corners of the image plane onto the real world plane, as well as one projected point.

The perspective transformation that was used can now be defined as follows:

The general equations for a line and plane in 3D are respectively

$$\vec{r} = \vec{r}_0 + \lambda\vec{a} \tag{6.1}$$

and

$$\vec{N} \cdot (\vec{r} - \vec{r}_1) = 0 \, . \tag{6.2}$$

Here $\vec{r}_0$ denotes a reference point through which the line passes, λ is a variable, and $\vec{a}$ denotes the direction of the line. For the plane, $\vec{N}$ denotes the normal vector, $\vec{r}$ is a variable and $\vec{r}_1$ is any point in the plane.
By substituting (6.1) into (6.2) the intersection of the line with the plane is obtained from

$$\vec{N} \cdot (\vec{r}_0 - \vec{r}_1) + \lambda\vec{N} \cdot \vec{a} = 0 \text{, only if } \vec{N} \cdot \vec{a} \neq 0 \, .$$

Note that $\vec{N} \cdot \vec{a} = 0$ denotes a line parallel to the plane in which case no intersection is possible (or an infinite number of intersections if the line is in the plane).
Thus

Stellenbosch University http://scholar.sun.ac.za

$$\lambda = \frac{\vec{N}\cdot(\vec{r}_1 - \vec{r}_0)}{\vec{N}\cdot\vec{a}}.$$

The point of intersection is then

$$\vec{r} = \vec{r}_0 + \left(\frac{\vec{N}\cdot(\vec{r}_1 - \vec{r}_0)}{\vec{N}\cdot\vec{a}}\right)\vec{a}. \qquad (6.3)$$

But in this specific case $\vec{r}_0 = 0$ because the line passes through the origin and (6.3) becomes

$$\vec{r} = \left(\frac{\vec{N}\cdot(\vec{r}_1)}{\vec{N}\cdot\vec{a}}\right)\vec{a} .$$

The following step is to determine the parameters that define the image plane and the real world plane. It is important to note that the image plane coordinates are in pixels and thus have no relation to a physical distance. In this study the image centre was defined as (0,0) and the coordinates were expressed in terms of pixels. Since pixels do not have a definite size, they had to be related to a physical measure. Therefore pixels were related to meters in a one to one ratio.

The parameters that need to be obtained are:

- The distance from the origin to the image plane along the z axis.
- The real world plane normal vector $\vec{N}$
- A point on the real world plane $\vec{r}_1$.

The above parameters can be obtained by using three points on the real world plane. The distance from the camera lens to each of the points, the distance between two of these points, and the projection coordinates of these points on the image plane are needed.

The following derivation serves to explain how the distance from the origin to the image plane was calculated.

Let $\vec{v} = (v_x; v_y; d)$ and $\vec{w} = (w_x; w_y; d)$ be two vectors to the image plane. Here d is the camera's focal length, it is important to note that since the size of the camera's actual image plane is not known d does not have to be equal to the actual focal length, but will be calculated later on (as mentioned before the size of the camera's image plane is set to

Stellenbosch University http://scholar.sun.ac.za

240m×320m, thus a pixel is 1m×1m). The vectors $\vec{v}$ and $\vec{w}$ are the projection points of the two real world points for which the distance between them is known. Note that v_x and w_x are the vertical pixel coordinates on the image, and v_y and w_y the horizontal coordinates with the image centre at *(0;0)*. In this study the *y* direction is chosen positive towards the right and *x* positive downwards.

Then $\frac{\vec{v}}{\|\vec{v}\|}$ is a unit vector in the direction of $\vec{v}$, and $\frac{\vec{v}}{\|\vec{v}\|}l_v$ is the point on the real world plane where l_v is the physical distance from the camera to the point.

In the same way $\frac{\vec{w}}{\|\vec{w}\|}l_w$ is the location of the point $\vec{w}$ on the real world plane.

If *b* denotes the distance between the two points in the real world, then:

$$\left\| \frac{\vec{v}}{\|\vec{v}\|}l_v - \frac{\vec{w}}{\|\vec{w}\|}l_w \right\| = b$$

which can be expanded into

$$\left\| \frac{(v_x;v_y;d)}{\sqrt{v_x^2+v_y^2+d^2}}l_v - \frac{(w_x;w_y;d)}{\sqrt{w_x^2+w_y^2+d^2}}l_w \right\| = b . \tag{6.4}$$

The only unknown in equation (6.4) is *d*, which is the distance from the origin to the image plane along the z-axis.
Equation (6.4) can be used to solve for *d*.
In this study a numerical approximation method (as provided in the MATLAB package) resembling the Newton-Raphson technique was used to obtain an estimate of *d*.

All that remains in order to calibrate the model is a normal vector on the real world plane. Now that *d* is known the real world coordinates of $\vec{v}$ and $\vec{w}$ are known. Let $\vec{t}$ be a third vector to a point on the real world plane obtained in the same way as the other two points. The normal vector on the real world plane is then:

$$\vec{N} = (\vec{v}-\vec{t})\times(\vec{w}-\vec{t}). \tag{6.5}$$

Stellenbosch University http://scholar.sun.ac.za

6.4 Inaccuracies

The problem with the above model is that the points are assumed to be located within one plane, which is not true. The points can be as low as the surface of the road surface, and as high as the roof of a truck or bus. This causes an error in the calculation of the point in 3-D space. The magnitude of the error is dependent on the location of the feature and height difference to the hypothetical plane.

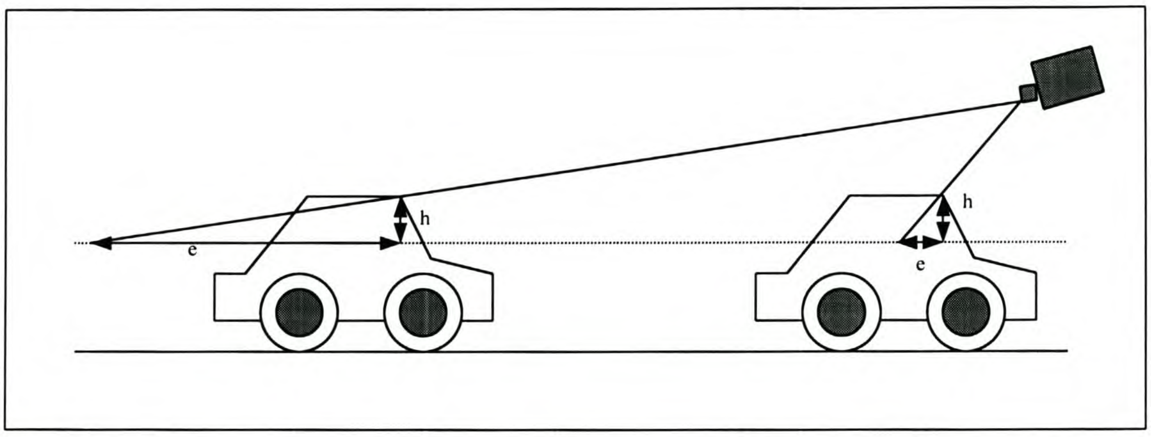


Figure 6.2: Perspective transformation errors

When the model calibration is done the assumed plane should be chosen at the height of an expected average of feature heights. This is done in order to minimise the perspective transformation error due to the difference in height between the feature and the plane that the feature is assumed to be in. Figure 6.2 illustrates these errors. In the figure *h* denotes the difference between the actual feature height and the assumed feature height, and *e* denotes the error that is made in calculating the real world coordinates. Another problem is that these errors are a function of the angle that the line of sight makes with the plane and is thus not a constant error. This is also illustrated in the difference between the errors of the two identical features on identical vehicles at different positions with respect to the camera in Figure 6.2.

Stellenbosch University http://scholar.sun.ac.za

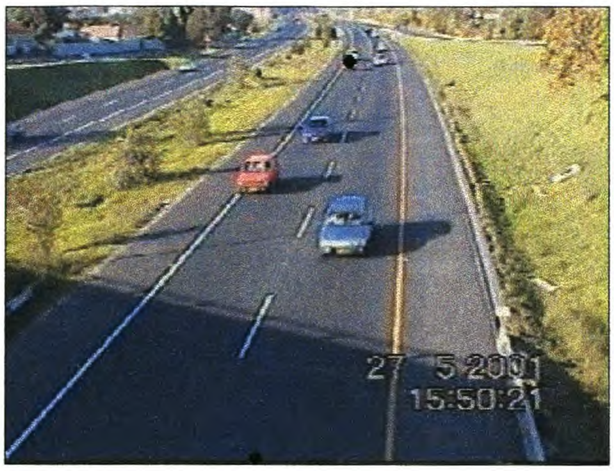


Figure 6.3: Image of section showing two places where error calculations were done

To illustrate the magnitude of these errors, they were calculated at two different positions in the image of Figure 6.3. The two black spots in the figure (one towards the top of the image and one at the bottom close to the centreline of the road) shows the positions used.

In the case of the lower/nearer of the two locations, the error calculated was approximately 2m for every 1m-height difference. For the higher/further of the two locations the error increased substantially to approximately 18m for every 1m-height difference.

The problem with the image in Figure 6.3 is the large deformation in size of the vehicles from one end of the section to the other. In order to reduce this problem the same section of road was filmed using a longer focal length lens. The resulting image is shown in Figure 6.4.

Stellenbosch University http://scholar.sun.ac.za

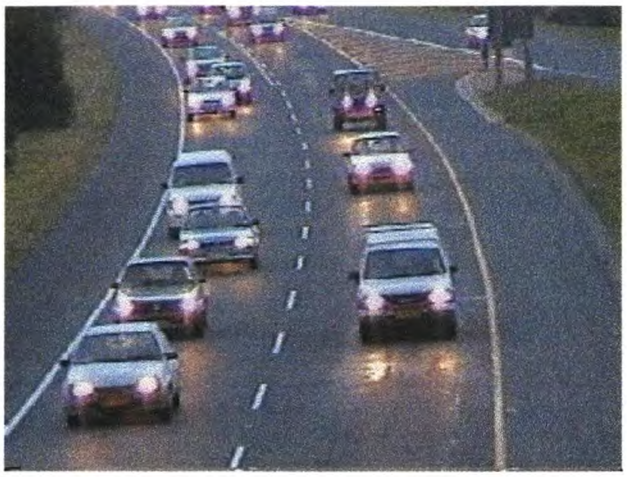


Figure 6.4: Using a longer focal length lens to reduce perspective distortions

However the use of an image such as Figure 6.4 increases the error from the perspective transformation by decreasing the angle that the camera makes with the plane. The advantage is that the perspective distortion is decreased significantly (vehicles sizes do not change by as much as the previous case). If the perspective distortion is small enough it might not be necessary to perform a perspective transformation on features before they are grouped. The grouping of features can then be done using the image coordinates. This however is an approximation and the advantages and disadvantages of both the options (performing the perspective transformation or not) should be considered for different camera locations and focal lengths.

6.5 Speed calculation examples

Speed calculations were performed on feature points that were tracked on the image sequence that Figure 6.3 was taken from. These speeds were calculated by first obtaining the 3-D coordinates of the features using the perspective transformation and then calculating their speeds.
Due to the small inter-frame displacements and the variability in the tracking accuracy, speed measurements taken between two sequential frames are highly variable. Figure 6.4 below shows the speeds for single features through a sequence of 30 frames. As can be seen the speeds are too variable to be of use. If however the speeds are calculated over a

Stellenbosch University http://scholar.sun.ac.za

longer time period such as over 5 frames (as shown in Figure 6.5) it can be seen that the speeds remain relatively constant.

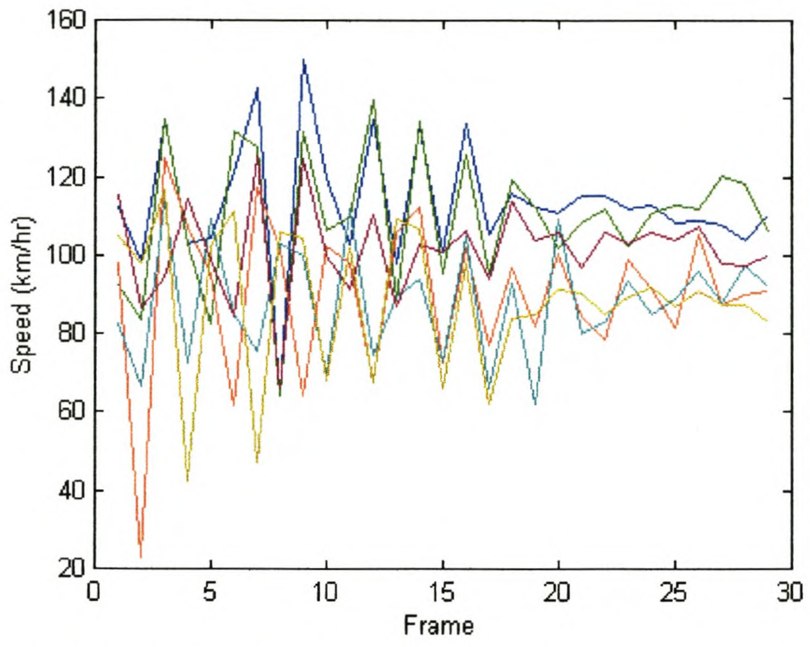


Figure 6.4: Real inter-frame speeds

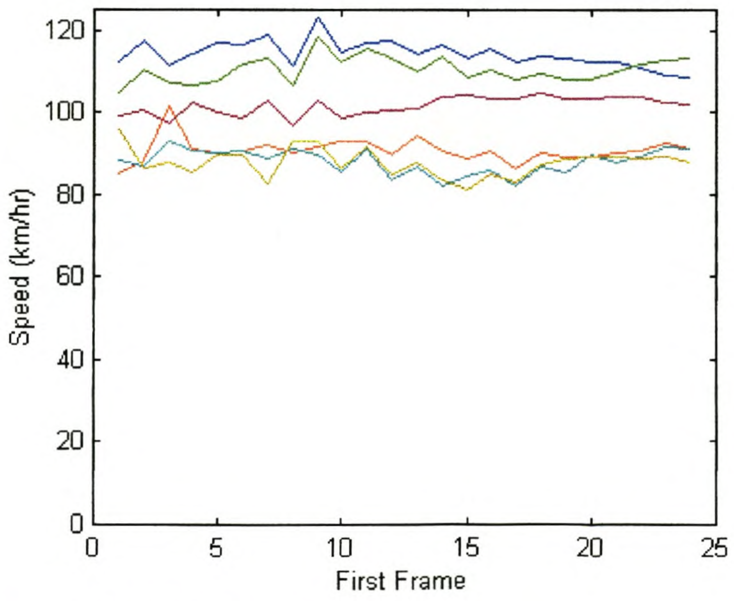


Figure 6.5: Real 5 frame average speeds

Stellenbosch University http://scholar.sun.ac.za

Figure 6.5 seems to represent reality since the speed limit at the filmed section of road is 120 km/h, and acceleration is expected to be small.

The features' speeds or optical flow speeds in the frame sequence do however not remain constant (as shown below in Figure 6.6), the features' speeds on the image increase as they approach the camera, and vice versa.

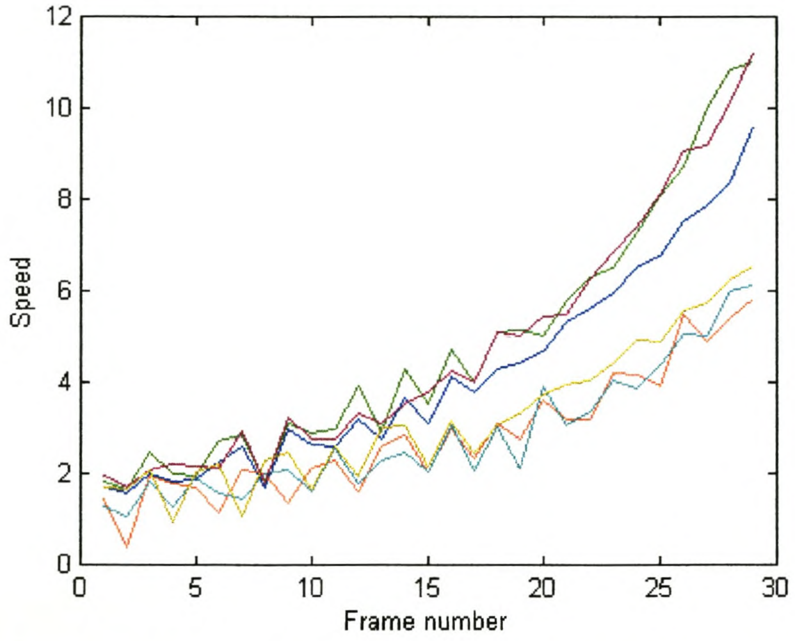


Figure 6.6: Feature speeds in pixels per frame (optical flow speeds)

Stellenbosch University http://scholar.sun.ac.za

Chapter 7

Implementation of algorithms

7.1 Program structure

The algorithms were implemented using MATLAB. The structure of the program as well as the individual procedures are discussed in this chapter

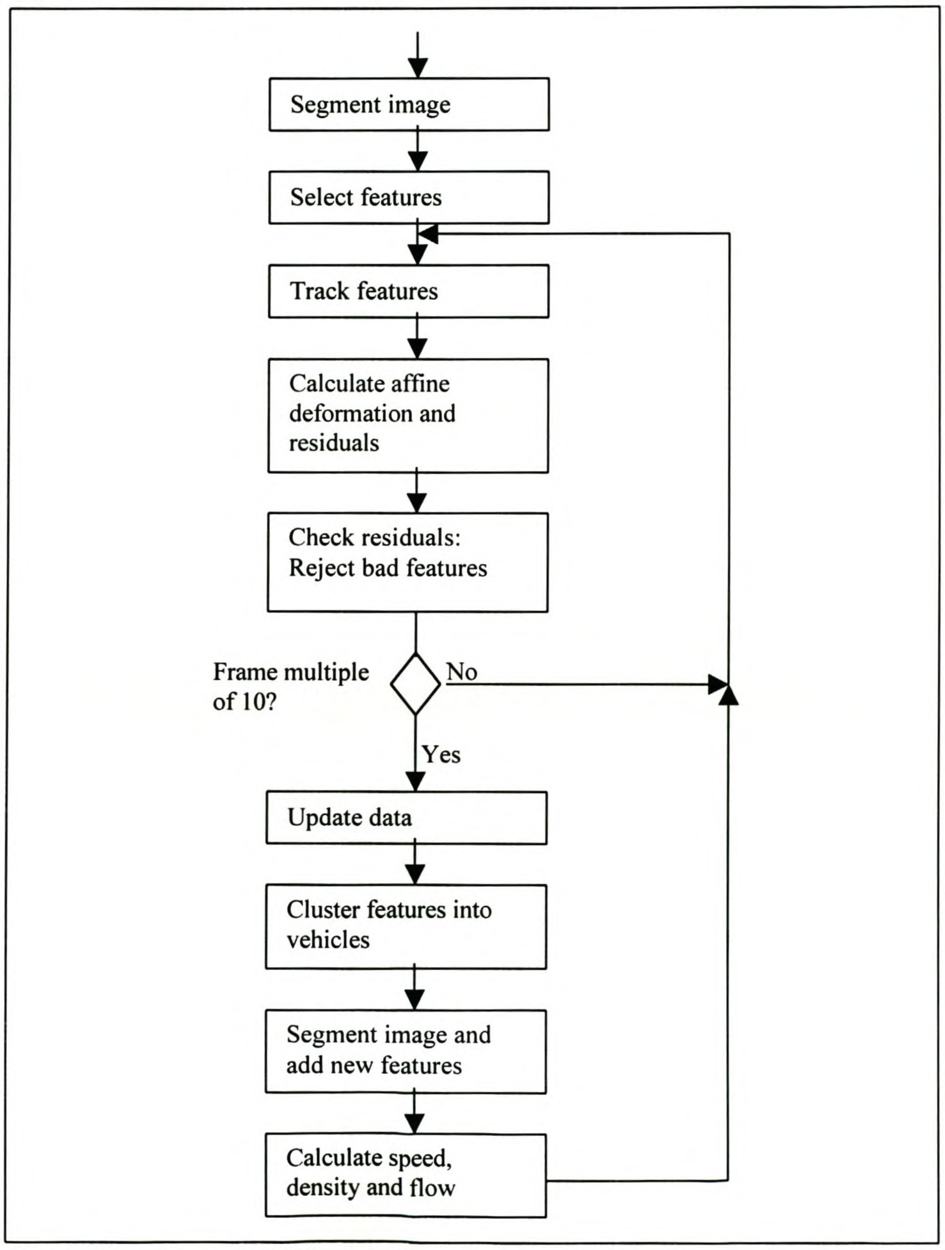

Segment image
Select features
Track features
Calculate affine deformation and residuals
Check residuals: Reject bad features
Frame multiple of 10?
No
Yes
Update data
Cluster features into vehicles
Segment image and add new features
Calculate speed, density and flow

Figure 7.1: Outline of the structure of the MATLAB code

Stellenbosch University http://scholar.sun.ac.za

Segment image

Running the "long_sequence_track.m" file starts the vehicle-tracking program.
The initial steps consist of defining some of the variables that will be used.
The background computation and the travelled way definition image has to be performed prior to the launch of the "*long_sequence_track.m*" file. In a proper implementation the background computation should be performed at selected intervals so that the background adapts to the changing conditions. For this study it was not necessary since the sequences tested were short enough so that the change in conditions were negligible.

The first image is read and the segmentation process performed on it. The segmentation process consists of comparing the first image to the background image, filtering it with a median filter, and comparing it to the travelled way definition image. These steps are described in more detail in chapter 3.

Select features

The features are identified by the function "*f_im = getfeat(im,threshold,w)*", where *im* is the input image. The variable *threshold* is the percentage of the maximum of the smaller of the two eigenvalues for the image, that has to be exceeded by the smaller of the two eigenvalues of a feature window in order for the feature to be selected. The variable w sets the size of the feature selection window as $(2w+1)\times(2w+1)$. This is described in section 4.4.
The output of the function is a black and white image (or matrix with entries of 0, except for 1's where a feature has been identified. This image is then compared to the segmented image (using AND function) so that all features that do not lie on the segmented objects are discarded.

The last step is to enter the selected features and their coordinates into matrices in order to keep track of them. These matrices are *feature_state* and *features*, and look as follows:

Stellenbosch University http://scholar.sun.ac.za

Table 7.1: Structure of the matrix "*feature_state*"

	State	Vehicle number	Nr of frames tracked
Feature 1			
Feature 2			
Feature 3			
…			
Feature *n*			

Table 7.2: Structure of the matrix "*features*"

	Frame 1	Frame 2	…	Frame *m*
Feature 1	x, y, residual	x, y, residual	…	x, y, residual
Feature 2	x, y, residual	x, y, residual	…	x, y, residual
Feature 3	x, y, residual	x, y, residual	…	x, y, residual
…	…	…	…	…
Feature *n*	x, y, residual	x, y, residual	…	x, y, residual

The first column of the "*feature_state*" matrix holds the state of the feature. The states include:

0: no feature

1: feature is being tracked

2: the feature has been lost due to tracking too close to the image boundary

3: the feature has been rejected due to a high residual.

The second column contains the cluster (vehicle) number that is assigned to each feature after the clustering has been performed.

The third column is the number of frames through which the feature has been tracked (age of the feature).

Stellenbosch University http://scholar.sun.ac.za

The "features" matrix is in fact a three-dimensional data structure. The rows denote the various features and the columns the various frames. The first level of the data block contains the image-row (or x) coordinate values; the second level contains the image-column (or y) coordinate values; and the third level contains the residual values for each of the features in each of the frames.

Track features

This step consists of tracking the features as identified in the previous step. A pyramidal implementation of the Lucas-Kanade feature tracker as discussed in Chapter 4 is used. The function *"[v, feature_state] =Pyramid(I0,J0,L,w,points,feature_state)"* is used. Here *points* is a vector containing the last coordinates of all the features, w is a 2×1 vector that determines the size of the tracking window $\{(2w_x+1)\times(2w_y+1)\}$, L is the number of levels in the Pyramid (not including the original image), and *I0* and *J0* are the first and second images respectively. The outputs are: v, which is a vector the same size as *points* and contains the displacement of each feature, "*feature_state*" is the matrix as defined in the previous section.
The function *Pyramid* sets up the image pyramid. This function then in turn calls function *"[v,status] = Sub_pix_Lucas_Kanade(imI,imJ,w,point,g)"* to perform the tracking at each level of the pyramid for each of the features. Here the inputs *imI* and *imJ* are the first and second images, g the estimate of the displacement from the previous level (as discussed in section 4.3), point is the last coordinate of the feature, and w has the same description as before. The outputs are v, which is the displacement vector for the feature in question, and *status* which is a variable that indicates when the feature can't be tracked anymore because it is too close to the image boundary.

Function *Sub_pix_Lucas_Kanade* uses a function called *"[s, status] = getsubvalue(point,M)"* which uses bilinear interpolation to calculate feature tracking windows. The value of *status* originates in this function. When a tracking window value cannot be interpolated because it is outside the image boundaries, the value of status is set to 0 instead of a normal 1.

Stellenbosch University http://scholar.sun.ac.za

Calculate affine deformation and residuals

The next step in the program is to calculate the residuals. The residuals are calculated by first determining the affine deformation parameters as described in section 4.5.
It is important to note that the residuals are calculated by comparing a current image to some reference image. In this study the reference image is initially chosen to be the first image, from then on the reference image is changed to every tenth image, for example images 11 to 20 are compared to image 10, 21 to 30 are compared to 20, and so on.
The affine deformation and residual calculations were done in a function called *"[res,status] = affine_def(first_im,im,w,point,d)"*. In this function the inputs *first_im* and *im* are the reference and current images respectively, *w* and *point* is as defined in the previous section, and *d* is the displacement vector of the feature as calculated by the tracking part of the program.
The outputs are *res*, which is the residual and *status* which has the same meaning as explained above and also originates from the function "*getsubvalue*" as described above.

Check residuals and reject bad features

This part uses a function called *"[feature_state] = check_res(res,feature_state)"*. The inputs to this function are *res*, which is the third level of the *features* matrix data block, and *feature_state*, which is the first column of the matrix "*feature_state*" as defined before. This function checks the residuals for outliers as described by the X84 rejection rule as discussed in section 4.5.2. The function then changes *feature_state* as necessary and returns it to the main procedure *long_sequence_track.*

Stellenbosch University http://scholar.sun.ac.za

Update data

This section of the code as well as the following sections is only performed every tenth frame.
The updating of the data block entails the removal of all entries of 2 and 3 in the *feature_state* matrix as well as their counterparts in the *features* data block, and then removing the blank rows in these two matrices. Thus the top parts of the updated *features* data block and *feature_state* matrix will consist only of entries of features that are being tracked, the rest will have no entries.

The updating of data also includes the updating of the reference image. The reference image is used in the calculation of the residuals. For example features in images 11 to 20 is compared to their counterparts in image 10, features in images 21 to 30 to features in image 20,and so on.

Cluster features into vehicles

This part of the code uses a function called "*feature_state = assign_points(features,feature_state,ind)*". This function uses the feature coordinates as given by the data-block *features* and groups the features into clusters as described in chapter 5. This function was also adapted to perform the clustering on the perspective transformed coordinates and was renamed to *assign_points(features,feature_state,ind,a,normaal,vekt)* to include the parameters necessary to perform the perspective transformation. This function is called from the main function *long_sequence_track2*. The inputs to this function are *features*, *feature_state* and *ind*. The variable *ind* indicates at which frame number the system is currently tracking, and *a*, *normaal* and *vekt* are the parameters for the perspective transformation.
The function *assign_points* allocates vehicle numbers and tries to keep these numbers the same through subsequent clustering. In cases where the clustering is not very accurate

Stellenbosch University http://scholar.sun.ac.za

features that belong to one cluster can be split into two different clusters, or clusters from different previous groupings can be clustered into the same group in the following clustering procedure. For this reason it is important that the numbering of clusters (vehicles) are kept as accurate as possible (one vehicle should have the same number throughout the tracking procedure).

In this study the rule was as follows: once the clustering has been performed, clusters are checked in a random fashion. The numbers assigned to features from a previous clustering are checked and the number that is most common in the cluster is used, given that it has not been used previously. If that specific number has been assigned in this round of clustering a new vehicle number is used. The numbers are assigned from 1 to 9999 and the vehicle number after 9999 is again 1.

There are two matrices and an index used in the number assignment. *OLD_VEHICLES* is a matrix containing the previous round's numbering. *VEHICLES* contains the latest vehicle numbering and *VEHICLENUMBER* is the index that repeats from 1 to 9 999. The last three variables are defined globally in the program structure and are therefore written in capital letters.

The matrix VEHICLES has five columns, these are:

1. Vehicle number
2. Centroid image-row (x) value
3. Centroid image-column (y) value
4. Vehicle speed
5. An index showing whether it has been counted as part of the traffic flow.

Segment image and add new features

In this part of the code the same procedure is followed as in the first section to segment the image. The next set of features are selected in the same way as the initial feature

Stellenbosch University http://scholar.sun.ac.za

selection except that a small region around each of the existing features is defined so that a new feature cannot be selected on top of an existing one.

Calculate speed, density and flow

The traffic flow parameters are discussed in section 2.1.1.
The displacement of each vehicle is determined by calculating the displacement of the centroid of features that has been tracked for more than 19 frames. Calculating the real world coordinates of the centroids 19 frames back and subtracting it from the current real world coordinates yields the displacement. This magnitude of the displacement is then divided by the elapsed time to obtain the speed of each vehicle.

The density is obtained by defining a zone in between two lines on the image. The length of this zone can be obtained by calculating the real world coordinates where the lines cross the section of road. The amount of vehicle centroids within this zone divided by the zone length is the density. Although not done the density can alternatively be calculated by dividing the flow by the space mean speed (using the relationship between traffic flow parameters as discussed in section 2.1.1).

The flow is incremented once a vehicle's centroid has passed a line on the image. The matrix *VEHICLES* also contains a column that is marked if a vehicle has been counted. This is used in order to prevent a vehicle being counted twice if the centroid is shifted backward at the stage when new features are added (e.g. if the centroid of the vehicle has only passed the line by a small amount in the last ten frames, the centroid may be shifted in front of the line if enough features are added towards the rear of the vehicle).

Stellenbosch University http://scholar.sun.ac.za

7.2 Processing time issues

The system as implemented in MATLAB code is extremely computationally intensive. The processing speed of the system should however be greatly improved if the system was to be coded into a programming language such as *C*. In extreme cases (heavy traffic and thus a large amount of frames) the processing speed of the current system can be as slow as 20 frames per hour.
The slow processing speed is attributed to the MATLAB code, but can also be partially explained by inefficient program structures. The focus of this study however was on the investigation of image processing techniques that can be used for vehicle tracking and thus the real time implementation was not a criterion that received much attention.
The slow processing speed has however limited the testing of the system to series smaller than 400 or 500 frames. The results of tests conducted are reported in the following chapter.

The most time consuming part of the current code appears to be that of setting up the data for the tracking algorithm, performing the iterations and repeating the process for each level of the image pyramid. The process of setting up the data for determining the affine deformations and iterating the process several times appears to be the second most time consuming part of the code.

Stellenbosch University http://scholar.sun.ac.za

Chapter 8

Results, conclusions and recommendations for future studies

8.1 Camera set-up

There are various issues that have to be taken into account when searching for a suitable camera position. These are:

- Shadows: Objects on the side of the road can often have shadows at dusk and dawn that will fall across the road surface. These shadows are problematic because they change the intensity of the feature window. The "shape" of the feature is often also changed because the direction of illumination is changed. The change in the feature is problematic for both the feature tracking and the feature monitoring. The feature-monitoring problem can be partially solved by using a photometric normalisation as described in section 4.6.1.1. However if the feature window is only partially shaded this technique will not work. In the case of shadows, features might not be tracked accurately and will be discarded by the feature monitor. In the event that the feature is tracked accurately, the feature monitor will most likely discard the feature due to the change in the appearance of the feature-tracking window.
- Focal length: The advantages of using a longer focal length for filming a scene is that the perspective distortions are reduced. The magnitude of vehicle sizes does not change by as much as when using shorter focal lengths. This reduces the need for perspective transformations before the clustering procedure is applied (discussed in section 6.4). This problem is only found when there is a relatively small angle between the line of sight of the camera and the road pavement. The disadvantage of using a longer focal length is that camera movement is much

Stellenbosch University http://scholar.sun.ac.za

more noticeable and the movement or vibration of the entire image can become a problem. This should be addressed by identifying a single fixed point or feature on the reference image, tracking the displacement of this feature, and moving the entire image in the opposite direction of the displacement for every frame. Large perspective deformations should be avoided.

- Vertical angle: Perspective distortion in road scenes (the closer a vehicle is to the camera the larger it seems) is a function of the angle that the camera's line of sight makes with the road (among other factors). These distortions are problematic for feature tracking, monitoring, clustering and the perspective transformation. Viewing the scene from directly above might solve some problems, but such vantage points are often not available. This also introduces different problems. If the road is viewed from above and the camera is not high enough the speed of the vehicles might be too fast for the feature tracker. The normal Lucas Kanade feature tracker is only accurate for movements of 3-4 pixels per frame (depending on the tracking window size). With the pyramidal implementation it is expected that this distance will be increased by a maximum factor of 2^m where m is the number of levels in the pyramidal implementation (excluding the full resolution image). If the vantage point is high enough to eliminate the above problem it can be found that the size of vehicles on the image might be fairly small. If vehicles display too small they might not have clearly recognisable features, or feature windows might overlap onto the pavement and would thus not be ideal for tracking.
- Horizontal angle: If the vertical angle is small then the horizontal angle (angle of camera with direction of travel) should preferably be small. If the angle is large (road filmed from its side) vehicles in the closer of the two lanes will obscure the view onto the lanes behind it. Furthermore traffic moving in the opposite direction will interfere with the segmentation process.
- Terrain: The road section that is filmed should have a constant gradient. If the gradient is not constant the assumption (made in the perspective transformation) that the features are located in a plane will be violated and the perspective transformation coordinates will not be accurate.

Stellenbosch University http://scholar.sun.ac.za

- Length of section: The length of the section filmed should be adequate that vehicles remain in line of sight for at least 25 frames. Due to the variability or inaccuracy of the feature tracker some of the calculations need to be averaged over a fixed number of frames and consequently the vehicle needs to be tracked for long enough.

8.2 Experimental results

Three image sequences of various lengths were tested. The output image sequences are included on a compact disc attached to this report. The output image sequences differ from the input images in the following ways:

- The input images are in colour and the output images are grey images.
- The output images contains three lines drawn across it, the outer two lines define the region within which the vehicle density is calculated and the line between the other two are used to increment the traffic flow counter (if a vehicle crosses this line the flow is increased by one).
- The feature positions are indicated by coloured squares. The features belonging to the same cluster are the same colour.

The output images were named *out000.jpg* where the input image number replaces the *000*. An image sequence output does not necessarily start at *out001.jpg*.

Stellenbosch University http://scholar.sun.ac.za

Sequence 1:

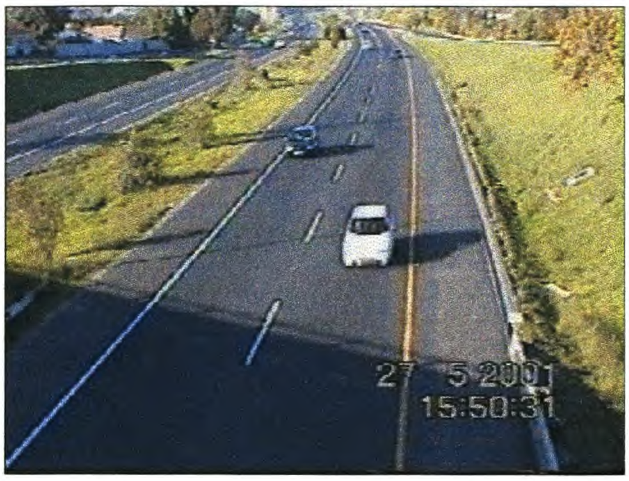


Figure 8.1: An image from sequence 1

Figure 8.1 was taken from this image sequence of 500 frames (*out100.jpg* to *out600.jpg*). Sequence 1 was taken of the N1 towards Cape Town from the Durban road bridge in the afternoon (approximately 15:50). The weather condition was sunny. The speed limit and expected operating speeds of vehicles are both 120 km/h. The length of roadway that could be used for vehicle tracking was approximately 120m.

This sequence was tested twice. In the first test the clustering was performed on the image coordinates, and in the second test the features were clustered using the coordinates as obtained through the perspective transformation.

Results from clustering without perspective transformation

The first important observation of this sequence is the rejection of features as the shadow of the bridge falls on the passing vehicles (file *out134.jpg-out136.jpg*).

The second important observation is the tendency of the clustering algorithm to group all features into at least two groups even if there is only one vehicle present on the section of

Stellenbosch University http://scholar.sun.ac.za

road when all the features belong to that vehicle. This problem is discussed in section 5.3 (files *out170.jpg* to *out200.jpg*).

The third observation displays the effect of a high comparison threshold when segmenting the image. Images *out290.jpg* to *out320.jpg* show a vehicle that passed through the images without being identified. The lowering of the comparison threshold could however cause an increase in the noise sensitivity and the identification of a feature where there is none. An example of this is found in the sequence in frames *out440.jpg* to *out600.jpg* where a feature was selected on the roadway. The feature on the roadway was stationary; consequently it could be tracked while there was no reason for it to be rejected. It would only be rejected once a vehicle passed over it.

The feature tracker counted a total of 12 vehicles and there were in fact only 12 vehicles that passed the reference line in this sequence. This is however not an indication of the accuracy of the tracker since there are instances where vehicles were not identified and instances where the features on a single vehicle were split into two or more clusters.

Results from clustering with perspective transformation

This sequence exhibited the same behaviour of missing dark vehicles and splitting single vehicles as in observations one and three above. This can be expected since the feature selection is done exactly as in the previous example. The feature clustering has however changed due to the transformation of image coordinates to real world coordinates but no marked improvement was observed. The clustering reached its worst state in frames *p_out130.jpg* to *p_out150.jpg*. The vehicle tracker however counted a total of 11 vehicles in this sequence; this seems to indicate that the vehicles might not have been split as much as in the previous test.

Stellenbosch University http://scholar.sun.ac.za

Sequence 2

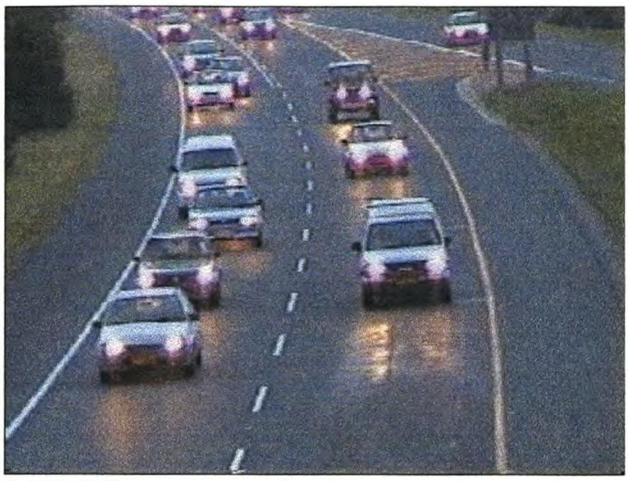


Figure 8.2: An image from sequence 2

Sequence 2 was 400 images long (*out100.jpg* to *out500.jpg*) and Figure 8.2 was taken from this sequence.
Sequence 2 was taken of the N1 towards Cape Town from the Durban road bridge in the early morning (approximately 07:30). The weather condition was cloudy. The speed limit and expected operating speeds of vehicles are both 120 km/h. The length of roadway that could be used for vehicle tracking was approximately 250m.

The first important observation of this sequence is the effect of large vehicles such as trucks. The features on these large vehicles tend to be grouped into more than one cluster even if they are among other vehicles. This is due to the often-large distance between features on the vehicle.

The feature clustering in this sequence was performed on the image coordinates and not the real world coordinates. The effect of the perspective distortion is clearly shown by the way that it causes the groups of features situated on the headlights of the same vehicles to be clustered into two groups because of the apparent increase in the distance between the headlights as the vehicle approaches the camera.

Stellenbosch University http://scholar.sun.ac.za

Another important observation is the apparent lack of accurate clustering around frames *out400.jpg* to *out410.jpg*. This is once again attributed to the lack of accurate real world coordinates. Vehicles far away from the camera seem smaller and their movement in relation to each other seems smaller. This often causes the vehicles that are far away from each other to be grouped into one cluster. As the vehicles approach the camera the features move further apart and their movement in relation to each other becomes larger. This tends to cause features on one vehicle to be grouped into more than one cluster. It is obvious that an accurate perspective transformation would solve this problem.

Sequence 3

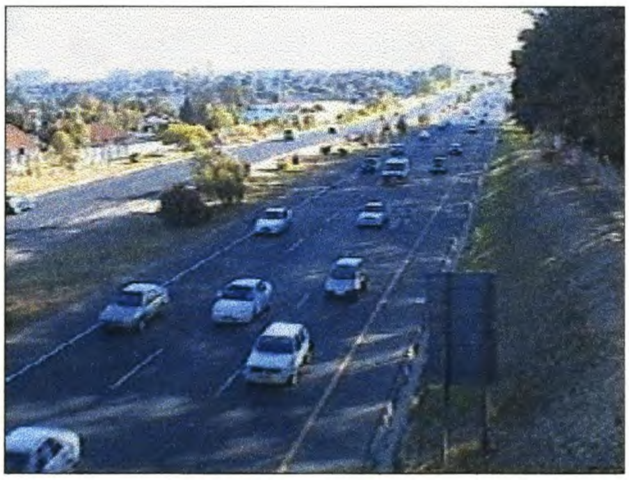


Figure 8.3: An image from sequence 3

Sequence 3 was taken of the N1 towards Paarl from the Bill Bezuidenhout road bridge in the afternoon (approximately 17:00). The weather condition was sunny. The speed limit and expected operating speeds of vehicles are both 120 km/h. The length of roadway that could be used for vehicle tracking was approximately 120m.

This sequence was used to show the detrimental effect of shadows across the road. In this section features could neither be tracked reliably nor accurately. This negatively affected the accuracy of the clustering. This behaviour of the vehicle surveillance system is an

Stellenbosch University http://scholar.sun.ac.za

example of the breakdown of the system. This phenomenon is discussed in section 8.3 dealing with the choice of camera location.

Once vehicles pass under a shadow, the intensity of its features changes and accurate tracking is not possible anymore. Furthermore the tracker is inclined to reject these features due to the changes that they undergo.

Another problem that can be seen in this sequence is features that are chosen on the road surface due to the movement of the shadows. Since the background computation is based on the assumption that the background is stationary, moving shadows cannot be removed. The moving shadows are segmented and subsequently features can be identified on these moving shadows.

Stellenbosch University http://scholar.sun.ac.za

8.3 Conclusions and recommendations for future studies

In this study some image processing techniques were investigated that could be used in a traffic surveillance system.
The study proved that the background of a highway scene could be obtained clearly and this allowed for reasonably accurate segmentation of the scenes.
Once a scene was segmented features could be identified in a way that suited the tracking algorithm. The most prominent features are always selected allowing the tracker to be used in a wide variety of lighting conditions (e.g. at night the headlights and tail lights are prominent while in daytime any corners are sufficient).

Features were tracked reliably using the Lucas Kanade feature tracker, which was implemented in a pyramidal image scheme. The study proved that features could be accurately monitored even when deformations were present so that when features were lost or obscured they could be rejected.

The clustering of features was also investigated and the Ward clustering algorithm is discussed as a tool for assigning features to individual vehicles.
A perspective transformation along with a method for its calibration was derived.

One of the most important conclusions drawn from this study is that in order to develop a robust and accurate traffic surveillance system the problems associated with the perspective transformation will have to be overcome.
One of the possibilities that could be researched in order to overcome the problems associated with a perspective transformation is three-dimensional reconstruction. In three-dimensional reconstruction the assumption that the features are located in a single plane is not present and it may be possible to calculate real world coordinates more accurately. Another possibility is the use of stereovision. In stereovision there are two cameras filming the same scene from different angles. If the same feature can be identified from both angles, the 3D coordinates of features can be calculated.

Stellenbosch University http://scholar.sun.ac.za

Another recommendation for future studies is to find a technique that will eliminate the adverse effect that shadows across the road surface have on the operation of the feature tracker. This will allow the tracker to be used in a wider variety of road conditions.

The use of colour information from the input images instead of the use of grey scale images only to increase accuracy of both the tracking and clustering of features is another research opportunity that could be considered.

The last opportunity for future research identified is the automated detection of the region of interest (roadway) and the automated calibration of the camera set-up. This will eliminate the need to calibrate the surveillance system for each different scene and therefore make the installation of such a system much easier.

Even though a robust and accurate traffic surveillance system could not be realized in this study, it is believed that some of the initial work for the development of a traffic surveillance system has been laid down and potential problems and pitfalls were identified. It is clear that more research effort will be needed in the future to realize a robust and accurate traffic surveillance system.

Stellenbosch University http://scholar.sun.ac.za

Bibliography

[1] K. Ogata, *Modern Control Engineering, Third Edition*. University of Minnesota, USA: Prentice Hall, 1997.

[2] B. Coiffman, D. Beymer, P. McLauchlan and J. Malik, *A Real-Time Computer Vision System for Vehicle Tracking and Wide-Area Traffic Surveillance*, Transportation Research Board, 1998, University of California, Berkeley, California, USA.

[3] J. P. S. A. Costeira, *A Multi-Body Factorization Method for Motion Analysis,* Instituto Superior Tecnico, Universidade Tecnica de Lisboa 1995.

[4] J. L. Barron, D. J. Fleet and S. S. Beauchemin, *Performance of Optical Flow Techniques*, International Journal of Computer Vision, Vol. 12:1, pp 43-77, 1994.

[5] J. Shi and C. Tomasi, *Good Features to Track*, IEEE Conference on Computer Vision and Pattern Recognition (CVPR94) Seattle, June 1994.

[6] T. Tommasini, A. Fusiello, E. Trucco and V. Roberto, *Making Good Features Track Better*, IEEE Conference on Computer Vision and Pattern Recognition, Santa Barbara, California, USA, 1998.

[7] F. R. Hampel, *Robust Statistics: The Approach Based on Influence Functions*, New York, Wiley, 1986.

[8] M. R. Anderberg, *Cluster Analysis for Applications*, University of Texas at Austin Academic Press, 1979

[9] J. C. Russ, *The Image Processing Handbook, 2nd* Edition, Florida, CRC Press, 1995.

[10] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, Addison Wesley 1992.

[11] C.S. Papacostas and P. D. Prevedouros, *Transportation Engineering and Planning, 2nd Edition*, Hawaii, Prentice Hall, 1993.

[12] South African National Department of Transport, *Moving South Africa – The Action Agenda: A 20 year Strategic Framework for Transport in South Africa*, 1999.

Stellenbosch University http://scholar.sun.ac.za

[13] South African Society for Intelligent Transport Systems, *ITS – Part of the Solution to the Transport Challenges of South Africa – Information Document for Discussion*, Pretoria, South Africa, 2000, http://www.sasits.com.

[14] J. Bouguet, *Pyramidal Implementation of the Lucas Kanade Feature Tracker: Description of the Algorithm*, Intel Corporation Microprocessor Research Labs, Open Source Computer Vision Library, http://www.intel.com/research/mrl/research/opencv/.

Stellenbosch University http://scholar.sun.ac.za

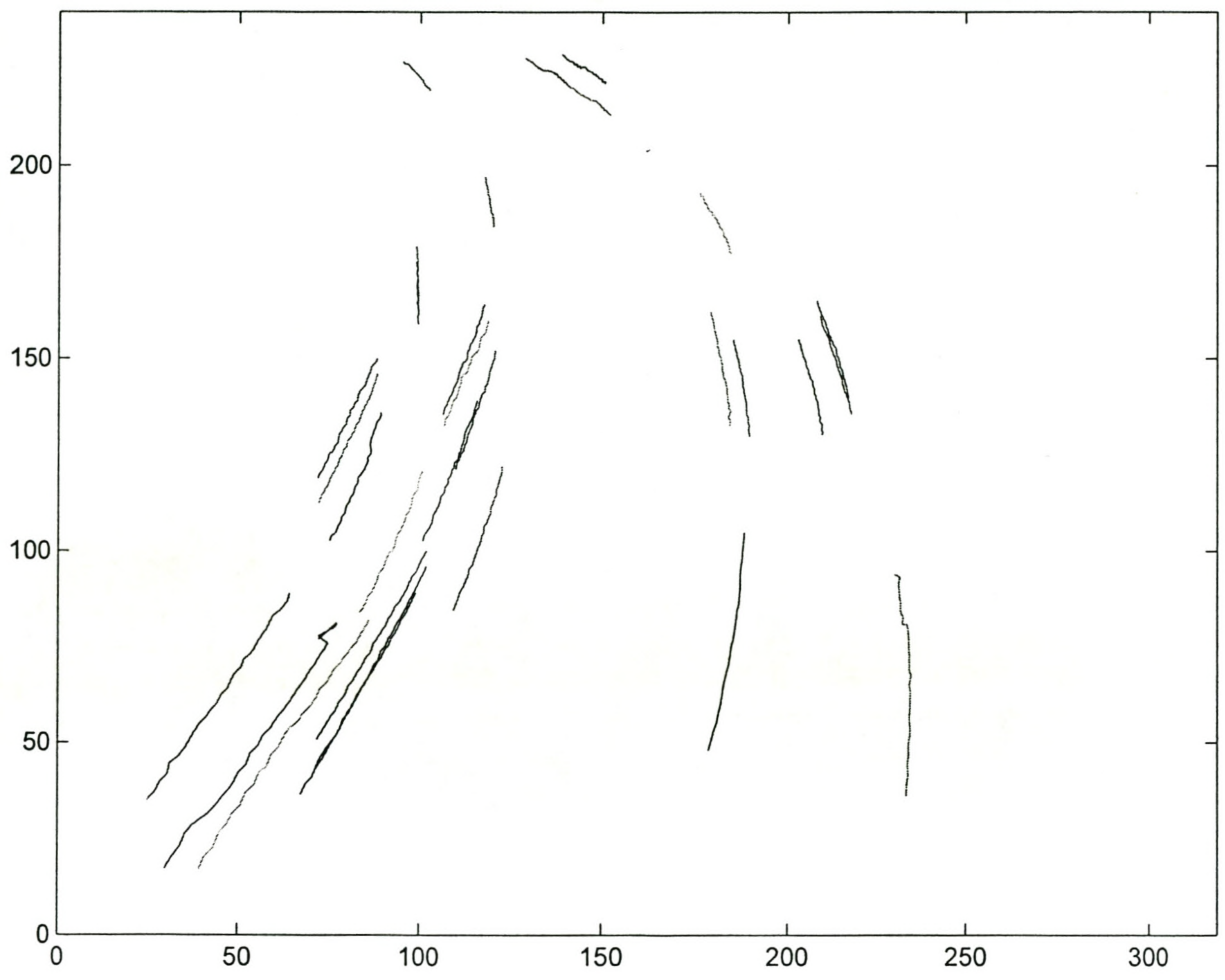
Appendix A

Tracking lines for different tracking window sizes and pyramid heights

Refer to section 4.4 p 41 for discussion.

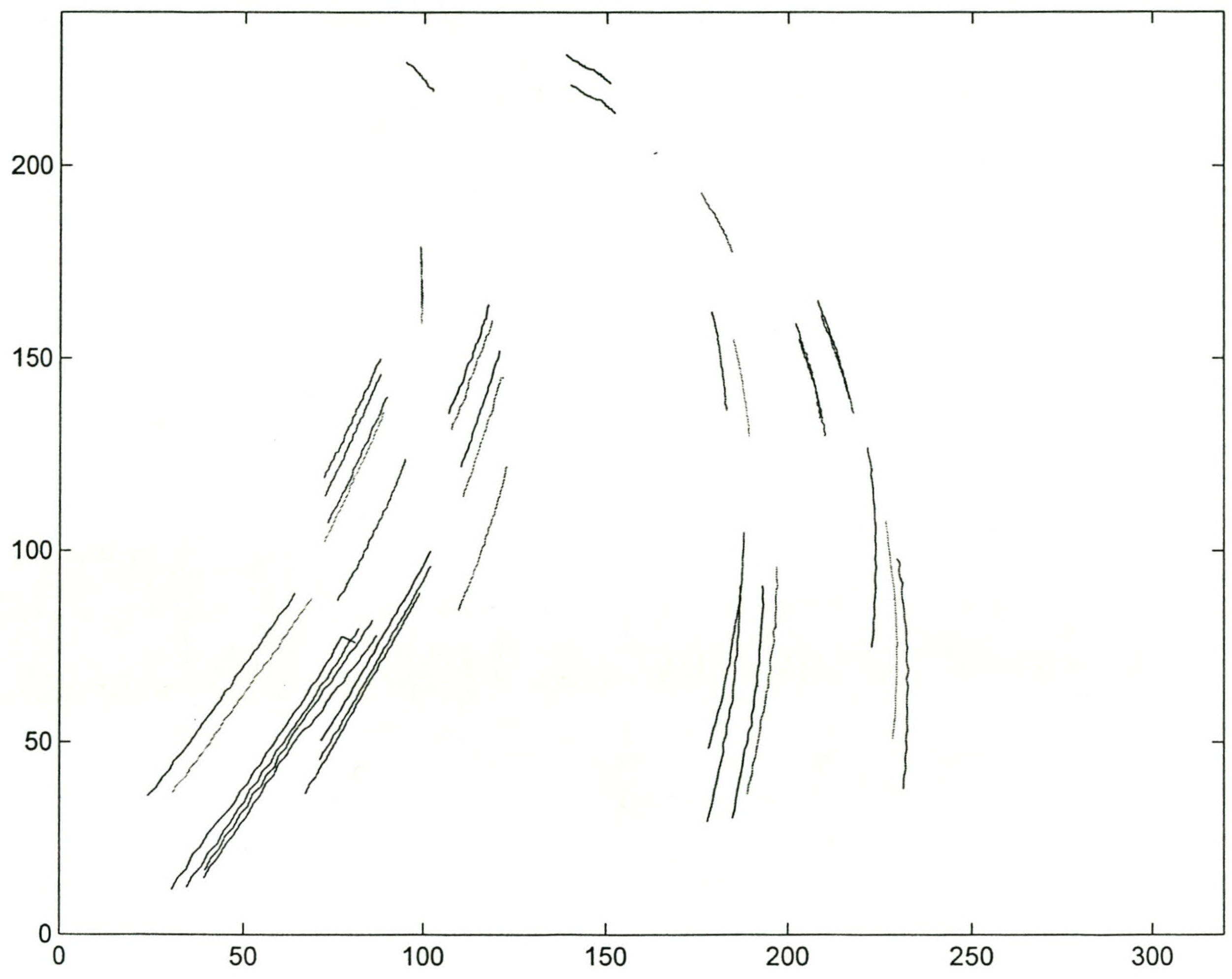
Stellenbosch University http://scholar.sun.ac.za

Figure A1: Tracking lines
No Pyramid, Tracking window: 5*5



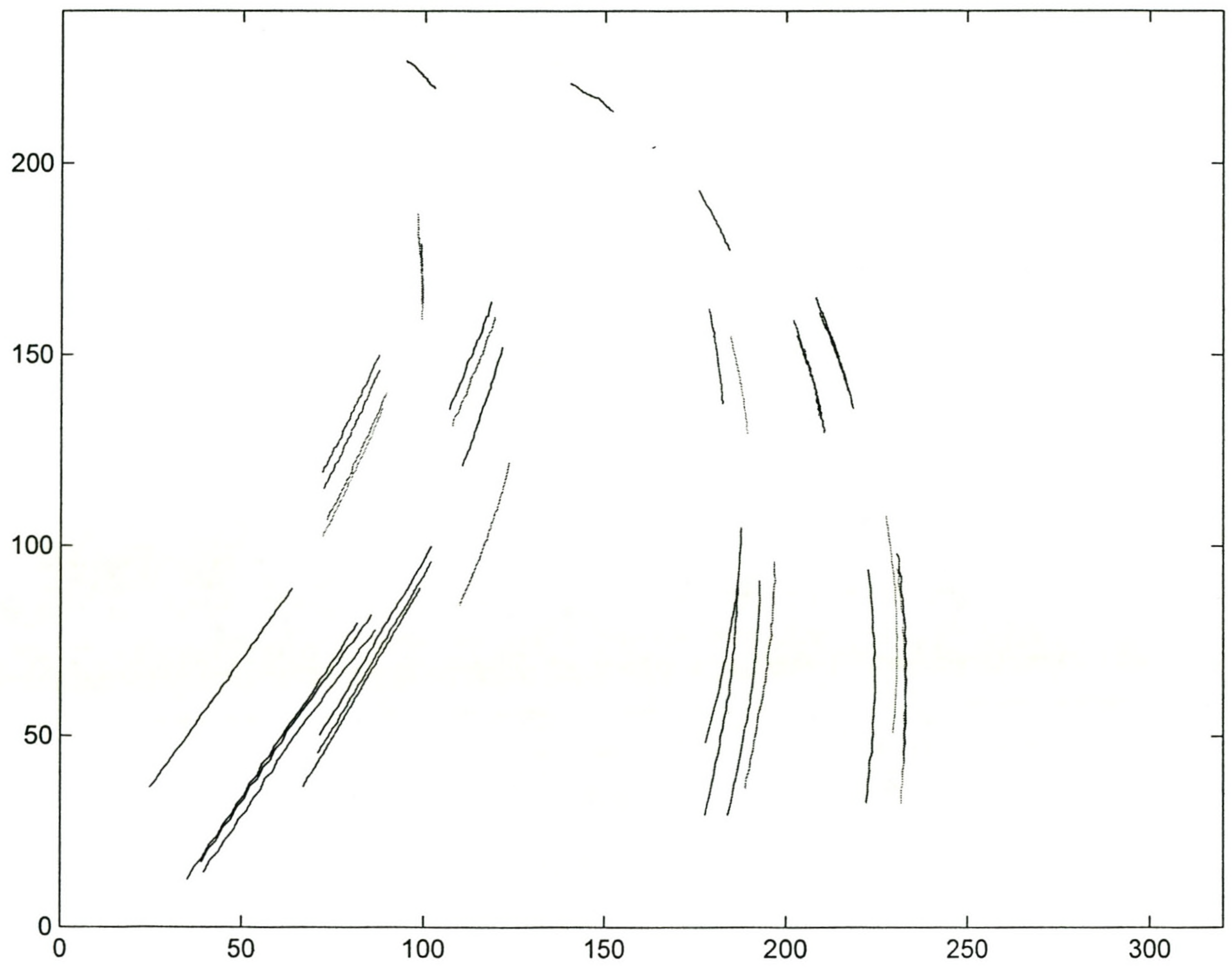
Stellenbosch University http://scholar.sun.ac.za

Figure A2: Tracking Lines
No Pyramid, Windowsize: 7*7



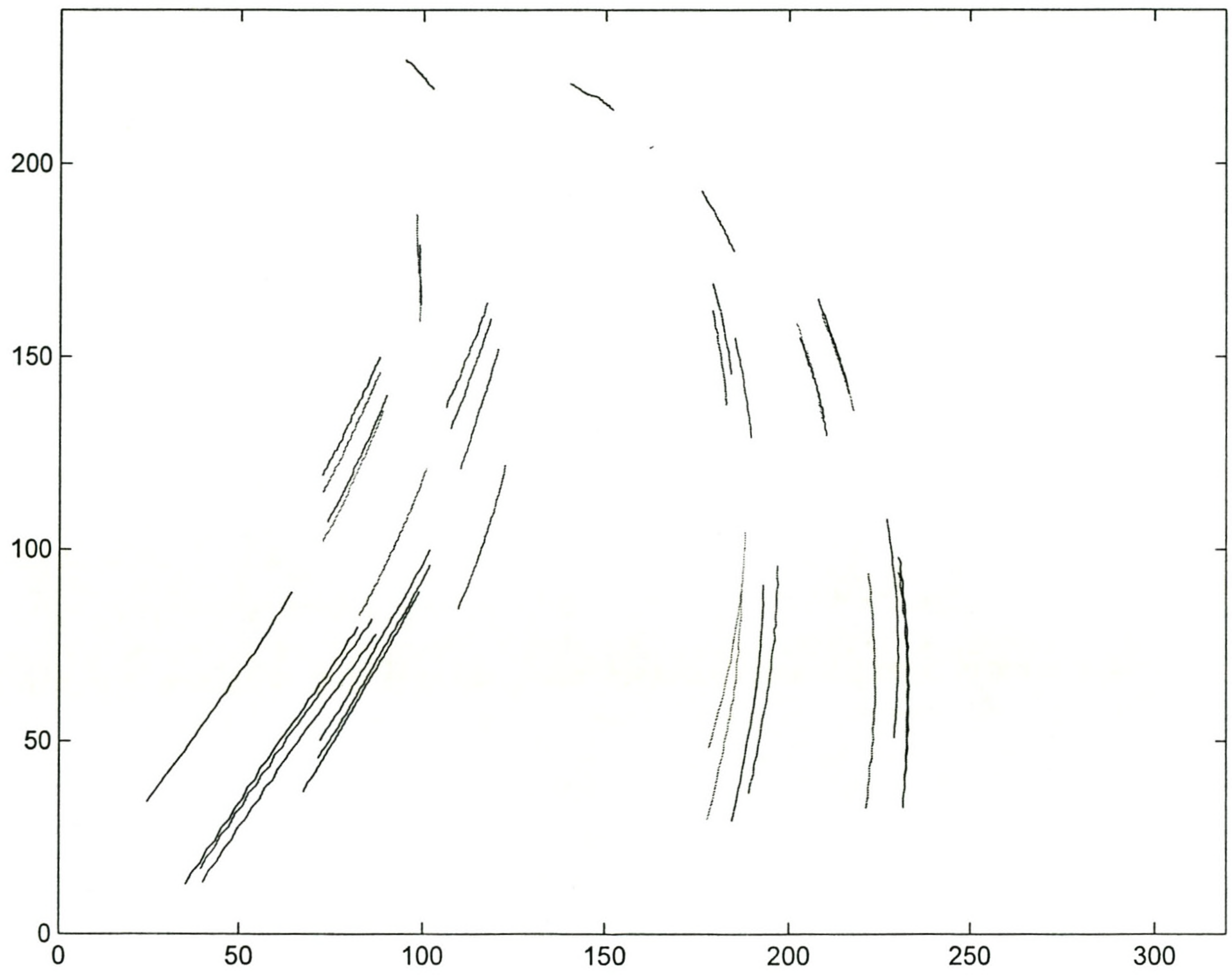
Stellenbosch University http://scholar.sun.ac.za

Figure A3: Tracking Lines
No Pyramid, Tracking Window: 9*9



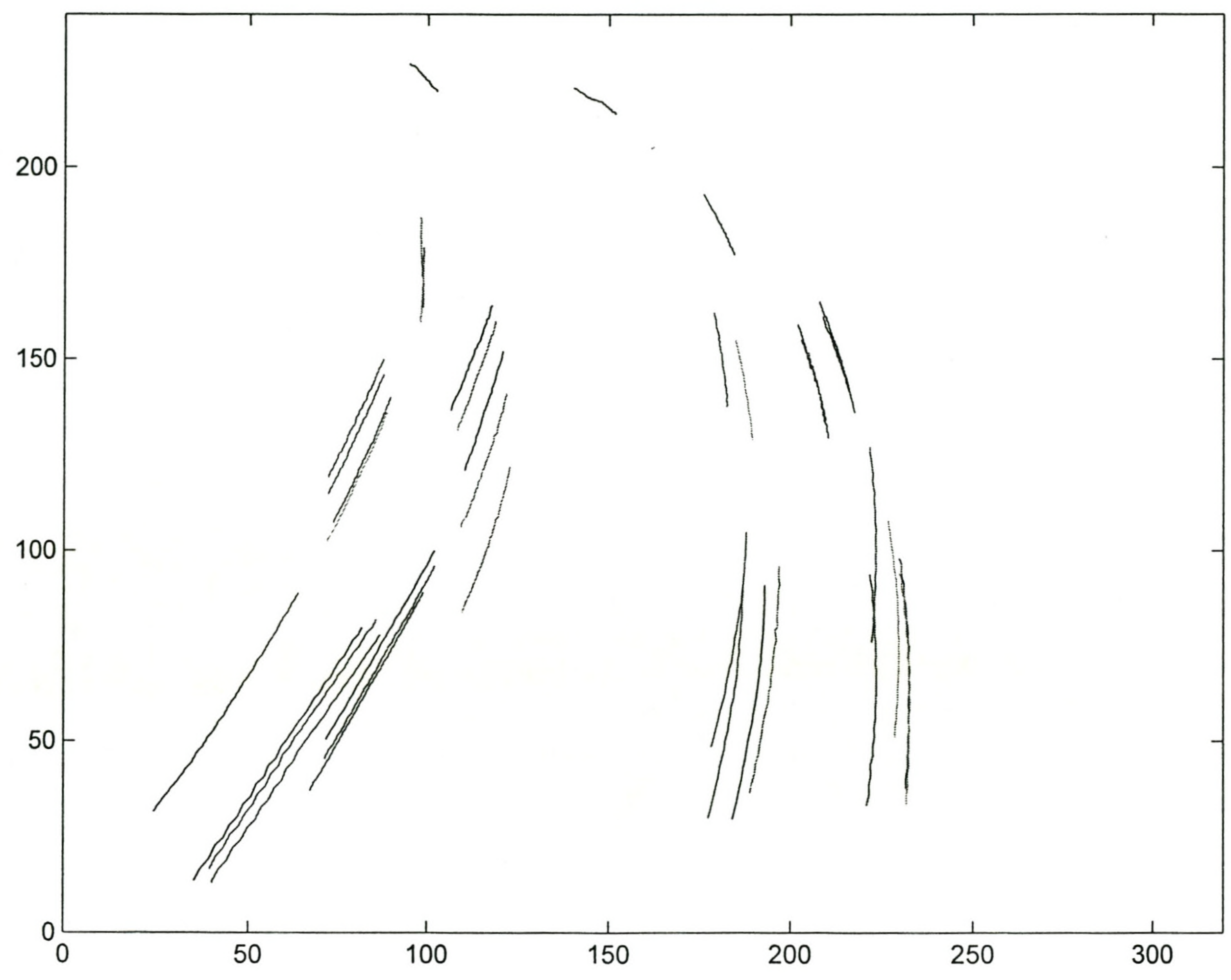
Stellenbosch University http://scholar.sun.ac.za

Figure A4: Tracking lines
No Pyramid, window Size: 11*11



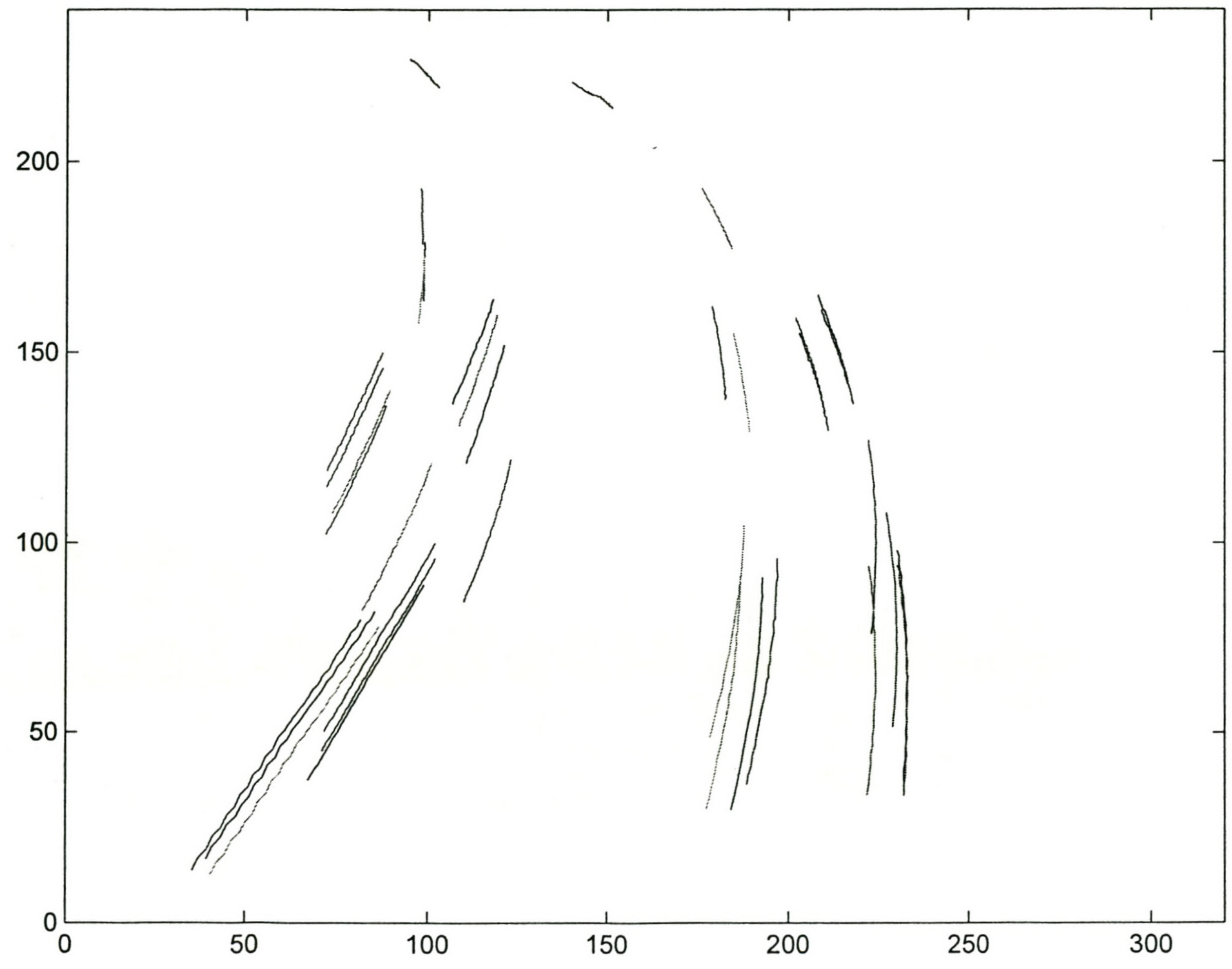
Stellenbosch University http://scholar.sun.ac.za

Figure A5: Tracking lines
No Pyramid, Window Size: 13*13



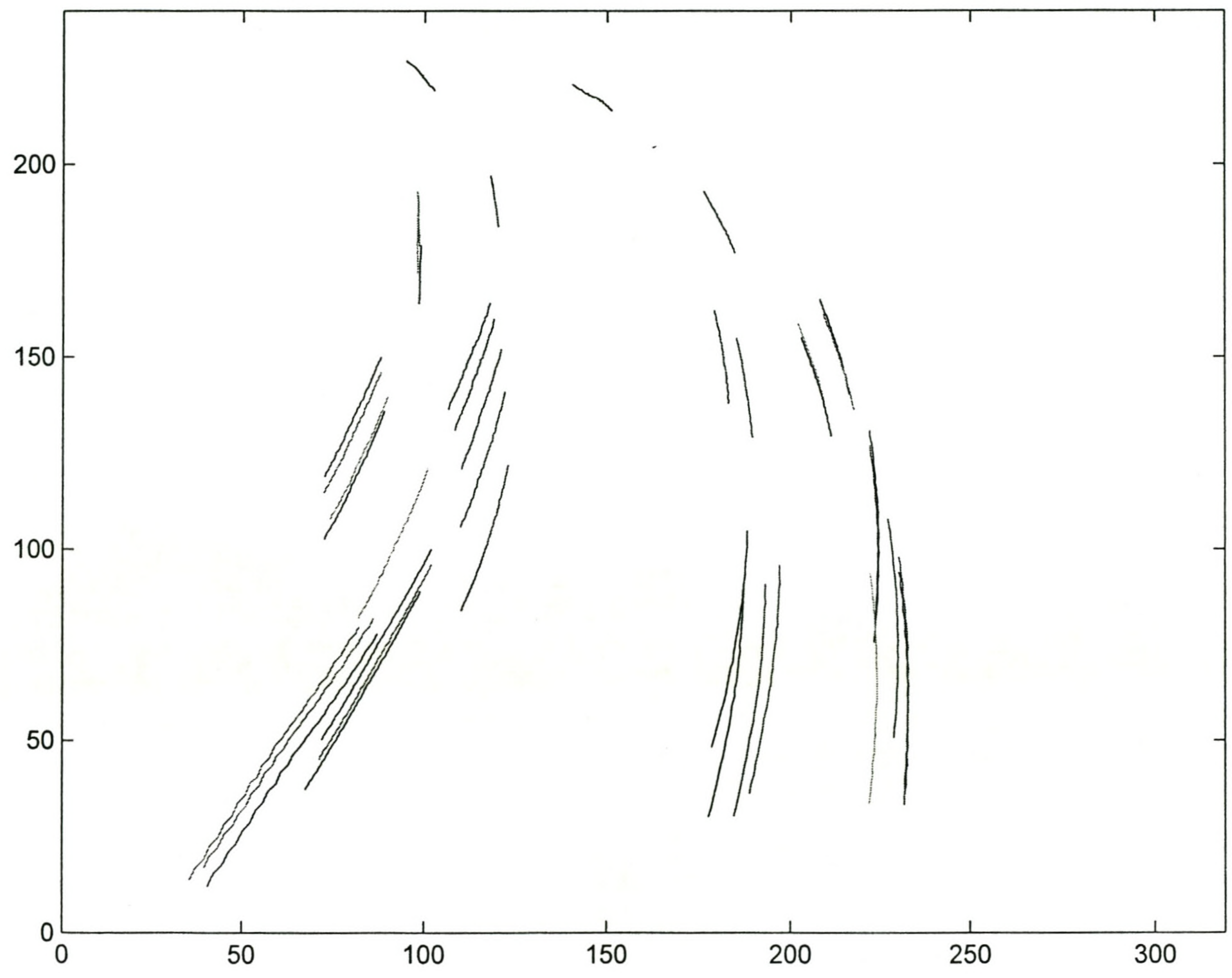
Stellenbosch University http://scholar.sun.ac.za

Figure A6: Tracking Lines
No Pyramid, Window Size: 15*15



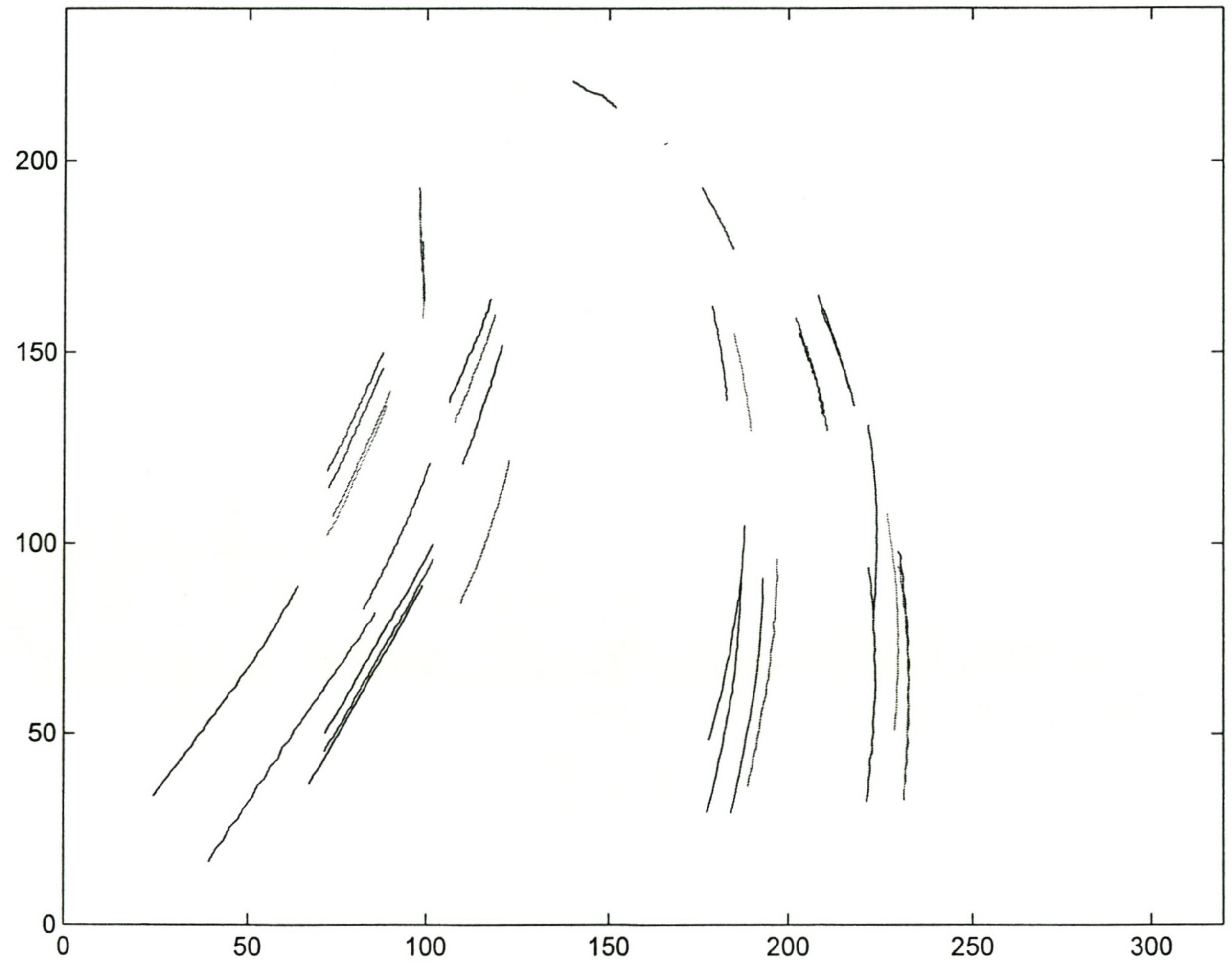
Stellenbosch University http://scholar.sun.ac.za

Figure A7: Tracking Lines
No Pyramid, Window Size: 17*17



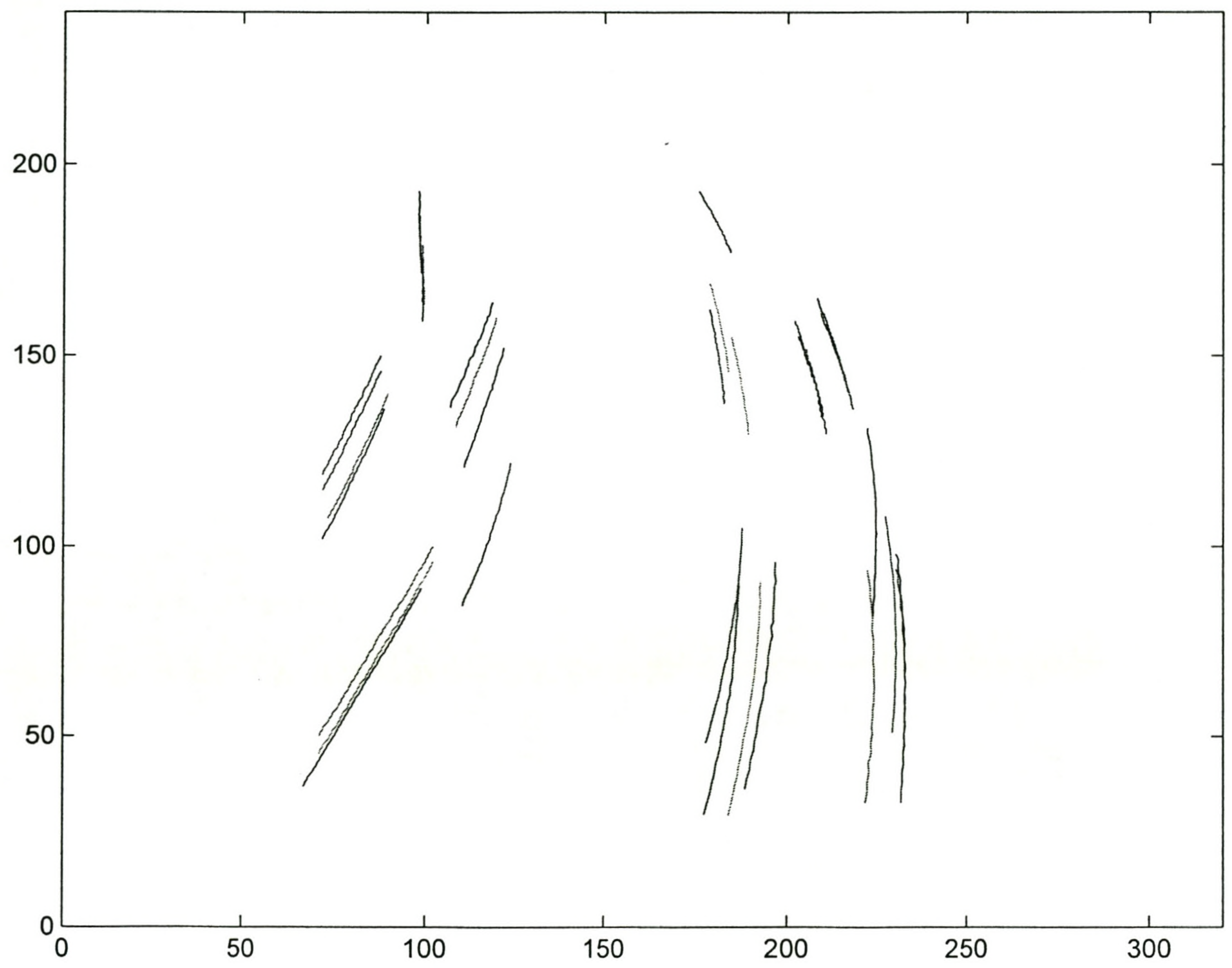
Stellenbosch University http://scholar.sun.ac.za

Figure A8: Tracking Lines
Pyramid Height: 1. Window Size: 11*11



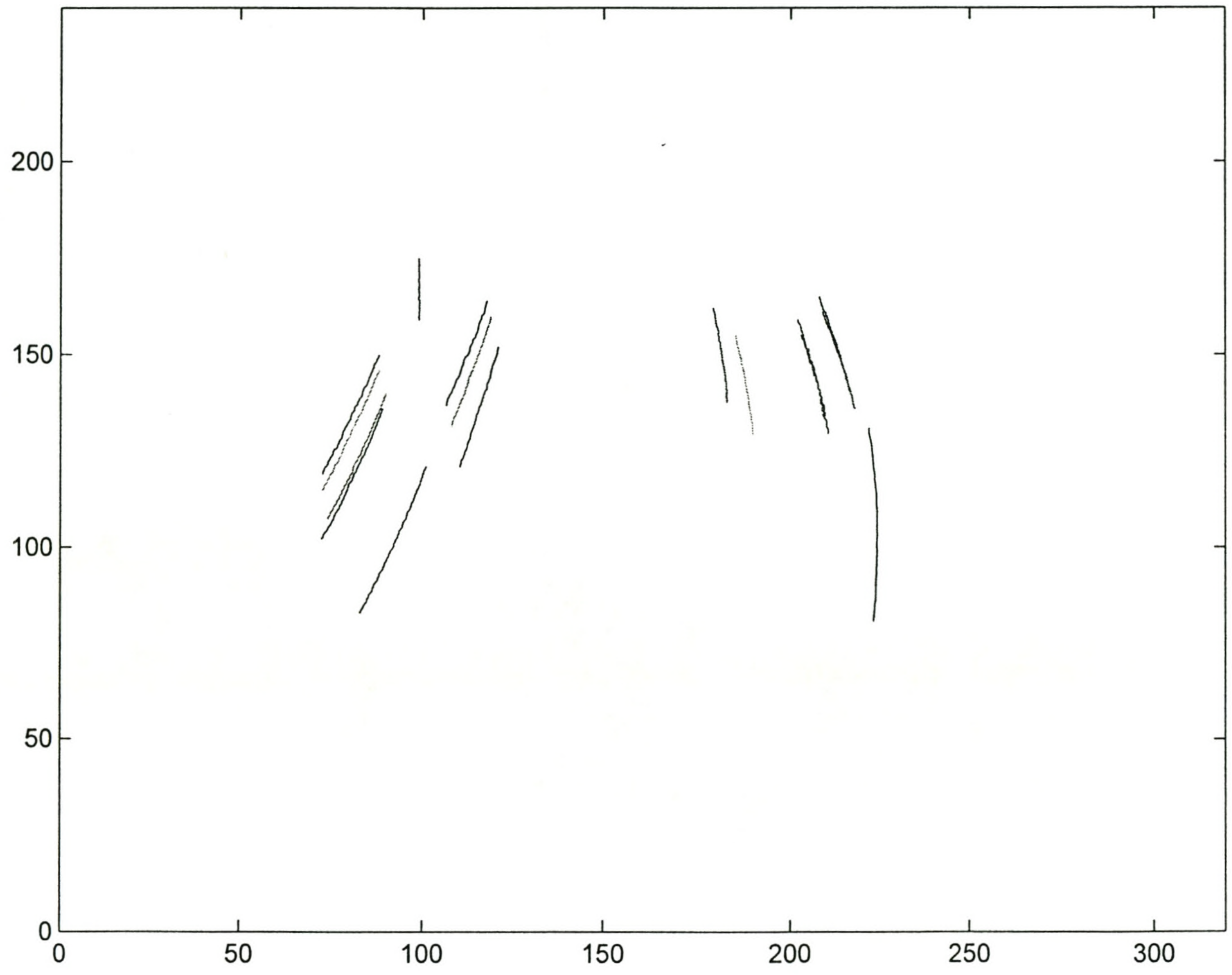
Stellenbosch University http://scholar.sun.ac.za

Figure A9: Tracking Lines
Pyramid Height: 2, Window Size: 11*11



Stellenbosch University http://scholar.sun.ac.za

Figure A10: Tracking Lines
Pyramid Height: 3, Window Size: 11*11



Stellenbosch University http://scholar.sun.ac.za

Appendix B

MATLAB code

Stellenbosch University http://scholar.sun.ac.za

Stellenbosch University http://scholar.sun.ac.za

```
function [res,status] = affine_def(first_im,im,w,point,d)

% This function finds the affine deformation matrix
% See "Making Good features track better"
% first_im and im are the first and current images respectively (grayscale images)
% w is a 1*2 matrix and specifies the window horizontal windowsize as (x+w(1,1),x-
w(1,1)
% and the vertical window size as y +w(1,2),y-w(1,2) and must be integer
% point is the coordinate around which will be searched for the deformation
% d is the displacement already calculated using the pure translational assumption
method
% d is in the form [Dx;Dy]
% if status is set to 0 from the getsubvalue routine the feature is too close to the
image perimeter
status = 1;
res = 0;
loops = 0;
rep = 1;
Adef = eye(2); % initially assume no deformation
% Extract subpixel values for A
A = zeros(2*w(1,1)+3,2*w(2,1)+3);
ry = 1;
kolom =1;
for i = point(1,1)-w(1,1)-1 :1: point(1,1)+w(1,1)+1
   for j = point(2,1)-w(2,1)-1 :1: point(2,1)+w(2,1)+1
      [A(ry,kolom),status] = getsubvalue([i;j],first_im);
      if status == 0
         return
      end % if status == 0
      kolom = kolom+1;
   end % for j
   ry = ry +1;
   kolom = 1;
end % for i
% Finished extracting matrix A
%imshow(A,[min(min(A)) max(max(A))]);

Ix = sobel(A,'x');
Iy = sobel(A,'y');

T = zeros(6);
G = zeros(2);
U = zeros(4);
Vt = zeros(2,4);
for i = 2:1:2*w(1,1)+2
   x = -w(1,1)-2+i;
   for j = 2:1:2*w(2,1)+2
      y = -w(1,1)-2+j;

      G = G + [(Ix(i,j))^2 , Ix(i,j)*Iy(i,j) ; Ix(i,j)*Iy(i,j) , (Iy(i,j))^2];
      U = U + [x^2*Ix(i,j)^2 , x^2*Ix(i,j)*Iy(i,j) , x*y*Ix(i,j)^2 ,
x*y*Ix(i,j)*Iy(i,j) ;...
               x^2*Ix(i,j)*Iy(i,j) , x^2*Iy(i,j)^2 , x*y*Ix(i,j)*Iy(i,j) ,
x*y*Iy(i,j)^2 ;...
               x*y*Ix(i,j)^2 , x*y*Ix(i,j)*Iy(i,j) , y^2*Ix(i,j)^2 ,
y^2*Ix(i,j)*Iy(i,j) ;...
               x*y*Ix(i,j)*Iy(i,j) , x*y*Iy(i,j)^2 , y^2*Ix(i,j)*Iy(i,j) ,
y^2*Iy(i,j)^2 ];
      Vt = Vt + [x*Ix(i,j)^2 , x*Ix(i,j)*Iy(i,j) , y*Ix(i,j)^2 , y*Ix(i,j)*Iy(i,j) ;...
                 x*Ix(i,j)*Iy(i,j) , x*Iy(i,j)^2 , y*Ix(i,j)*Iy(i,j) , y*Iy(i,j)^2 ];
   end % for j
end % for i

T(1:4,1:4) = U;
T(5:6,5:6) = G;
T(5:6,1:4) = Vt;
T(1:4,5:6) = Vt';

% start iteration here
while rep ==1
   % Extract subpixel values for B
   B = zeros(2*w(1,1)+1,2*w(2,1)+1);
```

Stellenbosch University http://scholar.sun.ac.za

```
   ry =1;
   kolom = 1;
   for i = point(1,1)-w(1,1):1:point(1,1)+w(1,1)
      for j = point(2,1)-w(2,1):1:point(2,1)+w(2,1)
         coord = Adef*[i-point(1,1);j-point(2,1)]+point + d;
         [B(ry,kolom),status] = getsubvalue(coord,im);
         if status == 0
            return
         end % if status == 0

         kolom = kolom +1;
      end % for j
      ry = ry+1;
      kolom = 1;
   end % for i
   % Finished extracting matrix B

   deltaI = double(A(2:2*w(1,1)+2,2:2*w(2,1)+2)) - double(B); % A-B
   a = zeros(6,1);
   for i = 2:1:2*w(1,1)+2
      x = -w(1,1)-2+i;
      for j = 2:1:2*w(2,1)+2
         y = -w(2,1)-2+j;
         a = a + deltaI(i-1,j-1)*[x*Ix(i,j) ; x*Iy(i,j) ; y*Ix(i,j) ; y*Iy(i,j) ;
Ix(i,j) ; Iy(i,j) ];
      end % for j
   end % for i

   Tn = T(:,1:4);
   z = zeros(4,1);
   an = a;
   z = Tn\an;
   Adef = Adef + reshape(z,2,2);
   res = sum(sum((deltaI -mean(mean(deltaI))).^2)); % This is the residual as defined
in "Making good features track better"

   loops = loops + 1;
   if loops > 30
      rep = 0;
   end % if looped more than 10 times stop iterating
   if sum(sum(reshape(z,2,2).^2)) < 0.0001
      rep = 0;
   end % if n smaller than threshold
end % while rep == 1
% end iteration here
%loops
```

Stellenbosch University http://scholar.sun.ac.za

```
function feature_state = assign_points(features,feature_state,ind);
% This function assigns points to vehicles.
% The function inserts the cluster number into the second column of te feature_state
variable
% Note that points can only be assigned once they have been tracked through more than
19 frames

global VEHICLES;
global VEHICLENUMBER;
global OLD_VEHICLES;
useable_points = and(feature_state(:,1) == 1,feature_state(:,3) >= 19);
num_useable_points = sum(useable_points);
sim_matrix = zeros(num_useable_points);
pointer_vektor = zeros(num_useable_points,4); % Kolom 1 = indeks in features,kolom 2 =
initial cluster nr, kolom 3 = vorige cluster nrs,
                                                % kolom 4 = point assigned (1) or not(0),
Kolom 5 = cluster getel in "Vloei"
d_point_vektor = zeros(num_useable_points,2); % Word gebruik in die clustering proses
om boek te hou van clusters
funksiewaarde = zeros(num_useable_points,1);  % Word gebruik om funksiewaardes te skets
(te sien waar die styging beqin)
if num_useable_points > 0
distance = zeros(19,1);
j = 1;
for i = 1:500
   if useable_points(i) == 1
      pointer_vektor(j,1) = i;
      j = j+1;
   end % if point is useable
end % for whole data block
pointer_vektor(:,2) = pointer_vektor(:,1);

for i = 2:num_useable_points
   for j = 1:i-1
      distance = ((features(pointer_vektor(i),ind-19:ind,1) -
features(pointer_vektor(j),ind-19:ind,1)).^2 ...
         + (features(pointer_vektor(i),ind-19:ind,2) - features(pointer_vektor(j),ind-
19:ind,2)).^2).^(0.5);
      sim_matrix(i,j) = var(distance);
   end % for
end % for

full_sim_matrix = sim_matrix;
% initialisasie van die dummy pointer vektor

d_point_vektor(:,1) = pointer_vektor(:,1);   % posisie in "features"
d_point_vektor(:,2) = 1;                     % Hoeveelheid punte in hierdie cluster


for teller = 2:num_useable_points
   ou_d_point_vektor = d_point_vektor;

   %Find lowest value
   kleinste = 1000000;
   p = 0;
   q = 0;
   for i = teller:num_useable_points
      for j = 1:i-teller+1
         if sim_matrix(i,j) < kleinste
            kleinste = sim_matrix(i,j);
            p = min(i,j);
            q = max(i,j);
         end % if
      end % for
   end % for

   % Finished finding lowest number
   p_index = d_point_vektor(p+teller-2,1);
   q_index = d_point_vektor(q,1);
   temp_sim = sim_matrix;

   % Merging clusters p and q
```

Stellenbosch University http://scholar.sun.ac.za

```
    mp = d_point_vektor(p+teller-2,2);
    mq = d_point_vektor(q,2);
    d_point_vektor(p+teller-2,2) = d_point_vektor(p+teller-2,2)+d_point_vektor(q,2);

    mt = d_point_vektor(p+teller-2,2);
    Epq = kleinste;

    for teller2 = q:-1:teller
       temp_sim(teller2,:) = temp_sim(teller2-1,:);
       d_point_vektor(teller2,:) = d_point_vektor(teller2-1,:);
    end % for teller2
    for teller2 = q-teller+2:num_useable_points-teller+1
       temp_sim(:,teller2) = temp_sim(:,teller2+1);
    end % for teller2
    d_point_vektor(teller-1,:) = 0;

    % Update entries in the similarity matrix

    if p~= 1
       for r = 1:p-1
          r_index = ou_d_point_vektor(r+teller-2,1);
          t = 1;
          while r_index ~= ou_d_point_vektor(t,1)
             t = t+1;
          end % while
          mr = ou_d_point_vektor(t,2);

          if r_index > p_index
             Erp = sim_matrix(t,p);
          else
             Erp = sim_matrix(p+teller-2,t-teller+2);
          end % if

          if r_index > q_index
             Erq = sim_matrix(t,q-teller+2);
          else
             Erq = sim_matrix(q,t-teller+2);
          end % if
          temp_sim(p+teller-1,r) = (1/(mr+mt))*((mr+mp)*Erp+(mr+mq)*Erq-mr*Epq); % Die -
1 ipv -2 is omdat kolom reeds uitgevee is
       end % for
    end % if

    for r = teller+p:num_useable_points
       r_index = d_point_vektor(r,1);
       t = 1;
       while r_index ~= ou_d_point_vektor(t,1)
          t = t+1;
       end % while
       mr = ou_d_point_vektor(t,2);
       if r_index > p_index
          Erp = sim_matrix(t,p);
       else
          Erp = sim_matrix(p+teller-2,t-teller+2);
       end % if

       if r_index > q_index
          Erq = sim_matrix(t,q-teller+2);
       else
          Erq = sim_matrix(q,t-teller+2);
       end % if
       temp_sim(r,p) = (1/(mr+mt))*((mr+mp)*Erp+(mr+mq)*Erq-mr*Epq);
    end %for

    sim_matrix =temp_sim;
    funksiewaarde(teller) = funksiewaarde(teller-1)+Epq;
end % for teller
funksiewaarde = funksiewaarde -
linspace(0,funksiewaarde(num_useable_points),num_useable_points)';
minste = 10000;
for i = 1:num_useable_points
    if funksiewaarde(i) < minste
```

Stellenbosch University http://scholar.sun.ac.za

```
      minste = funksiewaarde(i);
      breekpunt = i;
   end % if
end % for

sim_matrix = full_sim_matrix;
% initialisasie van die dummy pointer vektor

d_point_vektor(:,1) = pointer_vektor(:,1);    % posisie in "features"
d_point_vektor(:,2) = 1;                      % Hoeveelheid punte in hierdie cluster


for teller = 2:breekpunt
   ou_d_point_vektor = d_point_vektor;

   %Find lowest value
   kleinste = 1000000;
   p = 0;
   q = 0;
   for i = teller:num_useable_points
      for j = 1:i-teller+1
         if sim_matrix(i,j) < kleinste
            kleinste = sim_matrix(i,j);
            p = min(i,j);
            q = max(i,j);
         end % if
      end % for
   end % for
   %p
   %q

   % Finished finding lowest number
   p_index = d_point_vektor(p+teller-2,1);
   q_index = d_point_vektor(q,1);
   temp_sim = sim_matrix;

   % Merging clusters p and q
   mp = d_point_vektor(p+teller-2,2);
   mq = d_point_vektor(q,2);
   d_point_vektor(p+teller-2,2) = d_point_vektor(p+teller-2,2)+d_point_vektor(q,2);

   % Opdateer die vektor am te wys watter punte aan watter clusters behoort.
   pointer_vektor(:,2) = (pointer_vektor(:,2) - (pointer_vektor(:,2) ==
q_index).*pointer_vektor(:,2)) + (pointer_vektor(:,2) == q_index).*p_index;
   %Klaar pointer_vektor opgedateer

   mt = d_point_vektor(p+teller-2,2);
   Epq = kleinste;

   for teller2 = q:-1:teller
      temp_sim(teller2,:) = temp_sim(teller2-1,:);
      d_point_vektor(teller2,:) = d_point_vektor(teller2-1,:);
   end % for teller2
   for teller2 = q-teller+2:num_useable_points-teller+1
      temp_sim(:,teller2) = temp_sim(:,teller2+1);
   end % for teller2
   d_point_vektor(teller-1,:) = 0;

   % Update entries in the similarity matrix

   if p~= 1
      for r = 1:p-1
         r_index = ou_d_point_vektor(r+teller-2,1);
         t = 1;
         while r_index ~= ou_d_point_vektor(t,1)
            t = t+1;
         end % while
         mr = ou_d_point_vektor(t,2);

         if r_index > p_index
            Erp = sim_matrix(t,p);
         else
```

Stellenbosch University http://scholar.sun.ac.za

```
            Erp = sim_matrix(p+teller-2,t-teller+2);
         end % if

         if r_index > q_index
            Erq = sim_matrix(t,q-teller+2);
         else
            Erq = sim_matrix(q,t-teller+2);
         end % if
         temp_sim(p+teller-1,r) = (1/(mr+mt))*((mr+mp)*Erp+(mr+mq)*Erq-mr*Epq); % Die -
1 ipv -2 is omdat kolom reeds uitgevee is
      end % for
   end % if

   for r = teller+p:num_useable_points
      r_index = d_point_vektor(r,1);
      t = 1;
      while r_index ~= ou_d_point_vektor(t,1)
         t = t+1;
      end % while
      mr = ou_d_point_vektor(t,2);
      if r_index > p_index
         Erp = sim_matrix(t,p);
      else
         Erp = sim_matrix(p+teller-2,t-teller+2);
      end % if

      if r_index > q_index
         Erq = sim_matrix(t,q-teller+2);
      else
         Erq = sim_matrix(q,t-teller+2);
      end % if
      temp_sim(r,p) = (1/(mr+mt))*((mr+mp)*Erp+(mr+mq)*Erq-mr*Epq);
   end %for

   sim_matrix =temp_sim;
   funksiewaarde(teller) = funksiewaarde(teller-1)+Epq;
end % for teller

% Here we collect the points' old vehicle numbers
for i = 1:num_useable_points
   pointer_vektor(i,3) = feature_state(pointer_vektor(i,1),2);
end % for i

VEHICLES = zeros(20,5);
% Next we assign a vehicle number to each cluster
k = 1;
for i = 1:num_useable_points
   if pointer_vektor(i,4) == 0 % if this point has not been assigned a vehicle number
yet
      % Now it should be decided wether to appoint a new vehicle number or keep an
existing one.
      % If the mode of the points' non zero cluster numbers are not zero and has not
been used already this time it is used again.
      potensiele_nr = modus((pointer_vektor(:,2) ==
pointer_vektor(i,2)).*pointer_vektor(:,3));
      if and(potensiele_nr ~=0 , sum(VEHICLES(:,1) == potensiele_nr) == 0 )
         % can now use potensiele_nr again
         pointer_vektor(:,2) = pointer_vektor(:,2) + (pointer_vektor(:,2) ==
pointer_vektor(i,2))*(potensiele_nr -pointer_vektor(i,2));
         VEHICLES(k,1) = potensiele_nr;
         k = k+1;
      else
         pointer_vektor(:,2) = pointer_vektor(:,2) + (pointer_vektor(:,2) ==
pointer_vektor(i,2))*(VEHICLENUMBER -pointer_vektor(i,2));
         VEHICLES(k,1) = VEHICLENUMBER;
         VEHICLENUMBER = VEHICLENUMBER+1;
         if VEHICLENUMBER > 10000
            VEHICLENUMBER = 1;
         end % if
         k = k+1;
      end % if
      pointer_vektor(:,4) = pointer_vektor(:,4) + (pointer_vektor(i,2) ==
```

Stellenbosch University http://scholar.sun.ac.za

```
pointer_vektor(:,2));
   end % if point assigned yet
end % for i

for i = 1:num_useable_points
   feature_state(pointer_vektor(i),2) = pointer_vektor(i,2);
end % for

% Each of the points in feature_state now has a vehicle number associated with it.That
links with the VEHICLES matrix

OLD_VEHICLES = VEHICLES;
end % if number_useable_points > 0
```

Stellenbosch University http://scholar.sun.ac.za

```
function feature_state = assign_points2(features,feature_state,ind,a,normaal,vekt);
% This function assigns points to vehicles.
% The function inserts the cluster number into the second column of te feature_state
variable
% Note that points can only be assigned once they have been tracked through more than
19 frames

global VEHICLES;
global VEHICLENUMBER;
global OLD_VEHICLES;
coord1 = zeros(3,1);
coord2 = zeros(3,1);
useable_points = and(feature_state(:,1) == 1,feature_state(:,3) >= 19);
num_useable_points = sum(useable_points);
sim_matrix = zeros(num_useable_points);
pointer_vektor = zeros(num_useable_points,4); % Kolom 1 = indeks in features,kolom 2 =
initial cluster nr, kolom 3 = vorige cluster nrs,
                                                 % kolom 4 = point assigned (1) or not(0),
Kolom 5 = cluster getel in "Vloei"
d_point_vektor = zeros(num_useable_points,2); % Word gebruik in die clustering proses
om boek te hou van clusters
funksiewaarde = zeros(num_useable_points,1);  % Word gebruik om funksiewaardes te skets
(te sien waar die styging beqin)
if num_useable_points > 0
   distance = zeros(20,1);

j = 1;
for i = 1:500
   if useable_points(i) == 1
      pointer_vektor(j,1) = i;
      j = j+1;
   end % if point is useable
end % for whole data block
pointer_vektor(:,2) = pointer_vektor(:,1);

for i = 2:num_useable_points
   for j = 1:i-1
      for k = 0:19
         coord1 = per_trans([features(pointer_vektor(i),ind-
k,1);features(pointer_vektor(i),ind-k,2)],a,normaal,vekt);
         coord2 = per_trans([features(pointer_vektor(j),ind-
k,1);features(pointer_vektor(j),ind-k,2)],a,normaal,vekt);
         distance(k+1) = ((coord1(1,1) - coord2(1,1)).^2 + (coord1(2,1) -
coord2(2,1)).^2 + (coord1(3,1) - coord2(3,1)).^2).^(0.5);
      end % for k
      sim_matrix(i,j) = var(distance);
   end % for
end % for

full_sim_matrix = sim_matrix;
% initialisasie van die dummy pointer vektor

d_point_vektor(:,1) = pointer_vektor(:,1);    % posisie in "features"
d_point_vektor(:,2) = 1;                      % Hoeveelheid punte in hierdie cluster


for teller = 2:num_useable_points
   ou_d_point_vektor = d_point_vektor;

   %Find lowest value
   kleinste = 1000000;
   p = 0;
   q = 0;
   for i = teller:num_useable_points
      for j = 1:i-teller+1
         if sim_matrix(i,j) < kleinste
            kleinste = sim_matrix(i,j);
            p = min(i,j);
            q = max(i,j);
         end % if
      end % for
   end % for
```

Stellenbosch University http://scholar.sun.ac.za

```
    % Finished finding lowest number
    p_index = d_point_vektor(p+teller-2,1);
    q_index = d_point_vektor(q,1);
    temp_sim = sim_matrix;

    % Merging clusters p and q
    mp = d_point_vektor(p+teller-2,2);
    mq = d_point_vektor(q,2);
    d_point_vektor(p+teller-2,2) = d_point_vektor(p+teller-2,2)+d_point_vektor(q,2);

    mt = d_point_vektor(p+teller-2,2);
    Epq = kleinste;

    for teller2 = q:-1:teller
       temp_sim(teller2,:) = temp_sim(teller2-1,:);
       d_point_vektor(teller2,:) = d_point_vektor(teller2-1,:);
    end % for teller2
    for teller2 = q-teller+2:num_useable_points-teller+1
       temp_sim(:,teller2) = temp_sim(:,teller2+1);
    end % for teller2
    d_point_vektor(teller-1,:) = 0;

    % Update entries in the similarity matrix

    if p~= 1
       for r = 1:p-1
          r_index = ou_d_point_vektor(r+teller-2,1);
          t = 1;
          while r_index ~= ou_d_point_vektor(t,1)
             t = t+1;
          end % while
          mr = ou_d_point_vektor(t,2);

          if r_index > p_index
             Erp = sim_matrix(t,p);
          else
             Erp = sim_matrix(p+teller-2,t-teller+2);
          end % if

          if r_index > q_index
             Erq = sim_matrix(t,q-teller+2);
          else
             Erq = sim_matrix(q,t-teller+2);
          end % if
          temp_sim(p+teller-1,r) = (1/(mr+mt))*((mr+mp)*Erp+(mr+mq)*Erq-mr*Epq); % Die -
1 ipv -2 is omdat kolom reeds uitgevee is
       end % for
    end % if

    for r = teller+p:num_useable_points
       r_index = d_point_vektor(r,1);
       t = 1;
       while r_index ~= ou_d_point_vektor(t,1)
          t = t+1;
       end % while
       mr = ou_d_point_vektor(t,2);
       if r_index > p_index
          Erp = sim_matrix(t,p);
       else
          Erp = sim_matrix(p+teller-2,t-teller+2);
       end % if

       if r_index > q_index
          Erq = sim_matrix(t,q-teller+2);
       else
          Erq = sim_matrix(q,t-teller+2);
       end % if
       temp_sim(r,p) = (1/(mr+mt))*((mr+mp)*Erp+(mr+mq)*Erq-mr*Epq);
    end %for

    sim_matrix =temp_sim;
```

Stellenbosch University http://scholar.sun.ac.za

```
   funksiewaarde(teller) = funksiewaarde(teller-1)+Epq;
end % for teller
funksiewaarde = funksiewaarde -
linspace(0,funksiewaarde(num_useable_points),num_useable_points)';
minste = 10000;
for i = 1:num_useable_points
   if funksiewaarde(i) < minste
      minste = funksiewaarde(i);
      breekpunt = i;
   end % if
end % for

sim_matrix = full_sim_matrix;
% initialisasie van die dummy pointer vektor

d_point_vektor(:,1) = pointer_vektor(:,1);   % posisie in "features"
d_point_vektor(:,2) = 1;                     % Hoeveelheid punte in hierdie cluster


for teller = 2:breekpunt
   ou_d_point_vektor = d_point_vektor;

   %Find lowest value
   kleinste = 1000000;
   p = 0;
   q = 0;
   for i = teller:num_useable_points
      for j = 1:i-teller+1
         if sim_matrix(i,j) < kleinste
            kleinste = sim_matrix(i,j);
            p = min(i,j);
            q = max(i,j);
         end % if
      end % for
   end % for
   %p
   %q

   % Finished finding lowest number
   p_index = d_point_vektor(p+teller-2,1);
   q_index = d_point_vektor(q,1);
   temp_sim = sim_matrix;

   % Merging clusters p and q
   mp = d_point_vektor(p+teller-2,2);
   mq = d_point_vektor(q,2);
   d_point_vektor(p+teller-2,2) = d_point_vektor(p+teller-2,2)+d_point_vektor(q,2);

   % Opdateer die vektor am te wys watter punte aan watter clusters behoort.
   pointer_vektor(:,2) = (pointer_vektor(:,2) - (pointer_vektor(:,2) ==
q_index).*pointer_vektor(:,2)) + (pointer_vektor(:,2) == q_index).*p_index;
   %Klaar pointer_vektor opgedateer

   mt = d_point_vektor(p+teller-2,2);
   Epq = kleinste;

   for teller2 = q:-1:teller
      temp_sim(teller2,:) = temp_sim(teller2-1,:);
      d_point_vektor(teller2,:) = d_point_vektor(teller2-1,:);
   end % for teller2
   for teller2 = q-teller+2:num_useable_points-teller+1
      temp_sim(:,teller2) = temp_sim(:,teller2+1);
   end % for teller2
   d_point_vektor(teller-1,:) = 0;

   % Update entries in the similarity matrix

   if p~= 1
      for r = 1:p-1
         r_index = ou_d_point_vektor(r+teller-2,1);
         t = 1;
         while r_index ~= ou_d_point_vektor(t,1)
```

Stellenbosch University http://scholar.sun.ac.za

```
            t = t+1;
         end % while
         mr = ou_d_point_vektor(t,2);

         if r_index > p_index
            Erp = sim_matrix(t,p);
         else
            Erp = sim_matrix(p+teller-2,t-teller+2);
         end % if

         if r_index > q_index
            Erq = sim_matrix(t,q-teller+2);
         else
            Erq = sim_matrix(q,t-teller+2);
         end % if
         temp_sim(p+teller-1,r) = (1/(mr+mt))*((mr+mp)*Erp+(mr+mq)*Erq-mr*Epq); % Die -
1 ipv -2 is omdat kolom reeds uitgevee is
      end % for
   end % if

   for r = teller+p:num_useable_points
      r_index = d_point_vektor(r,1);
      t = 1;
      while r_index ~= ou_d_point_vektor(t,1)
         t = t+1;
      end % while
      mr = ou_d_point_vektor(t,2);
      if r_index > p_index
         Erp = sim_matrix(t,p);
      else
         Erp = sim_matrix(p+teller-2,t-teller+2);
      end % if

      if r_index > q_index
         Erq = sim_matrix(t,q-teller+2);
      else
         Erq = sim_matrix(q,t-teller+2);
      end % if
      temp_sim(r,p) = (1/(mr+mt))*((mr+mp)*Erp+(mr+mq)*Erq-mr*Epq);
   end %for

   sim_matrix =temp_sim;
   funksiewaarde(teller) = funksiewaarde(teller-1)+Epq;
end % for teller

% Here we collect the points' old vehicle numbers
for i = 1:num_useable_points
   pointer_vektor(i,3) = feature_state(pointer_vektor(i,1),2);
end % for i

VEHICLES = zeros(20,5);
% Next we assign a vehicle number to each cluster
k = 1;
for i = 1:num_useable_points
   if pointer_vektor(i,4) == 0 % if this point has not been assigned a vehicle number
yet
      % Now it should be decided wether to appoint a new vehicle number or keep an
existing one.
      % If the mode of the points' non zero cluster numbers are not zero and has not
been used already this time it is used again.
      potensiele_nr = modus((pointer_vektor(:,2) ==
pointer_vektor(i,2)).*pointer_vektor(:,3));
      if and(potensiele_nr ~=0 , sum(VEHICLES(:,1) == potensiele_nr) == 0 )
         % can now use potensiele_nr again
         pointer_vektor(:,2) = pointer_vektor(:,2) + (pointer_vektor(:,2) ==
pointer_vektor(i,2))*(potensiele_nr -pointer_vektor(i,2));
         VEHICLES(k,1) = potensiele_nr;
         k = k+1;
      else
         pointer_vektor(:,2) = pointer_vektor(:,2) + (pointer_vektor(:,2) ==
pointer_vektor(i,2))*(VEHICLENUMBER -pointer_vektor(i,2));
         VEHICLES(k,1) = VEHICLENUMBER;
```

Stellenbosch University http://scholar.sun.ac.za

```
         VEHICLENUMBER = VEHICLENUMBER+1;
         if VEHICLENUMBER > 10000
            VEHICLENUMBER = 1;
         end % if
         k = k+1;
      end % if
      pointer_vektor(:,4) = pointer_vektor(:,4) + (pointer_vektor(i,2) ==
pointer_vektor(:,2));
   end % if point assigned yet
end % for i

for i = 1:num_useable_points
   feature_state(pointer_vektor(i),2) = pointer_vektor(i,2);
end % for

% Each of the points in feature_state now has a vehicle number associated with it.That
links with the VEHICLES matrix

OLD_VEHICLES = VEHICLES;
end % if number_useable_points > 0
```

Stellenbosch University http://scholar.sun.ac.za

```
cd c:\mydocu~1\tesis2~1\durban~2;
A = rgb2gray(imread('frame0500.jpeg','jpeg'));  % Die begin beeld sien ook lyn 6
s = size(A);
D = zeros(s(1),s(2),20); % Die "20" verwys na die hoeveelheid beelde gebruik. Sien ook
lyne 5,16
for k = 1:20
   B = rgb2gray(imread(['frame0' int2str(20*k+500) '.jpeg'],'jpeg')); % Die "5*k"
verways na die interval tussen beelde, en die 100 na die eerste beeld nr.
   D(:,:,k) = double(B);
   k
end

ou_aantal = zeros(s(1),s(2));
modus = zeros(s(1),s(2));
for l = 5:250
   tydelik = zeros(s(1),s(2),20);
   aantal = zeros(s(1),s(2));
   for m = 1:20
      tydelik(:,:,m) = and((l+5)>D(:,:,m),D(:,:,m)>(l-5));
      aantal = aantal + tydelik(:,:,m);
   end % for m
   modus = modus +(aantal >= ou_aantal).*(l-modus);
   ou_aantal = ou_aantal + (aantal > ou_aantal).*(aantal-ou_aantal);
   l
end % for l
modus = uint8(modus);
imshow(modus);
imwrite(modus,'agtergrond.jpg','jpg');
```

```
function out = funksie(a)
```
Stellenbosch University http://scholar.sun.ac.za

```
Vx = -103.5;      % x komponent van vektor V
Vy = 14.5;      % Y komponent van vektor V
Wx = -15.5;     % x komponent van vektor W
Wy = -101.5;
Lv = 202.548;  % Afstand van oorsprong na punt op wereldvlak
Lw = 36.253;
b  = 168.082;

out = (Vx*Lv/sqrt(Vx^2 + Vy^2 + a^2)-Wx*Lw/sqrt(Wx^2 + Wy^2 + a^2))^2 +
(Vy*Lv/sqrt(Vx^2 + Vy^2 + a^2)-Wy*Lw/sqrt(Wx^2 + Wy^2 + a^2))^2 + (a*Lv/sqrt(Vx^2 +
Vy^2 + a^2)-a*Lw/sqrt(Wx^2 + Wy^2 + a^2))^2 - b^2;
```

Stellenbosch University http://scholar.sun.ac.za

```
function f_im = getfeat(im,threshold,w)

% This function selects trackable features from an image by using minimum eigenvalues
% See Paper by Jean Yves Bouguet p 8
% im is the input image (must be grayscale)
% Threshold is the percentage of the maximum of the smaller of the two eigenvalues of G
% for which the feature is kept.
% 2*w+1 is the windowsize that is being used for the Calculation of G

% features is a M*2 matrix where the first column is the x value, the second the y
value

Ix = sobel(im,'x');
I2y = sobel(im,'y').^2;
Ixy = sobel(Ix,'y');
I2x = Ix.^2;

s = size(im);
D = zeros(s(1),s(2));
for i = w+1:s(1)-w
   for j= w+1:s(2)-w
      G = [sum(sum(I2x(i-w:i+w,j-w:j+w))) sum(sum(Ixy(i-w:i+w,j-w:j+w)));sum(sum(Ixy(i-
w:i+w,j-w:j+w))) sum(sum(I2y(i-w:i+w,j-w:j+w)))];
      D(i,j) = min(eig(G));
   end
end

D = (D > threshold*max(max(D))).*D;
features = zeros(200,2);
count = 0;
for i = w+1:s(1)-w
   for j= w+1:s(2)-w
      if D(i,j) ~= 0
         if D(i,j) ~= max(max(D(i-3:i+3,j-3:j+3))) % This eliminates 2 local maximums
within the span of 3*3 pixels
            D(i,j) = 0;                          % otherwise stated two local max has
at least 3 pixels in between them
         else
            count = count+1;
            features(count,:) = [i,j];
         end
      end
   end
end
%features =features(1:count,:);
f_im = D >0;
text = 'features selected'
```

Stellenbosch University http://scholar.sun.ac.za

```
function [s, status] = getsubvalue(point,M);

% This function interpolates matrix M bilinearly in order to find a value
% of point which does not have to be integer
% point is defined as [row;column] and denotes the positin in M which has to be
interpolated.
% An error message is returned if the point is out of the Matrix dimensions
% status is used to notify the calling procedure that the value cannot be extracted.
% status is 0 if matrix dimensions are exceeded
status = 1;
m = size(M);
if ((((ceil(point(1,1)) >= m(1))|(floor(point(1,1)) <= 1))|(ceil(point(2,1)) >=
m(2)))|(floor(point(2,1)) <= 1))
   status = 0;
   s = 0;
   %text = (['point exceeds matrix dimensions in function getsubvalue ('
num2str(transpose(point)) ')'])
   return
end % if

x = floor(point(1,1));
ax = point(1,1) - x;

y = floor(point(2,1));
ay = point(2,1) - y;

s = (1-ax)*(1-ay)*double(M(x,y)) + ax*(1-ay)*double(M(x+1,y)) + (1-
ax)*ay*double(M(x,y+1)) + ax*ay*double(M(x+1,y+1));
```

Stellenbosch University http://scholar.sun.ac.za

```
% File Long_sequence_track

global VEHICLENUMBER
VEHICLENUMBER = 1;          % This number in incremented so that each vehicle is
assigned a new number
                            % It should repeat once it reaches the largest uint16 number
global VEHICLES;
global OLD_VEHICLES;
VEHICLES = zeros(20,5);     % This matrix contains the vehicle number,centroid(x,y
coordinates), speed, and wether it has been counted in flow.


features = zeros(500,1500+1,3);  % level 1 =x, level = y, level = residues
feature_state = zeros(500,3);    % column 1: 0 = no feature, 1 = tracking , 2 = out of
bounds , 3 lost (residu too high)
                                 % column 2 is the vehicle number
                                 % column 3 is the number of frames through which the
point has been tracked


Vloei = 0;

comp_thresh = 20;           % This threshold is used to decide whether to keep existing
pixel value or not when comparing images
num_frames = 400;           % number of frames through which the process will go
w = [5;5];                  % tracking windowsize
L = 3;                      % Number of levels in Pyramid
cd c:\mydocu~1\tesis2~1\n1bill~1
agtergrond = imread('agtergrond.jpg','jpg');
road = imread('travelled_way.jpg','jpg')>125;
kleure = jet(12);
h = 1;
figure(h);
ref_im = rgb2gray(imread('car~1000.jpe','jpg'));
vergelyk = uint8((double(ref_im)-double(agtergrond)) > comp_thresh);
imshow(vergelyk)
vergelyk = medfilt2(vergelyk);
pause
vergelyk = and(road,vergelyk);
imshow(vergelyk)
feat_im = getfeat(ref_im,0.1,3);
feat_im = and(vergelyk,feat_im);
s = size(ref_im);
%volg_lyne = zeros(s(1),s(2));
output_im = uint8(zeros(s(1),s(2),3));
count = 0;
for i = 1:s(1)
   for j = 1:s(2)
      if feat_im(i,j) ~= 0
         count = count+1;
         features(count,1,1) = i;
         features(count,1,2) = j;
         feature_state(count,1) = 1;
      end % if
   end
end

text = ['finished counting features' int2str(count)]
last_im = ref_im;
ref_index = 1;
reset(h);


for i = 1:num_frames
   next_im = rgb2gray(imread(['car~' int2str(1000+i) '.jpe'],'jpg' ));
   %i
   points = zeros(500,2);
   points(1:count,1) = features(1:count,i,1);
   points(1:count,2) = features(1:count,i,2);
   [v,feature_state] = Pyramid(last_im,next_im,L,w,points,feature_state);

   features(1:count,i+1,1) = features(1:count,i,1)+v(1:count,1);
```

Stellenbosch University http://scholar.sun.ac.za

```
   features(1:count,i+1,2) = features(1:count,i,2)+v(1:count,2);
   for j = 1:count
      if feature_state(j,1) == 1
         [features(j,i+1,3),status] =
affine_def(ref_im,next_im,[5;5],[features(j,ref_index,1);features(j,ref_index,2)],[feat
ures(j,i+1,1)-features(j,ref_index,1);features(j,i+1,2)-
features(j,ref_index,2)]);%ref_im,im,w,point,d
         if status == 0;
            feature_state(j,1) = 2;
         end % if feature too close to image boundary

         % this section is used for drawing lines in the image "volg_lyne"
         %slope = (round(features(j,i+1,2))-
round(features(j,i,2)))/(round(features(j,i+1,1))-round(features(j,i,1))+0.001);
         %if slope >= 1
         %   for n =
min([round(features(j,i,2)),round(features(j,i+1,2))]):max([round(features(j,i,2)),roun
d(features(j,i+1,2))])
         %      m = round(features(j,i,1))+round((n-
round(features(j,i,2)))*(1/(slope+0.001)));
         %      volg_lyne(m,n) = 255;
         %   end
         %else
         %   for m =
min([round(features(j,i,1)),round(features(j,i+1,1))]):max([round(features(j,i,1)),roun
d(features(j,i+1,1))])
         %      n = round(features(j,i,2))+round((m-round(features(j,i,1)))*slope);
         %      volg_lyne(m,n) = 255;
         %   end % for
         %end % if
         % End of line drawing section

      %j
      end % if feature state is tracking
   end % for number of features]

   if i-20 >= 0
      feature_state(:,1) = check_res(features(1:count,i-18:i+1,3),feature_state(:,1));
   end % if more than 20 residues to check

   pause(0.1);

   % Update data block here

   if mod(i,10) == 0

      ref_im = next_im;
      ref_index = i+1;
      %volg_lyne = zeros(s(1),s(2));
      last_used_row = 500;
      for row = 1:500
         if and(feature_state(row,1) ~= 1,row < last_used_row)
            while and(feature_state(last_used_row,1) ~= 1 , last_used_row > row)
               if feature_state(last_used_row,1) ~= 0
                  features(last_used_row,:,:) = 0;
                  feature_state(last_used_row,:) = 0;
               end % if feature_state not 0 (also not 1)
               last_used_row = last_used_row - 1;
            end
            features(row,:,:) = features(last_used_row,:,:);
            feature_state(row,:) = feature_state(last_used_row,:);
            features(last_used_row,:,:) = 0;
            feature_state(last_used_row,:) = 0;
            last_used_row = last_used_row - 1;
         end % if
      end % for row 1 to 500
      if and(feature_state(last_used_row,1) ~= 1,last_used_row >0)
         features(last_used_row,:,:) = 0;
         feature_state(last_used_row,:) = 0;
      end % if

      % Hier word die toekenning van punte aan motors gedoen
```

Stellenbosch University http://scholar.sun.ac.za

```
      if i > 15
         feature_state = assign_points(features,feature_state,i+1);
      end % if


      % Adding new features
      feat_im = getfeat(ref_im,0.1,3);
      vergelyk = uint8((double(next_im)-double(agtergrond)) > comp_thresh);
      vergelyk = medfilt2(vergelyk);
      %vergelyk = bwmorph(vergelyk,'erode',1);
      %vergelyk = bwmorph(vergelyk,'dilate',2);
      vergelyk = and(road,vergelyk);

      num_existing_features = sum(feature_state(:,1) == 1);
      % Black out squares where threre is already features - so that features don't
heap up.
      for tel = 1:num_existing_features
         vergelyk(round(features(tel,i+1,1))-
1:round(features(tel,i+1,1))+1,round(features(tel,i+1,2))-
1:round(features(tel,i+1,2))+1) = 0;
      end % for

      feat_im = and(vergelyk,feat_im);

      count = num_existing_features;;

      for l = 1:s(1)
         for j = 1:s(2)
            if feat_im(l,j) ~= 0
               count = count+1;
               features(count,i+1,1) = l;
               features(count,i+1,2) = j;
               feature_state(count,1) = 1;
            end % if
         end
      end
      % Finished adding new features
      text = ['finished counting features' int2str(count)]

      % Speed, flow and density to be calculated next
      Digtheid = 0;
      amount_of_veh = sum(VEHICLES(:,1)~=0);
      for veh = 1:amount_of_veh
         [VEHICLES(veh,2),VEHICLES(veh,3)] =
getcentroids(features,feature_state,i+1,VEHICLES(veh,1));
         [old_x,old_y] = getcentroids(features,feature_state,i-18,VEHICLES(veh,1));
         % Calculate incremental density
         if and(VEHICLES(veh,2) > 50,VEHICLES(veh,2) < 210)
            Digtheid = Digtheid +1;
         end % if

         % Calculate incremental flow and speed of vehicle that crossed the line.
         if and(VEHICLES(veh,2) > 150 , old_x < 150)
            if sum((OLD_VEHICLES(:,1) == VEHICLES(veh,1)).*OLD_VEHICLES(:,5)) == 0
               Vloei = Vloei + 1;
               VEHICLES(veh,5) = 1;
            end % if
            Spoed = (25/19)*sqrt((old_x - VEHICLES(veh,2))^2 + (old_y -
VEHICLES(veh,3))^2) % pixels per sekonde
            VEHICLES(veh,4) = Spoed;
         end % if

      end % for all vehicles in register VEHICLES
      Digtheid
      VEHICLES
      OLD_VEHICLES = VEHICLES;

   end % if tenth frame
   % Finished Updating data block
   last_im = next_im;

   output_im(:,:,1) = next_im;
```

Stellenbosch University http://scholar.sun.ac.za

```
    output_im(:,:,2) = next_im;
    output_im(:,:,3) = next_im;
    for l = 1:500
       if and(feature_state(l,2) ~= 0,feature_state(l,1) == 1)
          output_im(round(features(l,i+1,1))-
1:round(features(l,i+1,1))+1,round(features(l,i+1,2))-1:round(features(l,i+1,2))+1,1) =
kleure(mod(feature_state(l,2),12)+1,1)*255;
          output_im(round(features(l,i+1,1))-
1:round(features(l,i+1,1))+1,round(features(l,i+1,2))-1:round(features(l,i+1,2))+1,2) =
kleure(mod(feature_state(l,2),12)+1,2)*255;
          output_im(round(features(l,i+1,1))-
1:round(features(l,i+1,1))+1,round(features(l,i+1,2))-1:round(features(l,i+1,2))+1,3) =
kleure(mod(feature_state(l,2),12)+1,3)*255;
       end % if
    end % for l
    output_im(50,:,:) =0;
    output_im(150,:,:) = 0;
    output_im(210,:,:) = 0;
    output_im = uint8(output_im);
    imshow(output_im);
    pause(0.1)
    imwrite(output_im,['out' int2str(1000+i) '.jpg'],'jpg');
end % for i = number of frames
```

Vloei

Stellenbosch University http://scholar.sun.ac.za

```
% File Long_sequence_track2

global VEHICLENUMBER
VEHICLENUMBER = 1;          % This number in incremented so that each vehicle is
assigned a new number
                            % It should repeat once it reaches the largest uint16 number
global VEHICLES;
global OLD_VEHICLES;
VEHICLES = zeros(20,5);     % This matrix contains the vehicle number,centroid(x,y
coordinates), speed, and wether it has been counted in flow.


features = zeros(500,1500+1,3);  % level 1 =x, level = y, level = residues
feature_state = zeros(500,3);    % column 1: 0 = no feature, 1 = tracking , 2 = out of
bounds , 3 lost (residu too high)
                                 % column 2 is the vehicle number
                                 % column 3 is the number of frames through which the
point has been tracked
[a,normaal,vekt] = perspektief2;

Vloei = 0;

comp_thresh = 15;           % This threshold is used to decide whether to keep existing
pixel value or not when comparing images
num_frames = 400;           % number of frames through which the process will go
w = [5;5];                  % tracking windowsize
L = 1;                      % Number of levels in Pyramid
cd c:\mydocu~1\tesis2~1\n1bill~1
agtergrond = imread('agtergrond.jpg','jpg');
road = imread('travelled_way.jpg','jpg')>125;
kleure = jet(12);
h = 1;
figure(h);
ref_im = rgb2gray(imread('cpic0100.jpg','jpg'));
vergelyk = uint8((double(ref_im)-double(agtergrond)) > comp_thresh);
imshow(vergelyk)
vergelyk = medfilt2(vergelyk);
pause
vergelyk = and(road,vergelyk);
imshow(vergelyk)
pause
feat_im = getfeat(ref_im,0.1,3);
feat_im = and(vergelyk,feat_im);
s = size(ref_im);
%volg_lyne = zeros(s(1),s(2));
output_im = uint8(zeros(s(1),s(2),3));
count = 0;
for i = 1:s(1)
   for j = 1:s(2)
      if feat_im(i,j) ~= 0
         count = count+1;
         features(count,1,1) = i;
         features(count,1,2) = j;
         feature_state(count,1) = 1;
      end % if
   end
end

text = ['finished counting features' int2str(count)]
last_im = ref_im;
ref_index = 1;
reset(h);


for i = 1:num_frames
   next_im = rgb2gray(imread(['cpic0' int2str(100+i) '.jpg'],'jpg' ));
   %i
   points = zeros(500,2);
   points(1:count,1) = features(1:count,i,1);
   points(1:count,2) = features(1:count,i,2);
   [v,feature_state] = Pyramid(last_im,next_im,L,w,points,feature_state);
```

Stellenbosch University http://scholar.sun.ac.za

```
   features(1:count,i+1,1) = features(1:count,i,1)+v(1:count,1);
   features(1:count,i+1,2) = features(1:count,i,2)+v(1:count,2);
   for j = 1:count
      if feature_state(j,1) == 1
         [features(j,i+1,3),status] =
affine_def(ref_im,next_im,[5;5],[features(j,ref_index,1);features(j,ref_index,2)],[feat
ures(j,i+1,1)-features(j,ref_index,1);features(j,i+1,2)-
features(j,ref_index,2)]);%ref_im,im,w,point,d
         if status == 0;
            feature_state(j,1) = 2;
         end % if feature too close to image boundary

         % this section is used for drawing lines in the image "volg_lyne"
         %slope = (round(features(j,i+1,2))-
round(features(j,i,2)))/(round(features(j,i+1,1))-round(features(j,i,1))+0.001);
         %if slope >= 1
         %   for n =
min([round(features(j,i,2)),round(features(j,i+1,2))]):max([round(features(j,i,2)),roun
d(features(j,i+1,2))])
         %      m = round(features(j,i,1))+round((n-
round(features(j,i,2)))*(1/(slope+0.001)));
         %      volg_lyne(m,n) = 255;
         %   end
         %else
         %   for m =
min([round(features(j,i,1)),round(features(j,i+1,1))]):max([round(features(j,i,1)),roun
d(features(j,i+1,1))])
         %      n = round(features(j,i,2))+round((m-round(features(j,i,1)))*slope);
         %      volg_lyne(m,n) = 255;
         %   end % for
         %end % if
         % End of line drawing section

      %j
      end % if feature state is tracking
   end % for number of features]

   if i-20 >= 0
      feature_state(:,1) = check_res(features(1:count,i-18:i+1,3),feature_state(:,1));
   end % if more than 20 residues to check

   %imshow(uint8(double(next_im) - volg_lyne));
   pause(0.1);

   % Update data block here

   if mod(i,10) == 0

      ref_im = next_im;
      ref_index = i+1;
      %volg_lyne = zeros(s(1),s(2));
      last_used_row = 500;
      for row = 1:500
         if and(feature_state(row,1) ~= 1,row < last_used_row)
            while and(feature_state(last_used_row,1) ~= 1 , last_used_row > row)
               if feature_state(last_used_row,1) ~= 0
                  features(last_used_row,:,:) = 0;
                  feature_state(last_used_row,:) = 0;
               end % if feature_state not 0 (also not 1)
               last_used_row = last_used_row - 1;
            end
            features(row,:,:) = features(last_used_row,:,:);
            feature_state(row,:) = feature_state(last_used_row,:);
            features(last_used_row,:,:) = 0;
            feature_state(last_used_row,:) = 0;
            last_used_row = last_used_row - 1;
         end % if
      end % for row 1 to 500
      if and(feature_state(last_used_row,1) ~= 1,last_used_row >0)
         features(last_used_row,:,:) = 0;
         feature_state(last_used_row,:) = 0;
      end % if
```

Stellenbosch University http://scholar.sun.ac.za

```
      % Hier word die toekenning van punte aan motors gedoen
      if i > 15
         feature_state = assign_points2(features,feature_state,i+1,a,normaal,vekt);
      end % if


      % Adding new features
      feat_im = getfeat(ref_im,0.1,3);
      vergelyk = uint8((double(next_im)-double(agtergrond)) > comp_thresh);
      vergelyk = medfilt2(vergelyk);
      %vergelyk = bwmorph(vergelyk,'erode',1);
      %vergelyk = bwmorph(vergelyk,'dilate',1);
      vergelyk = and(road,vergelyk);

      num_existing_features = sum(feature_state(:,1) == 1);
      % Black out squares where threre is already features - so that features don't
heap up.
      for tel = 1:num_existing_features
         vergelyk(round(features(tel,i+1,1))-
1:round(features(tel,i+1,1))+1,round(features(tel,i+1,2))-
1:round(features(tel,i+1,2))+1) = 0;
      end % for

      feat_im = and(vergelyk,feat_im);

      count = num_existing_features;;

      for l = 1:s(1)
         for j = 1:s(2)
            if feat_im(l,j) ~= 0
               count = count+1;
               features(count,i+1,1) = l;
               features(count,i+1,2) = j;
               feature_state(count,1) = 1;
            end % if
         end
      end
      % Finished adding new features
      text = ['finished counting features' int2str(count)]

      % Speed, flow and density to be calculated next
      Digtheid = 0;
      amount_of_veh = sum(VEHICLES(:,1)~=0);
      for veh = 1:amount_of_veh
         [VEHICLES(veh,2),VEHICLES(veh,3)] =
getcentroids(features,feature_state,i+1,VEHICLES(veh,1));
         [old_x,old_y] = getcentroids(features,feature_state,i-18,VEHICLES(veh,1));
         % Calculate incremental density
         if and(VEHICLES(veh,2) > 50,VEHICLES(veh,2) < 210)
            Digtheid = Digtheid +1;
         end % if

         % Calculate incremental flow and speed of vehicle that crossed the line.
         if and(VEHICLES(veh,2) > 150 , old_x < 150)
            if sum((OLD_VEHICLES(:,1) == VEHICLES(veh,1)).*OLD_VEHICLES(:,5)) == 0
               Vloei = Vloei + 1;
               VEHICLES(veh,5) = 1;
            end % if
            Spoed = (25/19)*sqrt((old_x - VEHICLES(veh,2))^2 + (old_y -
VEHICLES(veh,3))^2) % pixels per sekonde
            VEHICLES(veh,4) = Spoed;
         end % if

      end % for all vehicles in register VEHICLES
      Digtheid
      VEHICLES
      OLD_VEHICLES = VEHICLES;

   end % if tenth frame
   % Finished Updating data block
   last_im = next_im;
```

Stellenbosch University http://scholar.sun.ac.za

```
    output_im(:,:,1) = next_im;
    output_im(:,:,2) = next_im;
    output_im(:,:,3) = next_im;
    for l = 1:500
       if and(feature_state(l,2) ~= 0,feature_state(l,1) == 1)
          output_im(round(features(l,i+1,1))-
1:round(features(l,i+1,1))+1,round(features(l,i+1,2))-1:round(features(l,i+1,2))+1,1) =
kleure(mod(feature_state(l,2),12)+1,1)*255;
          output_im(round(features(l,i+1,1))-
1:round(features(l,i+1,1))+1,round(features(l,i+1,2))-1:round(features(l,i+1,2))+1,2) =
kleure(mod(feature_state(l,2),12)+1,2)*255;
          output_im(round(features(l,i+1,1))-
1:round(features(l,i+1,1))+1,round(features(l,i+1,2))-1:round(features(l,i+1,2))+1,3) =
kleure(mod(feature_state(l,2),12)+1,3)*255;
       end % if
    end % for l
    output_im(50,:,:) =0;
    output_im(150,:,:) = 0;
    output_im(210,:,:) = 0;
    output_im = uint8(output_im);
    imshow(output_im);
    pause(0.1)
    imwrite(output_im,['p_out' int2str(100+i) '.jpg'],'jpg');
end % for i = number of frames
```

Vloei

Stellenbosch University http://scholar.sun.ac.za

```
function modu = modus(vektor);

% Hierdie funksie bepaal die nie-nul getal wat die meeste voorkom in 'n gegewe matriks
% Met 'n maksimum van 100 individuele getalle
% As modus = 0 gegee word beteken dit daar was geen nie-nul getal in die vektor nie.
modu = 0;
s = size(vektor);
tydelik = zeros(100,2);
k = 1;
for i = 1:s(1)
   for j = 1:s(2)
      if vektor(i,j) ~= 0
         if sum(sum(tydelik(:,1) == vektor(i,j))) == 0
            tydelik(k,1) = vektor(i,j);
            tydelik(k,2) = 1;
            k = k+1;
         else
            tydelik(:,2) = tydelik(:,2) + (tydelik(:,1) == vektor(i,j));
         end
      end
   end
end

i = 1;
k = 0;
while tydelik(i,1) ~= 0
   if tydelik(i,2) > k;
      k = tydelik(i,2);
      modu = tydelik(i,1);
   end
   i = i+1;
end
```

Stellenbosch University http://scholar.sun.ac.za

```
function [a,n,v] = perspektief2;
% This function calculates the true coordinates of features given the image coordinates
% Onthou dat as waardes hier verander word moet dit ook in funksie verander word

Vx = -103.5;       % x komponent van vektor V
Vy = 14.5;      % Y komponent van vektor V
Wx = -15.5;     % x komponent van vektor W
Wy = -101.5;
Lv = 202.548;  % Afstand van oorsprong na punt op wereldvlak
Lw = 36.253;
b  = 168.082;
Tx = -56.5;
Ty = 75.5;
Lt = 53.038;

a = fzero('funksie',439);

v = (Lv/sqrt(Vx^2 + Vy^2 + a^2))*[Vx;Vy;a];
w = (Lw/sqrt(Wx^2 + Wy^2 + a^2))*[Wx;Wy;a];
t = (Lt/sqrt(Tx^2 + Ty^2 + a^2))*[Tx;Ty;a];

n = cross((v-t),(w-t));
% vergelyking van vlak is nou n'*(r-v)
% lyn is uit oorsprong dus vgl is r = l*c waar l 'n veranderlike is en c die rigting

% as beeld koordinaat nou f = [g;h;a] is, is die snypunt met die vlak
% s = ((n'*v)/(n'*f))*f
```

Stellenbosch University http://scholar.sun.ac.za

```
function [v,feature_state] = Pyramid(I0,J0,L,w,points,feature_state);

% This function implements the pyramidal Lucas Kanade feature tracker
% imI and imJ are both grayscale images of the same size
% L is the number of levels used for the pyramid (integer)
% 2*w + 1 is the size of the window and w is a 2,1 matrix
% points is the points on imI that is to be found on imJ (array of points 2*n)
% if status is set to 0 from the getsubvalue (in sub_pix_lucas_kanade) routine the
feature is too close to the image perimeter

s = size(I0);
status = 1;
si = size(points);
if L == 0
   v = zeros(si(1),si(2));
   for i = 1:si(1)
      if feature_state(i,1) == 1
         point = points(i,:)';
         [d,status] = sub_pix_Lucas_Kanade(I0,J0,w,point,[0;0]);
         if status == 0
            feature_state(i,1) = 2;
         else
            v(i,:) = d';
            feature_state(i,3) = feature_state(i,3)+1;
         end % if status

      end % if feature_state == 1
   end % for i



else % if L > 0
I = zeros(s(1)+1,s(2)+1,L+1);
J = zeros(s(1)+1,s(2)+1,L+1);
for i = 1:s(1)
   for j = 1:s(2)
      I(i,j,1) = I0(i,j);
      J(i,j,1) = J0(i,j);
   end
end
temp = [0.0625 0.125 0.0625;0.125 0.25 0.125;0.0625 0.125 0.0625];
for k = 1:L
   % Sit hier randwaardes in (sien Jean Bouguet p2)
   I(:,floor(s(2)/2^(k-1))+1,k) = I(:,floor(s(2)/(2^(k-1))),k);
   I(floor(s(1)/2^(k-1))+1,:,k) = I(floor(s(1)/(2^(k-1))),:,k);

   J(:,floor(s(2)/2^(k-1))+1,k) = J(:,floor(s(2)/(2^(k-1))),k);
   J(floor(s(1)/2^(k-1))+1,:,k) = J(floor(s(1)/(2^(k-1))),:,k);
   % Randwaardes nou voltooi
   for i = 1:floor(s(1)/(2^k))
      for j = 1:floor(s(2)/(2^k))
         I(i,j,k+1) = sum(sum(temp.*I(2*i-1:2*i+1,2*j-1:2*j+1,k)));
         J(i,j,k+1) = sum(sum(temp.*J(2*i-1:2*i+1,2*j-1:2*j+1,k)));
      end % for
   end % for
end % for
%Finished building pyramid

g = zeros(2,L+2); % initial guess = [0;0]

v = zeros(si(1),si(2));
for i = 1:si(1)
   if feature_state(i,1) == 1
      point = points(i,:)';
      for l = L+1:-1:1
         u = point./(2^(l-1));
         [d,status] = sub_pix_Lucas_Kanade(I(1:floor(s(1)/(2^(l-
1))),1:floor(s(2)/(2^(l-1))),l),...
            J(1:floor(s(1)/(2^(l-1))),1:floor(s(2)/(2^(l-1))),l),w,u,g(:,l+1));
         if status == 0
            feature_state(i,1) = 2;
            break % terminates the for loop
```

Stellenbosch University http://scholar.sun.ac.za

```
         end
         g(:,l) = 2*(g(:,l+1)+d);
      end % for l

      %new_point =point + g(:,2)+d;
      if status == 1
         v(i,:) = (g(:,2)+d)';
         feature_state(i,3) = feature_state(i,3)+1;
      end % if

   % u+d is the location point of "point" on image J
   end % if feature state is tracking
end % for i = 1:number of points
end
```

Stellenbosch University http://scholar.sun.ac.za

```
function iout = sobel(iin,direction)
% This function performs the sobel operator on an image in either the x or y
directions.
% See page 419 Gonzales/Woods for details of Sobel operator
% iin is a grayscale image and direction is either 'x' or 'y' for the direction of the
partial derivatives.



s = size(iin);
iout = zeros(s(1),s(2));
if direction == 'x'
   op = [-1 -2 -1;0 0 0;1 2 1];
elseif direction == 'y'
   op = [-1 0 1;-2 0 2;-1 0 1];
else
   text = 'direction must be either x or y in quotes'
end % if direction
for i = 2:(s(1)-1)
   for j = 2:(s(2) -1)
      iout(i,j) = 0.125*sum(sum(op.*double(iin(i-1:i+1,j-1:j+1))));
   end % for j
end % for i
```

```
function [v,status] = Sub_pix_Lucas_Kanade(imI,imJ,w,point,g)
```

Stellenbosch University http://scholar.sun.ac.za

```
% This function implements the Lucas_Kanade optical flow algorithm with sub_pixel
accuracy
% See the paper by Jean Yves Bouguet
% imI and imJ are the first and second images respectively (grayscale images)
% w is a 1*2 matrix and specifies the window horizontal windowsize as (x+w(1,1),x-
w(1,1)
% and the vertical window size as y +w(1,2),y-w(1,2) and must be integer
% point is the coordinate around which will be searched for the feature

% Note that the case of a pixel too close to the perimeter of the picture has not been
handled yet.
% if status is set to 0 from the getsubvalue routine the feature is too close to the
image perimeter

status = 1;
v = [0;0];
loops = 0;
rep = 1;
% Extract subpixel values for A
A = zeros(2*w(1,1)+3,2*w(2,1)+3);
ry = 1;
kolom =1;
for i = point(1,1)-w(1,1)-1 :1: point(1,1)+w(1,1)+1
   for j = point(2,1)-w(2,1)-1 :1: point(2,1)+w(2,1)+1
      [A(ry,kolom),status] = getsubvalue([i;j],imI);
      if status == 0
         return
      end % if status == 0
      kolom = kolom+1;
   end % for j
   ry = ry +1;
   kolom = 1;
end % for i
% Finished extracting matrix A
Ix = sobel(A,'x');
Iy = sobel(A,'y');

G = zeros(2);
for i = 2:1:2*w(1,1)+2
   for j = 2:1:2*w(2,1)+2
      G = G + [(Ix(i,j))^2 , Ix(i,j)*Iy(i,j) ; Ix(i,j)*Iy(i,j) , (Iy(i,j))^2];
   end % for j
end % for i

% start iteration here
while rep ==1

   % Extract subpixel values for B
   B = zeros(2*w(1,1)+1,2*w(2,1)+1);
   ry =1;
   kolom = 1;
   for i = point(1,1)+g(1,1)+v(1,1)-w(1,1) :1: point(1,1)+g(1,1)+v(1,1)+w(1,1)
      for j = point(2,1)+g(2,1)+v(2,1)-w(2,1) :1: point(2,1)+g(2,1)+v(2,1)+w(2,1)
         [B(ry,kolom),status] = getsubvalue([i;j],imJ);
         if status == 0
            return
         end % if status == 0
         kolom = kolom +1;
      end % for j
      ry = ry+1;
      kolom = 1;
   end % for i
   % Finished extracting matrix B
   deltaI = double(A(2:2*w(1,1)+2,2:2*w(2,1)+2)) - double(B); % A-B
   b = zeros(2,1);
   for i = 2:1:2*w(1,1)+2
      for j = 2:1:2*w(2,1)+2
         b = b + [deltaI(i-1,j-1)*Ix(i,j) ; deltaI(i-1,j-1)*Iy(i,j)];
      end % for j
   end % for i
```

Stellenbosch University http://scholar.sun.ac.za

```
   n = G\b;
   v = v + n;

   loops = loops + 1;
   if loops > 6
      rep = 0;
   end % if looped more than 10 times stop iterating
   if (n(1,1)^2+n(2,1)^2) < 0.03
      rep = 0;
   end % if n smaller than threshold
end % while rep == 1
% end iteration here
```