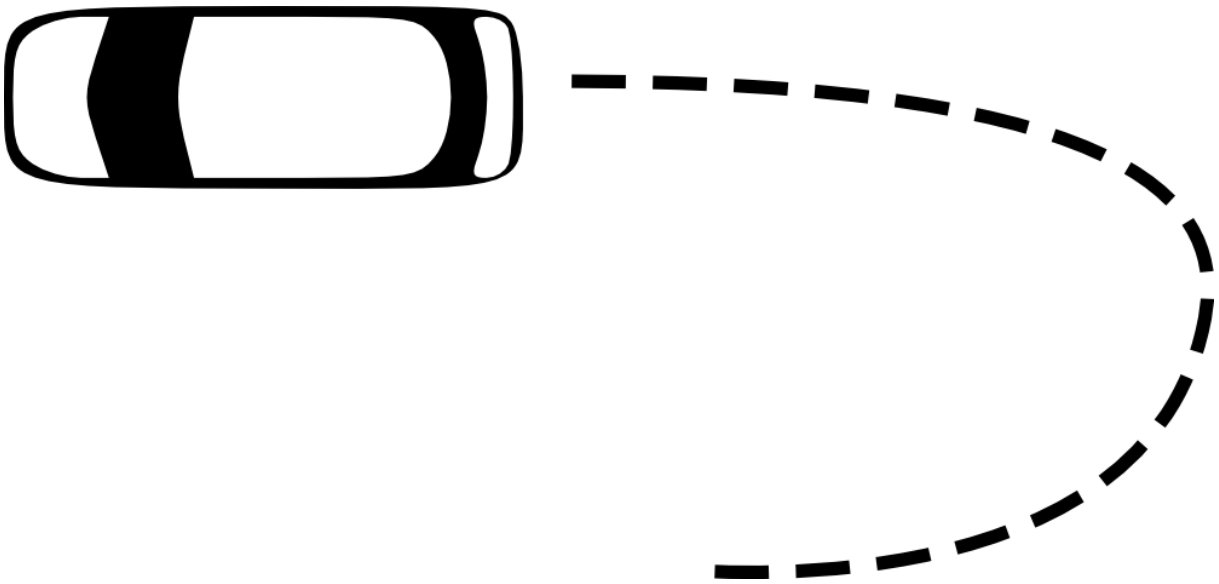
Mapping Environments with Vision Systems



A vehicle tracking concept

Course	Technical computer sciences
Subject	Graduation project
Study load	30 EC
Date	2 June 2014

Student	Koen Braham
Student no.	1583711

Company	Alten PTS
Supervisors	S. Timmerman and M. van Dijk
Examiner	A. Kamphuis and H. Beerlage

Foreword

This report describes the internship work at Alten PTS were I have been building a vehicle tracker.

During my internship at Alten PTS I had a great experience working with the consultants at Apeldoorn. During the project they offered advice as necessary, tips on building things better and created a friendly working environment.

1. Table of contents

1 Management summary

This report is written to conclude the graduation project on the subject of building a vehicle tracking using a computer vision system. This project researches the methods required to build a computer vision system – able to track individual object. The system aims to track objects such as vehicles – including cars, vans and automated robots. It is required to have video input to detect these objects.

It is recommended to:

1. Update the grouping algorithm
2. Find a better position for using the camera system
3. Run more tests using the software
4. Use a better camera than a cell phone

Motivation

As the vehicle tracker has issues tracking tailing vehicles – vehicles move closely behind each other – a new parameter is required to allow dynamic sizing of groups. This will compensate for big vehicles as vans and trucks and separates smaller vehicles.

After conducting the positioning test it is concluded that the effects of positioning the camera will greatly improve the detection and tracking of the system. An elevated position on a roof will help.

During the tests with the algorithms some detection algorithms may have performed poorly due noise in the camera source. Using a better camera device will allow a better detection of motion using more advanced techniques.

More tests are required to validate the program for non-stop operation. Current tests have been executed using recorded videos and may not represent all conditions present on a parking lot.

Consequences

1. It requires time to improve the grouping algorithm as it needs a new dynamic approach to grouping.
2. A new position may not be accessible.
3. Buying a new camera may possibly not improve the captured image quality.

2 Introduction

Alten PTS is a consultancy company focussing on technical software. The business unit Apeldoorn has run a couple of projects focussing on the safety in a working environment. This includes research on applying modern techniques to the working environment, such as mapping the environment. One of the projects involved development of an android application detecting obstacles in a passageway of a hospital. This application emits warnings when it detects obstacles and a stretcher is incoming. It allows removal of the objects prior to blocking the stretcher, and thus possible avoiding accidents.

Project MEVS – Mapping Environments with Vision Systems – is a graduation project focussing on computer vision systems. MEVS is a project aiming to map movement of vehicles. The system developed in this project will map the movement using a camera system and analyses the captured video. Upon analysing MEVS will detect motion and stores all detected events in a database.

The structure of this thesis is divided in four topics, the introduction to the problem, the research for the problem, the implementation of the system and the results of the system. The introduction is written in Chapter 2, describing the current problem, assignment and research questions. The research in Chapter 3 contains the research as done during the project, based upon previous work of tracking systems and related to other projects. It describes the algorithms commonly used for tracking, the relevance of the algorithm for tracking and the advantages or disadvantages of each algorithm. Topic three on the implementation and a demonstration of the system is written in Chapter 4. The implementation describes a new hybrid method build for the MEVS system. The demonstration is a system to track vehicles on a parking lot. It uses MEVS to track vehicles and stores all tracks in a database, storing it for further processing. The last topic is the conclusion of the project, including the experiments in Chapter 5 for testing the system. Chapter 6 shows the results of implementing the system and experiments as described in Chapter 5. The conclusion is written in Chapter 7 and the final recommendations in Chapter 8.

This reports concludes that the MEVS system is a good tracker for moving objects. The combination creating a hybrid system as discussed in Chapter 4 creates a flexible system allowing detection of moving objects and precise tracking after the detection phase.

2.1 Alten

Alten, the European leader in engineering and technology consulting, provides support for its clients' development strategies in the fields of innovation, R&D and it systems (Alten Operations, 2014). Over 14000 engineers are employed by Alten. Alten in the Netherlands is situated in Eindhoven, Capelle aan de IJssel and Apeldoorn. Alten in the Netherlands consists of three business units, all working on different fields of engineering.

Alten PTS is focusing on technical automation. This business unit is the largest of the three and provides engineer to consult at projects, in house research and development for projects and trainings for both internal and external courses. Alten mechatronics is the second business unit providing services in electro-mechanics and robotics. Alten DDA is the third business unit providing installation for technical projects.

Project MEVS is a project started at Alten Apeldoorn. The project is developed as a graduation project. As Alten focusses on consultancy at external companies only a few internal projects are executed. These projects are built in the Alten Development Centre (ADC). During the period of February – June four projects are situated in Apeldoorn. Two of these projects are for external clients developed by Alten. The other two projects are both graduation projects.

2.2 Background

In a modern working environment employees will be supported by automation systems. This includes moving systems as robots which supply parts or other objects to the employees. Each moving robot is equipped with sensors to detect its location, movement and obstacles on its path. Central control of the vehicles could improve the coordination and efficiency by using globally planned routes. Such a system can predict occupied spots where vehicles have to unload and thus are blocking the track.

MEVS focusses on the tracking of these vehicles. Its designed to be a modular system allowing flexible detection of moving objects, including vehicles. MEVS uses vision technology combined with camera's – could be existing security camera's – to track vehicles seen. In this first version of MEVS the system will be tested using cars parking on a parking lot.

2.3 Assignment

Design a adaptive system able to track movement of vehicles. The system will use camera system in order to have low cost coverage of an environment. It is desired to create an adaptive flexible system to allow multiple application building on top of this system. The system should show all detected movement on a map. The map has to be preloaded in the system in order to create the connection and conversion from camera image to map.

Follow-up projects using this system could be one of the following:

- Recognition of collisions (could be a person and a moving object, or two moving objects)
- Recognition of risks in a zone. A warning may be given as a person closes in on a hazard position near a machine.
- Route optimization by tracking routes often used and observed by the system.

2.4 Research questions

In order to reach completion of the desired situation as per assignment the following research question will be answered:

What is required to map motion in an environment using a camera system?

The following questions will help to answer the research question:

1. How can displacement (velocity and direction) of an object be determined?
2. What is needed to avoid object identity switching?
3. What is required to implement such a system?

2.5 Relation to other projects

The focus on a parking lot is due the relation with "Parkeer Beheer Tool", a project running at Alten Eindhoven to track parking vehicles. As the parking lot at Eindhoven has limited spots reserved for Alten it is desired to have an indication of free spots. Using a parking tracker the number of occupied spots could be determined, and the number of free spots calculated. As the park is shared with more companies a certain number of cars are parking in the reserved spots of other companies. This results in frustration. The tracker might be able to indicate vehicles parked on reserved spots where they don't belong and could generate statistics on how many cars park wrong. All cars are identified upon the entry of the parking lot by a license plate scanner. The parking lot in question is displayed in *Figure* 1.

Figure 1: Overview of the parking lot in Eindhoven

3 Research

Nowadays much research has been done in the field of object tracking using computer vision. Modern day computer vision for object tracking can be categorised broadly in three categories, colour segmenting, background removal and feature detection. The first is a simple method using tracking the colours of objects. Colour tracking of objects is relatively easy to implement and can be used for tracking single objects with predefined colour (patterns). However, tracking colour has the major drawback of being dependent on the current environment. One of the big influences on colour tracking is lighting. Lighting (dark / bright) has an impact on the colour as seen by a camera. Adding colour variations due other changes in the lighting will worsen the detection rate. Having these drawbacks makes colour tracking less interesting in the field of object tracking. It depends on uncontrollable parameters which will limit the use cases for the tracker.

The second method is background segmentation in combination with blob detection. This algorithm "learns" the background as it runs. Due the learning nature of this algorithm it needs time for initialisation. After the initialisation phase the background segmentation provides a good extraction of the motion observed in the video. It represents motion by marking pixels in an output image. Grouping these blobs of pixels allows on to detect objects as they move in a video.

The third category is feature point tracking. This method is based on locating points on objects and track them in subsequent frames. An often used method for feature tracking is based on optical flow. This method tracks the changes in the image to determine the overall flow in the image. A typical use case for this method is image stabilisation. As the whole image is tracked algorithms can calculate the amount of correction that is required and where to correct the image. Another use case is grouping the optical flow points, allowing tracking of objects. As the distortion is moving as a cluster it can be grouped and tracked as one.

This chapter on research will summarize previous work done on park monitoring and traffic monitoring systems and the methods usable for project MEVS. As the lighting conditions on the parking lot are uncontrollable the tracking by colour method will not be further investigated.

3.1 Previous work

Tracking vehicles on a parking lot is a common topic for projects in computer vision projects. It allows testing of new algorithms for tracking and image procession in semi controllable environments. Vehicles are large objects to track allowing many cameras to pick up the motion and use it for tracking. In this paragraph some of the previous results in the field of tracking vehicles will be discussed.

Many park tracking systems are not focussed on following vehicles (True), (Wu & Yi), (AI-Absi, Sebastian, Dinesh, Devaraj, & Yoon, 2010), (Tschentscher & Neuhause, 2012). They scan all spots and detect whether or not a car is parked. These park trackers use static images of the park to determine the status of the park spots. Each spot is a region of interest – a subset used for processing. To determine whether or not a spot is vacant a variation of algorithms is used. In some systems an image of the empty spot is compared to the current image of the spot (Modi, Morellas, & Papanikolopoulos, 2008). Changes indicate a possible parked vehicle. Doing this comparison has many flaws, most of all it being affected by the weather. As the shades of a building next to the parking lot move over the spots they get marked as occupied. This also happens when a car parks on a spot and its shade overlaps the spots next to it. This results in false positives. A better method proposed in (True) is to count the number of corners detected in the spot. As empty spots and shades are large smooth areas the number of detected corners should be lower than detected on a vehicle.

Another field of vision studies is traffic monitoring. These systems focus on counting the number of vehicles passing point x. To do so they have to detect motion and count them as they pass over a certain point. Most systems are using a virtual line as a marker to increase the counter. The vehicle type is based on the size of a vehicle – as seen on the source of the detector – allowing distinction between car, van and truck.

In “A real-time computer vision system for measuring traffic parameters” (Beymer, McLauchlan, Coifman, & Malik) a method is proposed to track vehicles on a motorway. Most algorithms will work during normal flow as the distance between cars is large. This allows the system to focus on one car per measurement. However during traffic jams the distance between cars decreases, down to only a few metres. Having multiple cars moving near each other makes it harder to separate them in the images and avoid counting errors. Their method proposes the tracking of vehicles based on optical flow. Tracking position of each vehicle, and this velocity allows detection during free flow. Add a tracker for displacement and direction and the system can discriminate cars based on flow on the road itself. Small changes like steering corrections during a traffic jam allows them to be tracked. This method has proved to be more reliable in the traffic jams.

The demonstrator build upon MEVS aims for using the monitoring approach for following vehicles. This allows them to be identified and mapped to a spot where the car is parking. Connecting MEVS to the parking monitor developed in Eindhoven allows license plate identification per car on the parked spots. Having this link allows one to check who parked wrong and warn the person as he repeats to park wrong.

3.2 Methods used

Prior to building a system a basic knowledge of object tracking is required to design such a system. In this section the background segmentation and feature detector will be explained with illustrations. Both methods allow object tracking although they offer different grades of tracking precision, performance and required initialisation.

3.2.1 Background remover

A commonly used method for tracking objects is a background remover. This class of algorithms detects the background and tries to separate it from the foreground. This is a so called background segmentation algorithm, as it segments the background from the foreground. The background separation allows easy detection of moving objects, as it creates an foreground image containing all moved pixels. To segment the background the algorithm tracks differences in frames, caused by motion. The moving objects are considered to be the foreground as it moves along a non-moving background.

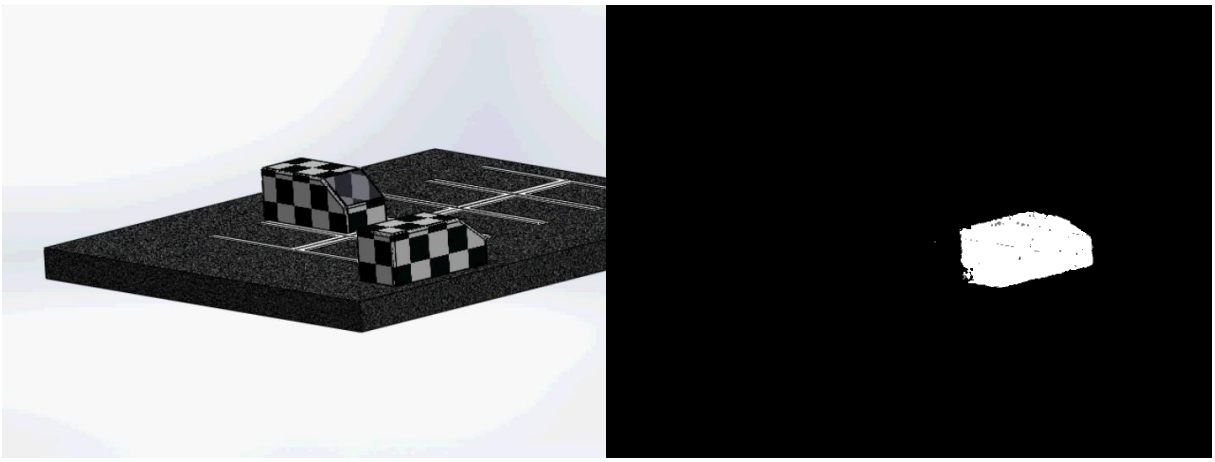


Figure 2: Example of background segmentation. (left) original source. (right) foreground mask found by the segmentation algorithm.

An example of background segmentation is demonstrated in Figure 2. This figure illustrates the motion detected and creates a white blob spanning over all moved pixels. The vehicle in the front – moving to the right – is moving in the video. The other vehicle is "parked", not showing any motion and thus not flagged as moved by the algorithm. As one can see in the right image is it easy to spot the motion in the video as it is clearly separated.

The algorithm holds internally a background image indicating the background. The background is subtracted from the source image in each frame. After subtraction pixels with exact the same value will have 0 as output and pixels with different values end up having a not null value. To magnify the differences a threshold – conversion to black and white image – is applied. This renders background as black and foreground as white pixels. This results in a clear segmentation of background and foreground. As foreground is based on motion – the pixels have changed value – this indicates all moved objects. However to initialise the method it requires the static background to detect the moving pixels. One of the main problems is the retrieval of the static background, as the algorithm has to provide this background for the system. Often the algorithm is started with the current frame and uses this as the current background. Doing this results in a bad detection rate for moving cars which were visible in the first frame capture. After some time the background will settle with the video as the motion reduces over time.

The background image is updated over time using a learning rate. This rate indicates the factor of which it blends in new frames. Lower rates result in slower updates, while higher rates indicate a faster update. As MEVS aims to track vehicles on parking lots a lot of environmental disturbance is expected. One of the most common effects will be sun light being blocked by clouds. This results in shades moving over the park and even changing light levels on all pixels. The learning rate used for the background segmentation must allow these slight changes over time while maintaining the tracking of slow vehicles as is. The lower detection rates allow slower moving vehicles and even short pauses of

movement when vehicles turn or wait for another vehicle. However too low learning rate will cause the lighting issues as explained earlier. An optimum for filtering the environmental noise is desired.

Detection of vehicles can be done using blob detectors on the foreground mask. Blob detection algorithms will find objects in that mask and track the location of each object as the system processes more frames. Tracking blobs is relatively easy as it involves looking for a cluster of pixels. However in environments where objects move closely the blobs can merge. During the event of merged blobs one cannot tell which blob is representing which car. So called identity switching may occur during overlying blobs.

Advantages and disadvantages of using background segmentation

- ⊕ Easy to implement and use
- ⊕ Relatively moderate processing requirements
- ⊖ Sensitive for extreme changes, for example white balancing of the camera changes the colours of all pixels.
- ⊖ No detection of two separate objects. As objects are occluded the system tracks them as one. Identity of objects might be switched by a blob tracker.

The background segmentation method allows detection of movement, but has trouble separating objects located close to each other. This is a problem for object tracking and must be solved in order to use the method in MEVS.

3.2.2 Feature tracker

The feature tracker works by tracking unique points in an image. The points need to be distinguishable as it needs to locate them in the next frame. Upon relocating the points the algorithm can calculate the displacement of each point. Feature trackers allow high tracking accuracy as they use the unique points. For example the Harris corners method finds corners and uses intersection methods to refine the accuracy up to sub-pixel level. Due the fact that corners often represent a big change in the image the intersection is unique enough to be tracked. An example of the Harris corner points is illustrated in Figure 3. Shi-Tomasi's "Good Features To Track" (GFTT) method is based on the Harris corner detection and adds an eigenvalue for each corner (Nourani-Vatani, Borges, & Roberts, 2012). Comparing the eigenvalues of each corner allows to create a selection of the best to be tracked corners, also known as the most unique points. The algorithm calculates the eigenvalues for each pixel in the image. After this he algorithm searches for the largest eigenvalue in the image as a base value. All corners larger than a set quality measure are considered good. Another addition is the minimum distance between corners. This ensures that weaker corners closer than distance_min as minimum distance to a strong eigenvalue corner is removed. An example of a high eigenvalue corner calculated by the Good Feature To Track method is illustrated in Figure 4. An zoomed in version of the corner is illustrated in Figure 5. The corner is located on the edge of the body of the car and the window.

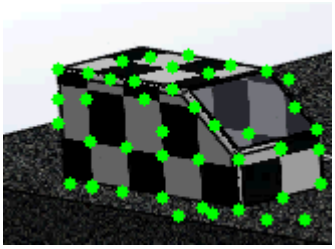


Figure 3: Illustrating Harris corners as found on a vehicle

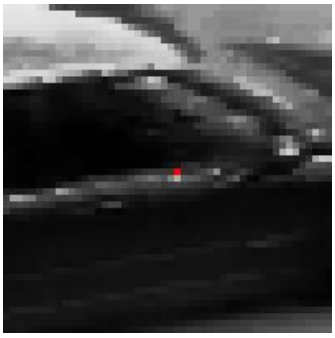


Figure 4: Example of high eigenvalue corner found by the Good Feature To Track method

Vehicles often have these corner points on edges of colours and change of materials. For example the glass windows often have a rubber seal. The difference between these materials is a good point to track, as seen in Figure 5. Other points are panel gaps, head and tail lights. Tracking feature points requires more processing power than colour tracking. To track a point the algorithm reads the location of point X in frame(i) and locates it in the next frame(i+1). To relocate the point the algorithm scans a searching window around the last known location. Having multiple points may cause the algorithm to read the same pixels multiple times as the search windows overlap. The feature points allow tracking position, motion and shape as described in "A Real-time Computer Vision System for Measuring Traffic Parameters" (Beymer, McLauchlan, Coifman, & Malik). This paper proposes a method by extracting the displacement information and track vehicles in congested traffic. During the traffic jams the distance between cars decreases but the individual direction changes per car. Monitoring these small changes allow grouping of the feature points and thus tracking the vehicles. As they are tracking sub-features of a vehicle it compensates for partial occlusion as well. There is no need to see the whole vehicle to detect its motion.

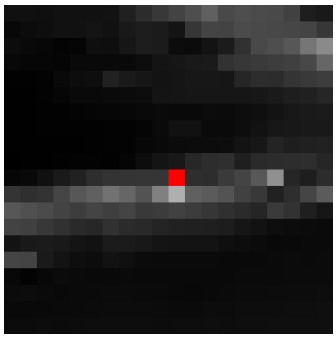


Figure 5: Example of zoomed in high eigenvalue corner

Optical flow is a generic name used for detecting motion and flow of objects – even backgrounds – using feature points. Optical flow is used for camera stabilization by tracking the background and correcting the detected movement. Another use case for optical flow is tracking of smaller objects such as vehicles. The algorithm tracks the flow throughout multiple frames by grouping multiple points and using them as a cluster. Following these clusters will result in a tracker built for tracking unidentified – not knowing the size, colour or any other parameter – objects.

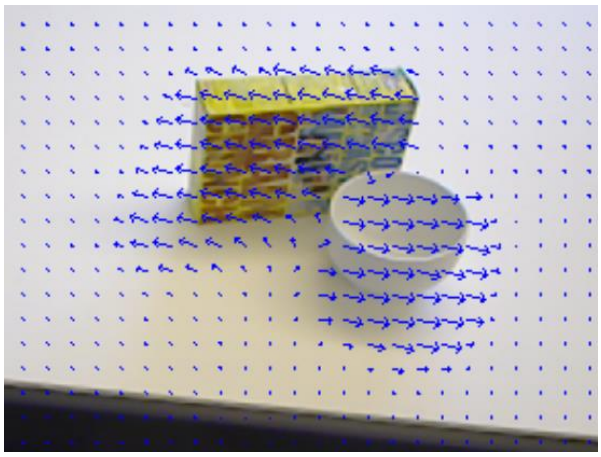


Figure 6: Example of a dense optical flow on two objects (GSoC'14, 2014).

Optical flow algorithms that estimate the displacement for a selected number of pixels (points of interest) are referred to as sparse optical flow algorithms. All feature point methods, including the Harris corners and the Good Features to Track method are sparse methods. These algorithms assume local smoothness in motion. As the algorithms only track very unique points in the region they are more robust to noise. As sparse algorithms only track a few points, dense algorithms will track all pixels in an image. As seen in *Figure* 6 a dense algorithm covers the whole image. Each point is tracked for movement. Grouping this flow will result in two moving groups and the background. Both moving groups represent one object as found in the image. Methods using this type of tracking are the scale invariant feature transform (SIFT) and the speeded-up robust feature (SURF). These algorithms excel at finding predefined objects in a larger image.

Advantages and disadvantages of using feature points

- ⊕ High accuracy on tracking points in an image.
- ⊕ Grouping points allow the average motion to be followed.
- ⊕ Reduces number of features based on position and motion. Stationary features are deleted and thus effectively removing the still background.
- ⊖ Requires much processing power to initialize (finding good features)
- ⊖ Requires more than average processing power to track features (find features in next frame) depending on the window the algorithm uses to find the new feature point.

As motion on vehicles is expected to be relatively smooth – moving small displacement per frame and not rotating by a large angle – the dense methods offer no benefits over the sparse methods. As tested in (Nourani-Vatani, Borges, & Roberts, 2012) the differences between the feature extractors for frame-by-frame tracking under normal conditions is very small. During their evaluation the sparse methods often score better than the dense methods on features found, number of features tracked and the processing time. The Good Features To Track method scores marginally better than the Harris method. It has less located features, due to the extra distance and eigenvalue checks, but scores better on the number of tracked features. As MEVS is a tech demo – focussing on the concept – the performance is

less important than the tracking score. Based on these tests the Good Features To Track method will be used as feature tracker.

3.2.3 Kalman filter

The Kalman filter is a filter uses states of a system to predict the changes over time. The filter supports estimations of past, present and future states given a matrix describing the modelled system (Welch & Bishop, 2006).

In some systems the Kalman filter is used to predict the movement of vehicles, and use it to detect new points (Beymer, McLauchlan, Coifman, & Malik). Such a system has a powerful prediction allowing to group the points and discriminated on the prediction system. This enables to system to identify vehicles based on their movement and prediction of movement.

Figure 7: Example of feature points grouped using a Kalman filter (Beymer, McLauchlan, Coifman, & Malik)

MEVS could benefit from the prediction step in the Kalman filter. This could be used as a detection zone for new feature points. Using this prediction step movement of vehicles can be predicted and corrected as needed. It allows tracking of very small changes in the vehicle heading. This enables a system to separate vehicles based on the heading as used in (Beymer, McLauchlan, Coifman, & Malik). This system uses the prediction for slow moving cars in a traffic jam, as seen in *Figure* 7. As drivers keep steering and in some degree move from side to side on a lane the system can identify and separate all vehicles visible in the camera feed.

4 MEVS algorithm

As described in Chapter 3 two methods commonly used to track movement are background segmentation and feature point tracking. Both methods excel at certain points. However due to the flaws in each method neither is perfect for tracking vehicles. The background segmentation algorithm has issues with objects moving closely and merges them. The feature point detector has a huge cost for initialization of points to track. To compensate for both problems a hybrid method will be investigated.

The MEVS vehicle tracker combines background subtraction and feature tracking to create a new hybrid method. Combining the strengths of both allows fluid detection of object and precise tracking up to sub-pixel accuracy. In this chapter the tracking algorithm will be explained using a fragment taken of a video captured at the Apeldoorn park.

4.1 Implementing MEVS

The MEVS algorithm consists of 5 steps for tracking. The 5 steps used to processing a frame are illustrated in Figure 8 using a diagram. The captured source have to be pre-processed to correct perspective issues in the camera. The positioning of the camera influences the accuracy of the detection and detection rate. The effects of positioning can be reduces by correcting the viewable plane to a semi top view situation. Transforming the image cannot results in worse image quality for tracking the transformation add pixels and does not remove them. It does improve the feature algorithms as the displacement of pixels is not dependant on position in the image.

Figure 8: Diagram of the process steps

The first detection step is executed after the pre-processing phase. This step uses a background segmentation algorithms to find all motion. Followed by the motion detection is the vehicle tracking, using feature points. New points have to be located and tracked. The old points are updated to the newly located positions. Finally after a vehicle has travelled a path the track is processed. This includes filtering and optimizing to reduce the data size in the algorithm. The processed path can be used in applications for statistics processing. An demonstrator using the tracks to determine occupied park spots has been developed.

A fragment of a recording at the Apeldoorn park is to demonstrate all steps. The park is visible in Figure 9. As OpenCV mainly works with grey images the input must be converted prior applying any method. On the right side in Figure 9 the grey image is visible. The image is converted using the standard RGB – a commonly used colour space using the Red, Green and Blue components – to grey conversion of OpenCV. This reduces the amount of data as only one data channel is used. MevsCV – the MEVS Computer Vision library – uses the VideoCapture from OpenCV, allowing to open (web) cameras and video files stored on disk. This ensures the algorithm can work with offline videos as online cameras tracking live movement.

Figure 9: Source image as captured on the Apeldoorn park.

4.2 Frameworks used

Project MEVS uses OpenCV – a computer program for image procession, mostly a collection of algorithms for various image procession techniques – as a basis for the MEVS tracking algorithm. OpenCV is one of largest libraries – collection of algorithms – available for computer vision. OpenCV is a freely available open source project for both academic and commercial use. It is maintained by willow garage. One of the main contributors is Intel, providing optimizations for faster execution times in each algorithm.

The background subtraction techniques and the feature detectors are embedded in OpenCV. This allows rapid development of the hybrid technique as the used algorithms are available.

Another framework used it the Qt framework. This allows easy use of signals and slots – a transmission system for communication between classes. It handles threading and other dangerous patterns which often cause errors in computer programs.

As MEVS is built for testing the hybrid method all parts of the application run in single threaded modus. This allows easy debugging and watching the memory for errors as no threads can cause conflicts during execution. However, it does set a penalty on the performance MEVS could have reached. It greatly reduces the number of frames it can process. An easy optimisation for using MEVS as a live tracking is increasing the number of threads used in this algorithm. A couple of steps in this algorithm could run in parallel while other steps could benefit of a pipelining. This allows the processor to start processing a new frame while the next step of the previous one still has to be executed on a different core.

4.3 Pre-processing

After the image is converted to a grey image the image is improved using a histogram equalisation. The equalization of the picture increases the contrast of the captured scene. This improves poor lighting conditions and makes it easier to detect cars. The final pre-process step is perspective correction. This allows objects to have roughly the same dimensions in the front and back of the image. This step is necessary for the feature point detector as it calculates distances between points. Without warping the cars in the front will be much bigger than the cars in the back. This results in hundreds of feature points detected on cars in the front and only a few to none in the back. The required transformation matrix (H) is calculated using the OpenCV `getPerspectiveTransform`. This function takes a quadrilateral and warps the image to match a pre-set quadrilateral. The transformation matrix is used for translation, rotation and scaling.

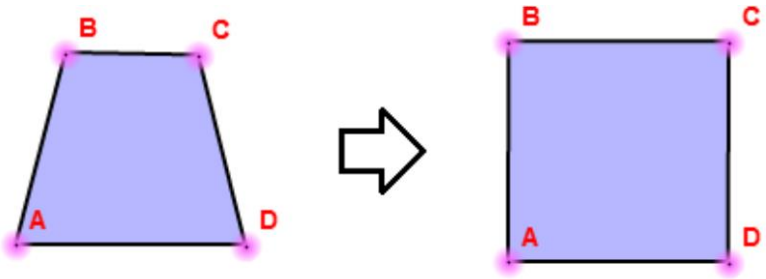


Figure 10: Example of source quadrilateral to transformed rectangle

As shown in Figure 10 the top of the transformed quad requires filling and stretches the image to the rectangle. This effect causes the objects visible in the top to have less detail than the objects in the bottom regions. Reducing the detail in the objects had an unnoticeable effect on the tracker. The tracker focusses on the corner points which are still visible for the algorithm. An example of the translation is illustrated in Figure 11.

Figure 11: (left) the overlay quadrilateral used for extracting the (right) warped image. Background of the park has less detail than the foreground as seen in this figure.

However the tracker does track more smaller objects as they are enlarged in the back projection. For example in one of the source videos a pram is tracked as it moves in the background. The enlarging of objects is visible in Figure 12, it shows how vehicles in the back are almost twice the size of vehicles in the front. This is caused by the effect of the current translation and the rotation.

Figure 12: Larger input quadrilateral for a "top view" of the park. The viewing angle of the camera is clearly visible.

To “undo” the transformation – converting back to the original source – an inverse matrix of the transformation matrix is required. This step is used in the algorithm to convert feature points located in the warped image back to the source for debugging purposes. Using the matrix formula the new X and Y coordinates for all feature points and groups is calculated using the inverse transformation matrix. This manual conversion is required as OpenCV does not support single point conversions.

$$\begin{bmatrix} x_i' \\ y_i' \\ t_i \end{bmatrix} = map_{matrix} * \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$ ref: (OpenCV Documentation, 2014)

Note: The coordinates used by the transformation matrix are homogeneous.

4.4 Background segmentation

The second step is motion detection. Regions of interest – a window to focus – are located by detecting motion in the video. For the initial motion detection in the video a background segmentation algorithm is used. The background segmentation algorithm is explained in Section 3.2.1. This motion detector is used to reduce the image size processed by MEVS. As the segmentation creates a mask layer containing the detected motion it effectively reduces the region of interest for object tracking. Using a blob detector all blobs – clustered pixels in the image – are retrieved and parsed. Blobs are checked for a minimal size to filter out small noise clusters. The minimal size for a blob is set to 21x21 pixels – a common window for the `cornerSubPix` method used in the feature tracker. This method uses a window of 10x10 and multiplies this to create the 21x21 window. It needs this window size in order to have enough information of the surrounding pixels of the feature point to calculate a sub-pixel point. In order to find the sub-pixel the function iterates to find the accurate location of corners. All gradients are summed within a neighbourhood – the 21x21 window. The algorithm continues the iteration of summation and formulas to find a new point. The algorithm stops when the last calculated point stays within a set threshold. (OpenCV, 2014)

OpenCV has two methods for background subtraction based on the Median of Gaussians (MOG). MOG is the original version as built by The Vison and Virtual Reality group (KaewTraKulPong & Bowden, 2001).This works using the background subtraction as explained in Section 3.2.1. MOG2 uses the same method but improves the detection, it has built-in shadow detection and fills object when possible (Zivkovic, 2004). Although it does have better detection it also finds more false positives, as displayed below. It enlarges detected motion and causes bigger noise blobs than the first version does.

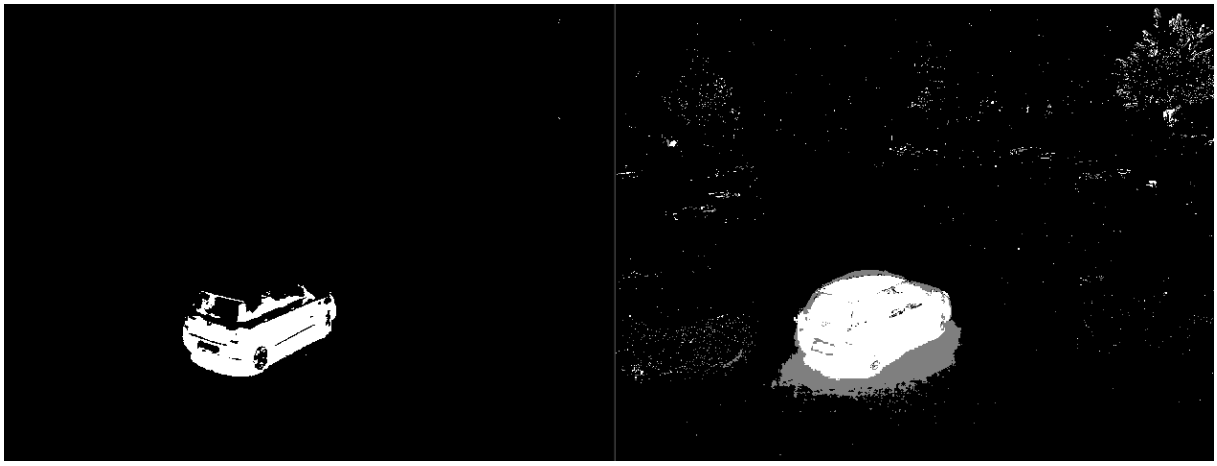


Figure 13: Example of MOG versus MOG2. MOG on the left having only one contour tracked. MOG2 on the right with a lot of noise detected. MOG2 also detects shadows indicated by the grey shade around the car.

Tests using a simulated environment indicated that the MOG2 version of the algorithm performed better, as it had better detection and clearer blobs of vehicles. The MOG2 separator fills the moving contours masker to mask all pixels of the object detected. But applying the MOG2 method on captured frames using a video camera indicated a huge increase of noise and false positives. As seen in Figure 13 and *Figure* 14 the MOG2 algorithm produces a lot more noise than the original version – the MOG2 image is filtered using a median to reduce the noise a bit. As the image quality of the source camera has some noise captured this algorithm worsens the detection by finding a lot of small blobs. Even after filtering a lot of random blobs remain. This results in a slower performing system as it filters out all found blobs using the tracking system and causes a bad detection rate for actual cars versus detected tracks.

Figure 14: The background segmentation. (left) input image segmented to (right) mask image produces by MOG2. White indicates a moving object. Shades detected in the source are drawn in grey.

MEVS uses the `BackgroundSubtractorMOG` for background separation as implemented in the OpenCV library. To update the background the separator uses a learning rate. This is a factor applied to all pixels before it is added to the background before updating. The learning rate allows adjusting the time required before a change in background is marked as background and no longer foreground. This allows a slowly updating background which can filter out subtle changes over time, like shadows. This rate should be low enough to allow slow moving cars to be tracked. Currently the system uses a learning rate which fades cars out in a few seconds after moving, as this rate allows tracking of slow objects.

4.5 Locating feature points

The region of interest mask as introduced in Section 4.4 is used to cut a region of interest from the grey source image. This region of interest is scanned for feature points using the OpenCV `goodFeaturesToTrack`. The found points are refined using the `cornerSubPix` for sub-pixel accuracy. Each feature point detected needs to have a minimum of distance to all other points, this ensures an evenly distribution of the points and avoids big clusters of points on a couple of pixels. An example of the point detection is illustrated in Figure 15. All detected points are matched with all groups. When the algorithm is unable to find a matching group a new temporarily group is created. This group will be marked as active as soon as it has the valid flag, requiring movement in at least 60% of the frames in the last second of the video – on average it means 10 frames per second and thus 6 frames out of the last 10. The 60% is a number chosen to avoid equal number of false and positive situations, as it caused groups to be removed while still being valid.

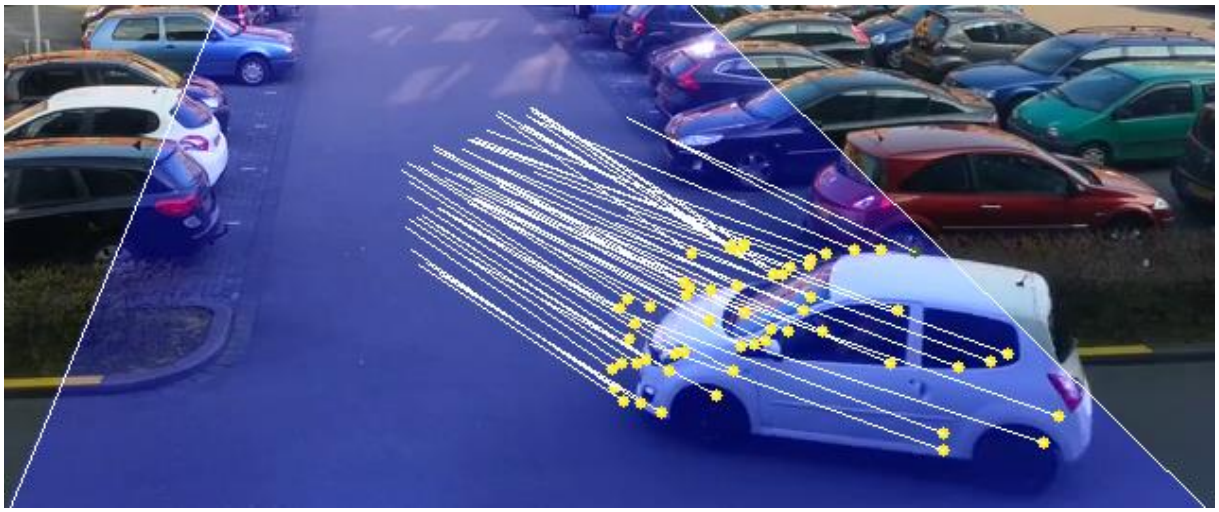


Figure 15: Detection of corner points using the feature detector

Feature groups are shaped as an circle. The centroid is used to determine distance is less than a set maximum the point is considered as possible feature point of the group. This form of shape is the easiest form to implement but may not yield the best results. Other options are rectangle shape overlaying the vehicle and a snake for interconnecting dots. An overview of the grouping shapes is illustrated in Figure 16. As the orientation of the vehicle is unknown the system cannot filter points based on the rectangle overlay. It may filter out good points and track more points in the background on the sides of the vehicle. In order to use the snake clustering an object identification phase is required. This should result in a skeleton body having connected dots on the vehicle. Due limited research time this method is left out of scope for feature grouping.

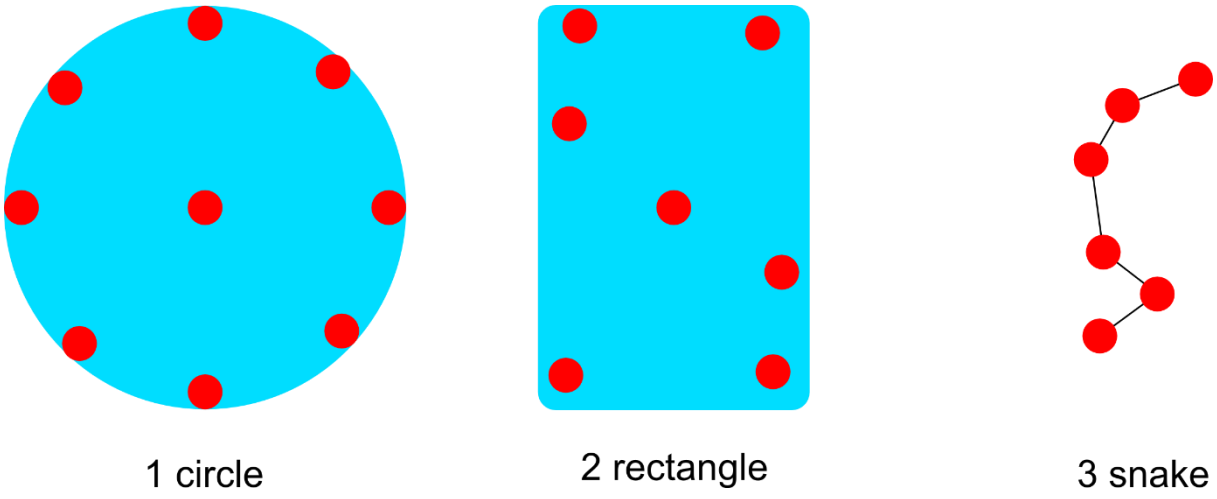


Figure 16: Example of grouping methods. (1) is a circle method based on distance between the points and centre. (2) is a rectangle grouping. This method checks to match a shape of a rectangle, often used for cars. (3) is a method of clustering the points as a snake or worm. This method is used for flexible shaping of more complex objects.

The second grouping requirement is based on the point displacement. Each point is compared with the previous location to calculate the displacement. Using the displacement vector of the point and the centroid of the group an angle is calculated. This angle represents the direction difference between the points. To add a point to a group a maximum angle of 90 degrees is allowed. This effectively ensures the point is globally moving in the same direction or not moving at all. This filtering allows two vehicles to pass eachother closely – within the max distance boundary of points – and still be tracked as separate groups.

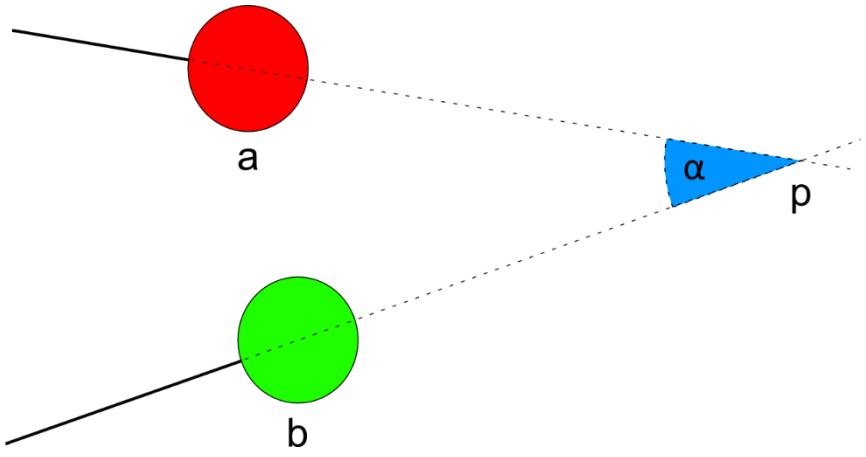


Figure 17: Illustration of two points and their relative angle

Given two feature points the relative angle can be calculated. Illustrated in Figure 17 is the angle between feature point A and B. Both points have a origin – the coloured circle – and a displacement indicator. This vector is used to calculate the relative corner between the points. Using an intersection between these vectors point p is calculated. The angle at α is used to determine whether or not the feature points are considered moving in the same heading.

4.5.1 Testing the feature points

The video source used in the research / testing phase of this project is generated using SolidWorks. This is a program to draw and simulate 3d objects. Using a motion study – method to animate objects in SolidWorks – the vehicle is moved on the ground plane. The vehicle is rendered with different patterns for testing the detection algorithms. Using the simulated environment the algorithms can be tested without environmental influences. The environment is visible in Figure 18.

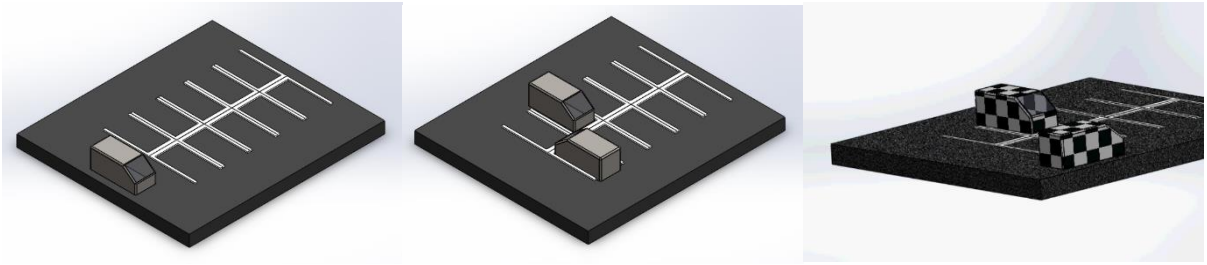


Figure 18: Overview of modelled terrain

Figure 19 illustrates an example of good features to track and tracks them using the optical flow algorithm. In this example the system groups of points by linking them based on distance. The criterion for linking points is when distance between the point and the centre of the cluster is less than set distance. Based upon the displacement of the individual points the mean displacement for the group is calculated. The mean displacements is shown by drawing a red line originated of the centre of the group. As seen in Figure 19 groups 00, 03 and 09 have an arrow pointing in the moving direction. The arrow represents the motion and displacement in the frames. As the vehicle is about to park in a set parking space the arrow bends off to the left and has a decreasing velocity. In this simulation the car rotates around its centre point, causing the back to gain more velocity as it is about to swing out.

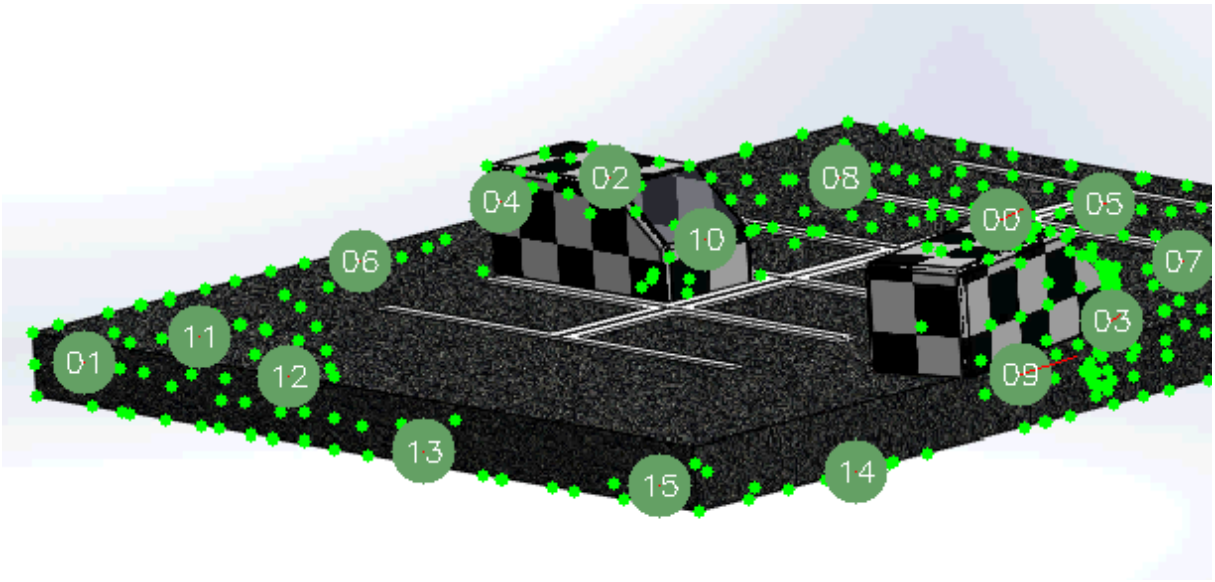


Figure 19: Optical flow on a simulated environment

Having three groups on the vehicle is an indication for over segmentation. The algorithm tracks more clusters than vehicles, and thus false positives on movement. To reduce the over segmentation the size of groups should be increased. However, too large groups will cause under segmentation, causing

two vehicles to be tracked as one and thus missing a potential vehicle. An optimum group size and other discriminating properties have to be found for the MEVS algorithm.

In Figure 20 an example of point reduction is shown. All points which have not been moving for the last frame are removed. The discrimination of the points is based upon the displacement per frame. Distance is calculated using the square root of both x and y coordinates of a point in both frames. The distance has to be more than 3 pixels to be marked as moving. The 3 pixels is used in this example to demonstrate the reduction in only a few frames.

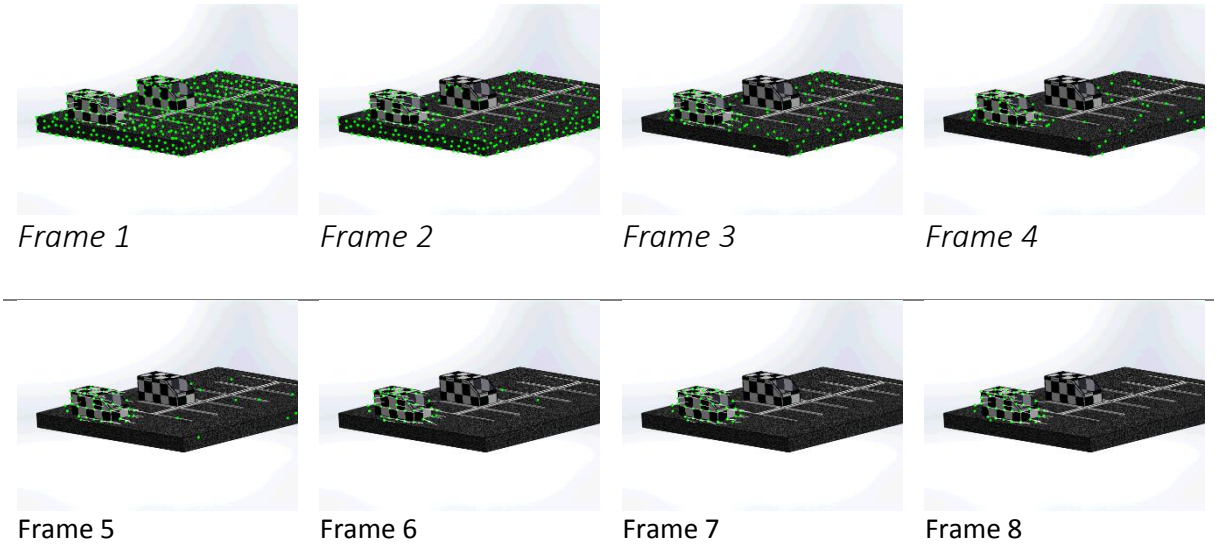


Figure 20: Example of points reduction based on displacement

By processing seven frames the system removed almost all still points in Figure 20. Using this method the vehicle detector is able to differentiate vehicles and background without knowing the background prior system starting. This is a huge benefit over systems relying on having good background data. As the weather changes the background will change in appearance. For example day and night have a huge impact on the colours and features seen in an image. But as this method tracks all features found in the image it requires a fair amount of processing to track all the points – in the first frame the system tracks over 400 points. This will be reduced to about 50 points per vehicle. This reduction is required to maintain a workable tracker. An optimum of 50 – 70 points is found to be working good for reliable tracking of the position of the vehicles while maintaining a fast tracker.

4.6 Feature point and group updates

After detection all points are assigned to a temporary group as described in Section 4.5. To promote a temporary group to an active tracking group it needs to fulfil the groups requirement described in this section.

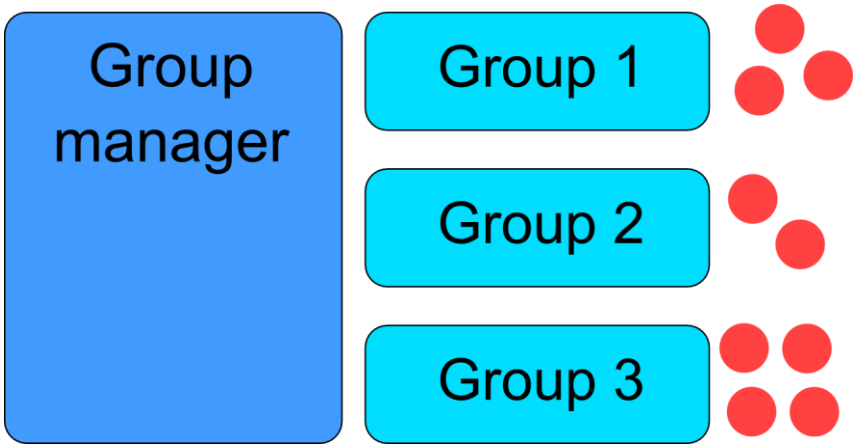


Figure 21: Showing the feature group manager holding 3 groups and a total of 9 points

MEVS uses a feature group manager to store all groups of points detected. To allow flexible addition of groups and removal the group manager uses a list of feature groups. Each group has its own vector containing points.

Feature points are validated by checking internal counters. As written in Section 4.5 a point is marked as invalid if the decay counter reaches 4 errors – thus 40% of the last 10 frames. Every 11 frames the decay counter is reset – this results in a window of one second for motion using ten frames per second. The counter is increased upon errors like not matching the distance to the group centroid or displacement not matching. This counter ensures fast removal of non-interesting points while keeping features that describe the movement of a vehicle.

All points are updated in a loop iterating over the list of points. This loop executes the OpenCV feature methods on each detected group to update. The OpenCV method `cv::calcOpticalFlowPyrLK` is used for updating the points.

A valid group requires a minimum of 10 valid points – this number of points has no foundation and was chosen to separate noise from vehicles. This means empty groups are removed from the tracker. As points are removed on the maximum 4 error counter, groups are removed as well due the lack of points. Temporary groups having at least 10 points at the first 11 frame reset are marked as active and will be tracked as vehicles. Active groups that fail the test once are immediately removed and set as inactive. This indicates the group is no longer moving nor has potential to be interesting any time soon. Each inactive group is processed to retrieve the travelled path and removed after. The path is transmitted as an event for other objects to be parsed.

Active groups can be merged if the centroids are within a set maximum distance and the direction of movement is the same. The illustrate the distance Figure 22 shows two circles. The outer circle is the maximum distance for each point. The direction – orientation and velocity – is an important discriminator between groups. It avoids merging vehicles moving in on each other as the orientation and velocity is most likely different.

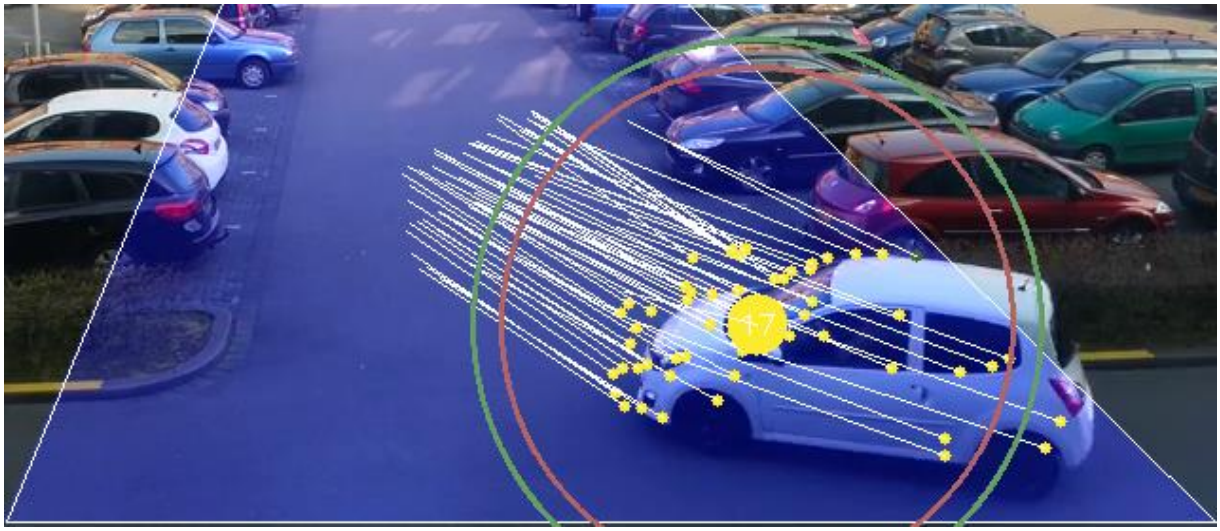


Figure 22: Feature points grouped using the distance and direction parameters. All points in the green circle (max distance radius) and having an equal displacement are added to the group. Once the centroid of another group moves into the red circle (group merge radius) and the group has an equal displacement the groups will be merged.

4.7 Path updates

After receiving the path event indicating the completion of a track the path is processed using the path tracker. This system executes four steps to retrieve the track, illustrated in Figure 23. Firstly it retrieves the meta data, information as number of points. This information is lost due the optimization, which reduces the number of points. The number of points is important to calculate the length (time wise) of the track to determine the velocity of the vehicle. The second step is filtering the path to remove all noise and high peaks. To filter an Kalman filter is used, running a model of a slow system. The third step is to optimize the path. It reduces the number of points required to represent the path. Finally the length of the path is calculated. This allows the system to determine the average velocity of the object.

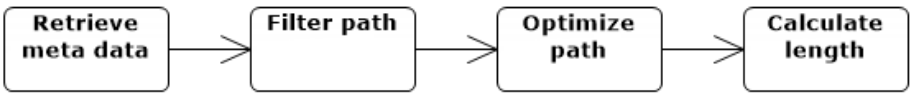


Figure 23: Illustration of the filter steps.

In this chapter each step of the path tracker will be described.

4.7.1 Filtering

Paths received from the feature tracker have to be filtered to remove high frequency noise captured in the video. To filter the paths all points are passed into a Kalman filter. A Kalman filter is an algorithm that filters a series of measurements observed over time to produce estimates (Welch & Bishop, 2006). Doing so removes noise and increases the accuracy to be more accurate than the measurements taken over time. The Kalman filter uses a physics model to predict new values.

MEVS uses the Kalman filter for path filtering. The model used in this system is a slow model. This reduces noise and spikes introduced by the camera and detection algorithm. Most of the noise is a side effect of cheap cameras. The used model is working fine as a current filter model, removing all noise. As Future work may include prediction of movement for better grouping and tracking of motion a better model should be calculated. The current one is not based on a foundation of tests or other vehicle specific settings.

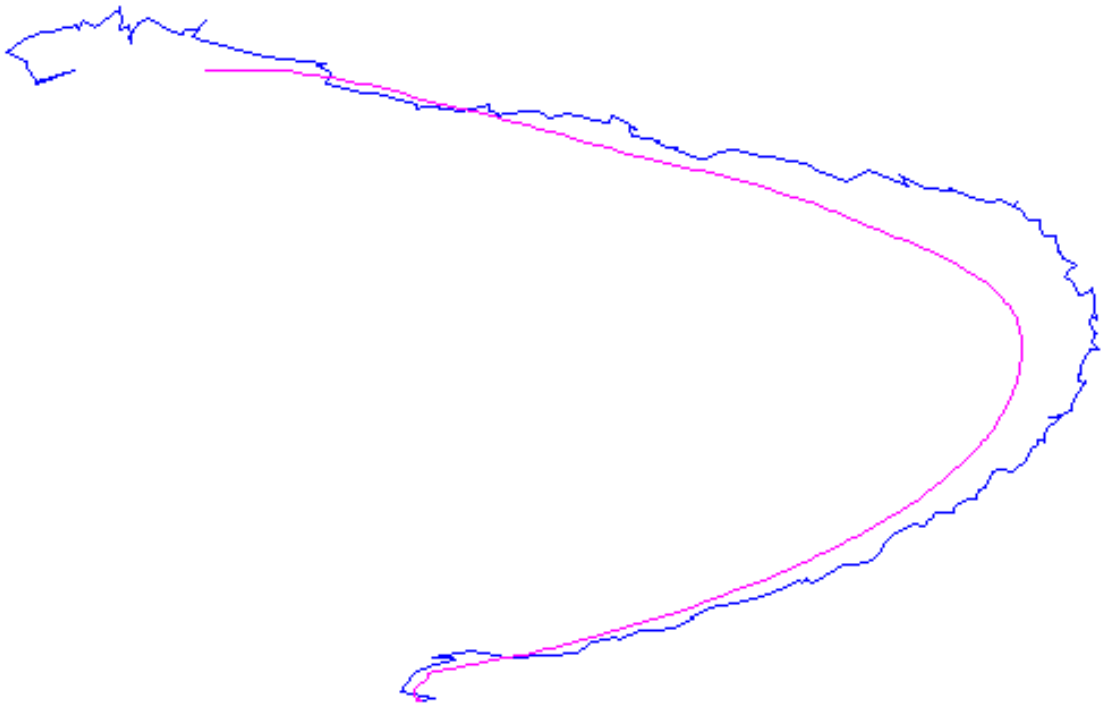


Figure 24: Showing the unfiltered path (blue) and the filtered path (purple) using a Kalman filter

As seen in Figure 24 using a filter greatly reduces the spikes and improves the route as travelled by a car. One should be aware that due the filter the path is shorter than the original path travelled by a vehicle. Further research should indicate whether or not this change in the track negatively influences the system.

4.7.2 Optimizing

After filtering the path an optimization algorithm is used to reduce the number of points stored. In the current prototype all tracks and points are stored for history and validation of the paths. Without optimizing each path uses one point per frame. As a car needs at least a few seconds to park this results in hundreds or thousands of points per path. Using the Douglas-Peucker algorithm an approximation of the path is created. The approximation uses the minimum amount of points required for a set accuracy of the original curve. Given a point on the original line and the corresponding point on the approximation the maximum distance between these points is the accuracy set for the algorithm. However after optimizing the path all meta information such as the velocity are lost. As the points are no longer mapped to the frames they are essentially decoupled from the time frame in which they were recorded. Several optimization settings have been tested and are illustrated in Figure 25.

Figure 25: Comparing different accuracy levels for the Douglas-Peucker algorithm

4.8 Demonstrator

During the development of MEVS a demonstrator – application to show the results of MEVS in a demonstration – has been developed. This demonstrator enables tracking of objects on a parking lot. It tracks all paths recorded by MEVS and stores them in a database, which can be used for further processing and statistics of the parking lot.

4.8.1 MEVS application

The MEVS application is separated in two parts: the application as user interface and a library for image processing. This library could be embedded in other projects not requiring the interface but only the tracking algorithm.

For this demonstrator a graphical user interface has been designed, as seen in Figure 26. This application shows the current frame in the top, having the input on the left side and the transformed image on the right side. On the bottom left side is a map showing the last detected path. The bottom right side has controls for the video, allowing one to play, stop, restart or step through the video. One can set the input of the algorithm and the current park, indicating the camera settings as perspective.

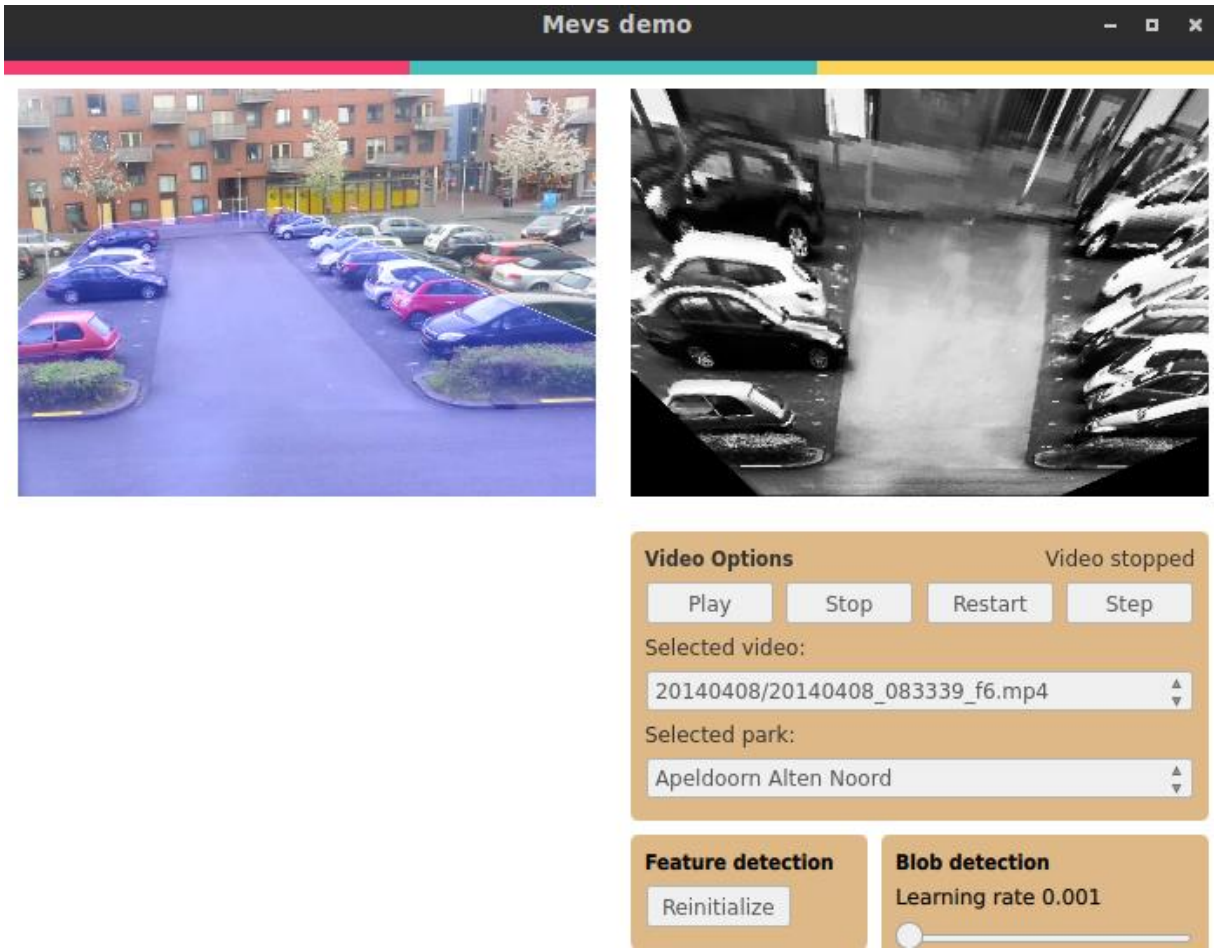


Figure 26: Application developed for MEVS to demonstrate the algorithm.

The park tracker is built upon the MEVS vision system and uses the path tracked as input for further processing. The park tracker requires a database containing all spots and their sizes to determine regions of interest to watch. The park tracker loops through each path it receives from the path tracker in order to detect the parked spot. As each park spot represents one region of interest for the park tracker it has to find the best matching region for the path. To do this a number of options are open for testing, due time limitations only a few have been tested for results.

To detect in which spot the car parked a numerous of options are available. Chosen for further investigation are the following options

1. Count the number of points per parking spot, spot with the most points wins.
   Easy method to implement and should provide reasonable results.
2. Compare the time between the first point in and the last point in the park spot. Spot with the most park time is the spot where the car is parked.
   As the car parks on a given spot, it needs to slow down. This increases the time it is seen in one of the regions, and should result in the longest time for the parked spot.
3. Last intersect of the path and the park spots. This should be the final park spot for the car as it no longer moves.
   This method is insensitive for speed and direction of the car.

An illustration of each method is in visible in Figure 27.

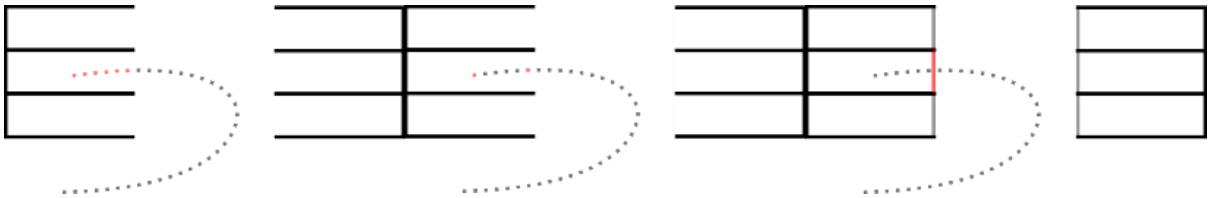


Figure 27: Illustration of three spots detection methods. (Left) counting the number of points in each spot. (Middle) counting the time between the first and last point in each spot. (Right) Intersecting the path with each spot.

Due to tracking errors using the counting method or time based method, MEVS uses the intersection method. As both the counting and time method failed in the situation were a vehicle first moves over a parking spot prior to parking, as illustrated in Figure 28. The counting method counted a lot of points during the turning of the vehicle, and not so many in the actual parking spot. Using the intersection method these situations do not exist. As it follows the track from the last point to the first it always intersects with the last parked spot.

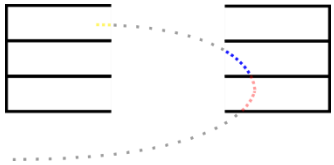


Figure 28: Example of vehicle moving over two parking spots prior to parking.

To track vehicles on a park the system has to map the tracked paths to parking spots. In *Figure* 29 such a system is illustrated.

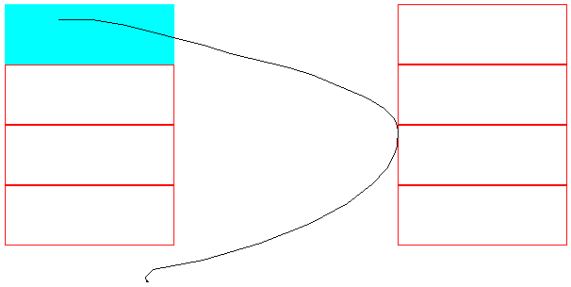


Figure 29: Illustrating a vehicle parked in a detected parking spot.

4.8.2 MEVS web

The MEVS web interface allows one to see all tracks as recorded by the system. It has three pages for showing a dash board – a welcome page visible in Figure 30, the list of tracks visible in Figure 31 and each track as seen in Figure 32. Each track in the database has a list of points, a date, event and length stored. Event is a field indicating whether a car is parked, passed or left the parking lot. Using this information a system showing all parking spots and its last state could be build.

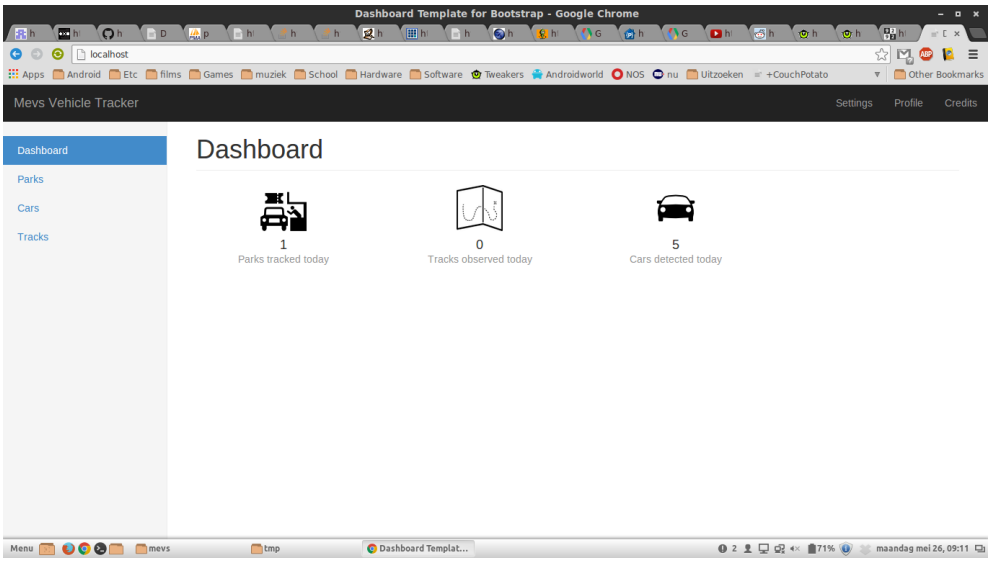


Figure 30: Dashboard of the web interface

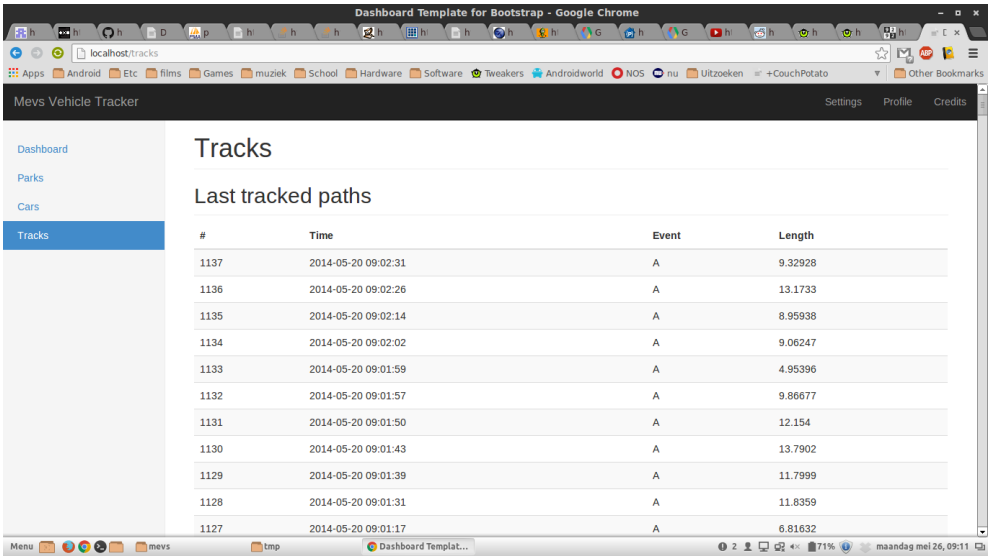


Figure 31: List of recorded tracks

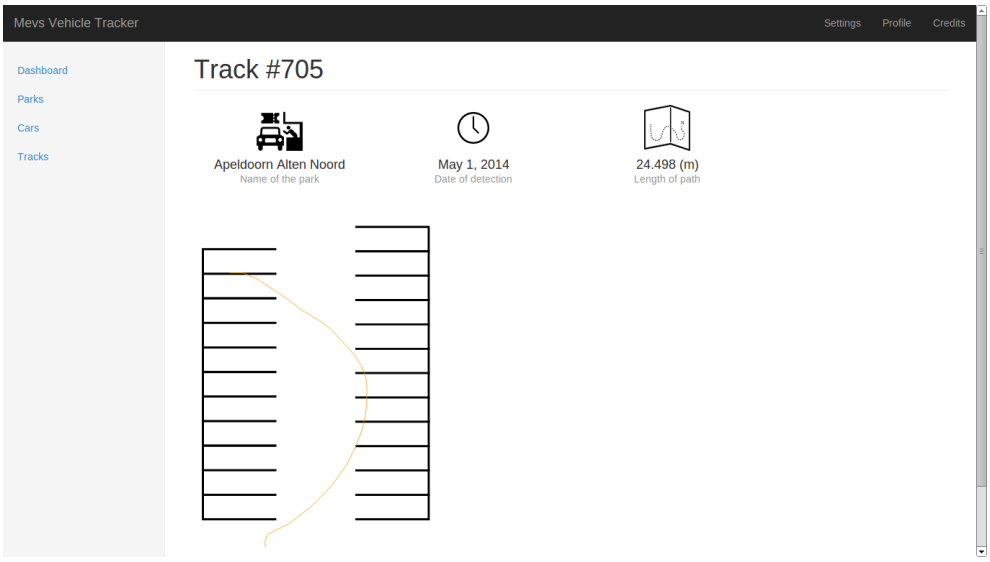


Figure 32: Track page showing the recorded track.

5 Experiments

As the implementation of the first version of MEVS is ready the system has to be tested on its detection performance. In order to test the MEVS algorithm a base line will be set using the background segmentation method. Due the implementation the feature detector the optical flow method cannot run by itself as it needs a region to initialise. This limits the algorithm and thus is not included in the experiments of MEVS.

This chapter will explain the experiments used to test MEVS using various scenarios and videos.

5.1 Two cars passing each other

Two cars (A and B) will pass each other as both cars move different directions, as illustrated in Figure 33. The algorithm has to be able to separate the movement of the vehicles. Identity switching should not be allowed, as it invalidates the track. This means the algorithm cannot switch the groups it uses to track the cars.

To test this scenario the following captured videos will be used. The videos include different environments for the vehicles to pass each other and different lighting conditions. Using human observation the algorithm will be tested in this scenario. During this test the background segmentation algorithm is tested versus the MEVS algorithm, as the MEVS algorithm should eliminate the switching problem.

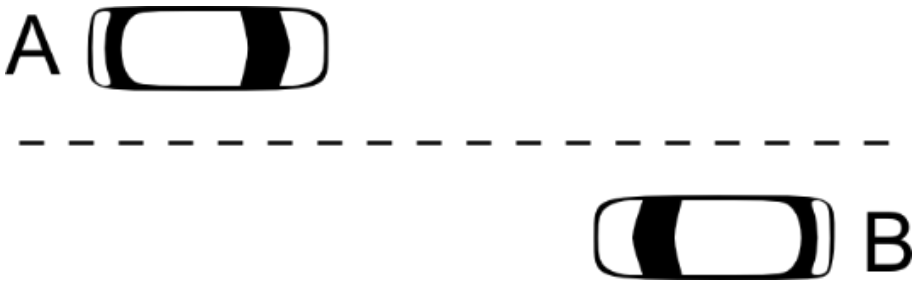


Figure 33: Illustration of two cars passing each other. Both cars have different directions.

5.2 Different lighting conditions

As MEVS is designed to operate on parking lots it has to be able to handle lighting conditions. The lighting on the parking lot in Eindhoven and Apeldoorn is uncontrollable, as both parks are outdoors. Different lighting may affect the accuracy of even render MEVS unable to track vehicles at all.

To test this scenario fragments of a video captured in Eemnes will be used. These fragments have been captured on a sunny day with some clouds. Each fragment is between 5 and 10 minutes. Changes in the lighting is clearly visible in the videos as the clouds move over. Using human observation the algorithm is tested on the handling of the different conditions.

The switch from bright to cloudy happens in about ten seconds. The system should filter this change out of the interesting changes as it is taking more time than set for a vehicle movement. This renders clouds having minimal effect on the tracking of vehicles.

5.3 Position of the camera

The positioning of the camera system has a huge impact on the accuracy tracked over the parking lot. The regions closer to the camera will always yield better accuracy than those further away. Using perspective transformations the feature algorithm is less affected by the distance of the moving vehicle. This test concludes the effects of positions the camera. A heat map can be used as an indication for a estimation of the reliability of the system. As the system tracks more feature points it has more detail in the image and thus better tracking of vehicles.

To test this scenario a heat map will be generated of the average number of feature points per vehicle. The average number of feature points indicates the error over the captured region. It is expected to observe less points in the background than the foreground, as objects slowly shrink due the perspective of the camera.

5.4 Repeatability

MEVS is designed to operate non-stop on a parking lot. As many vehicles will be moving over the same tracks – routes to the empty spots – repeatability of tracking is required. Different vehicles moving in to the same parking spot should result in the same occupied spot calculated by MEVS. Due to the learning nature of some algorithms and semi random picking of feature points the system may deviate while playing the same track.

This scenario will be tested by processing multiple times the same video and map the deviation of each tracking attempt. The maximum deviation will be calculated to test the repeatability.

5.5 Set-up

The set-up used for the experiments consists of a laptop for video processing and a camera for video capturing.

Videos will be shot at the parking lot in Apeldoorn, the office in Eindhoven and a street in Eemnes. In the office building of Alten Eindhoven a camera is directed at the park. The videos will be shot using a mobile phone, specifically the Galaxy Note 2 produced by Samsung. This phone shoots the video using 1080p. After capturing the video it is cropped to 1440 x 1080 and resized to 640 x 480. This allows the system to process less data – as it gets smaller. This reduce step is a necessity to for fast image processing. During the first observations a 1080p video using the background segmentation ran at less than 1 frame per second. It is important to have a support mechanism for the camera, as it cannot be allowed to move during the recording of the video. As the algorithm tracks all motion present in the video it would trigger every pixel as moved when the camera moves.

The laptop is an ASUS K52JB running on a i3 M350 processor. The software runs on Linux Mint, one of the many Ubuntu variations, and runs Linux kernel 3.11. For compiling the software GCC 4.8 is used. As the laptop supports 64bits the compilation was done on the 64bit version of the compiler.

As the experiments focus on the detection rate and tracking performance the speed of the used PC during this experiment is less important. No real-time video stream is processed, resulting in a constant 30 fps video to process.

6 Results

In this chapter the results of building MEVs are summarized. Each test as per Chapter 5 is divided in sections.

6.1 Implementation of MEVS

The current implementation of MEVS uses OpenCV and Qt. These frameworks allow rapid development of a computer vision system. OpenCV contains all (sub-)algorithms as used in the MEVS algorithm and program. Our current implementation is a proof of concept of a system designed for everyday use. The implementation lacks classification of objects and thus tracks both vehicles and humans. A new filter should be added to classify each object and only track the objects that are currently set as tracked class. For example the parking tool in Eindhoven should be able to only track vehicles. This ensures that humans do not cause false positives during the tracking.

6.2 Results of experiments

This section describes the results for each individual test as written in Chapter 5. All fragments used for the test are available in the MEVS project. This allows one to reproduce the results and verify all steps per experiment.

6.2.1 Two cars passing each other

This test demonstrates the differences between a blobs detector and hybrid detector. The first test is using a video captured in Eemnes to demonstrate two moving cars passing each other at a normal road. A blob detector is scanning the output of the background segmentation for objects. All objects are assigned a random colour.

In Figure 34 the two cars approaching each other are visible. Both cars have distinct colours and are easily separated on the detection frame (right side). As the cars pass each other in Figure 35 the detection frame shows one colour for both cars. This indicates the system cannot separate the cars and thus lost track of them. After the pass in Figure 36 both cars are detected again. A system could use information gained before of the cars displacement to identify each car again. However, this only works for non-holonomic systems – these systems depend on their trajectory of states for the final state, and thus cannot change during the time the system tracks the merged blob. Given two people walking on road the system is unable to tell whether or not they walked closely or switched sides. To create a universal flexible tracker as MEVS this method is not sufficient for tracking by itself.

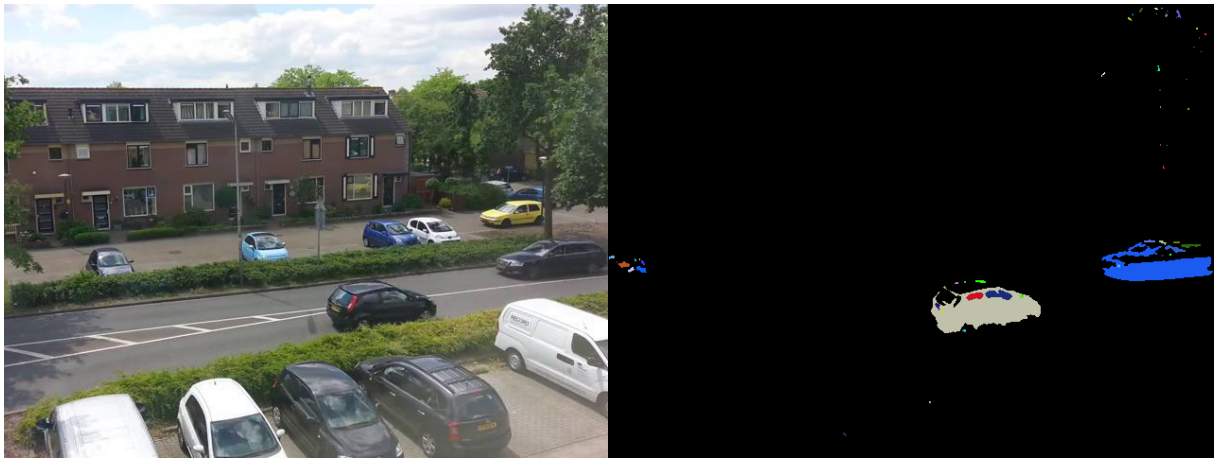


Figure 34: Two cars approaching each other (frame num. 521)

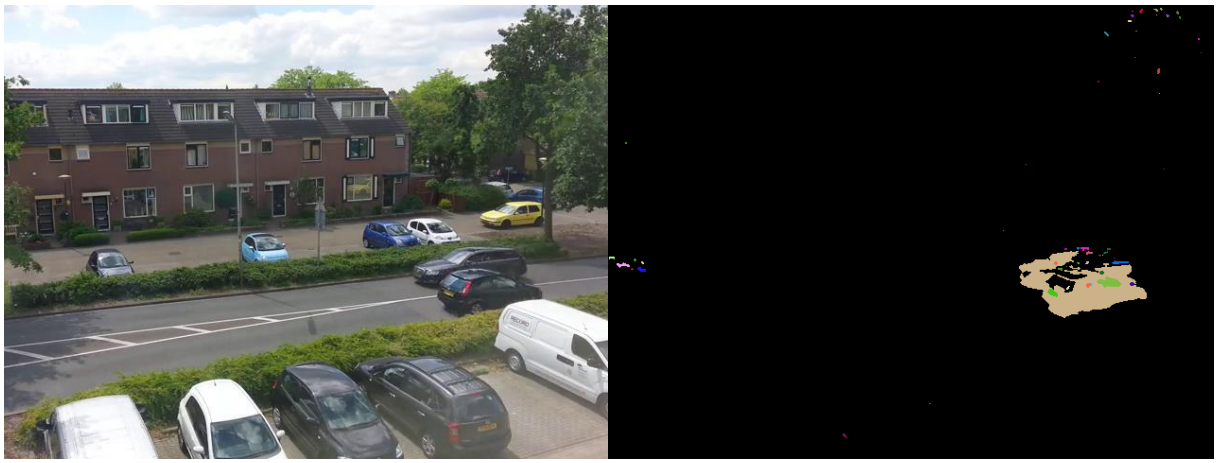


Figure 35: Two cars passing each other (frame num. 533)

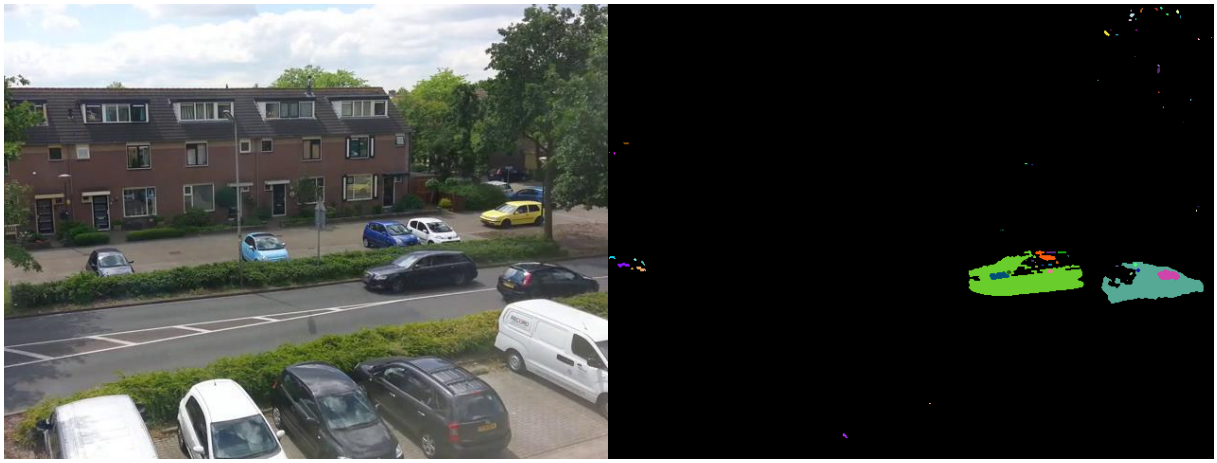


Figure 36: Two cars just passed each other (frame num. 540)

The same fragment is tracked by MEVS. This shows how both cars are tracked while passing each other using the addition of feature points. In Figure 37 both cars are within merging distance. Due the different headings the cars remain separated. In Figure 38 the cars have passed each other. The tracks of the cars – purple and green lines – have never intersected. This indicates that both cars have been tracked without interruption by switching them or missing some points.

Figure 37: Two cars approaching each other, tracked by MEVS

Figure 38: Two cars passed each other, tracked by MEVS

This test concludes the necessity of adding feature points for tracking objects. It improves the detection rate and the tracking of objects close to each other. However, it has only proven to be working for objects moving in the opposite heading. Objects closely tailing each other are merged and tracked as one object. This is due the MEVS tracker only discriminates on distance and heading. A third option of displacement – the velocity of vehicles – may allow more distinct groups and filter out vehicles tailing each other.

6.2.2 Different lighting conditions

Tests using the video captured in Eemnes show strong differences between cloudy scenes and bright scenes. The switch from bright to cloudy happens in about ten seconds. During the tests the system always filtered the effect of the brightness change. However, the system did detect movement in the trees. The leaves were moved by a breeze, having a lot of constant motion. This was filtered out by the small size of most leaves and the short duration of the movement – each time the leaves change direction the group is marked as invalid and thus no longer tracked.

Table 1: Observation of fragments

Video Eemnes	Results
Fragment one	The weather changes from bright sun to dark as clouds move over. Three cars are tracked and no influence of the weather is present.
Fragment two	The weather changes from cloudy to sunny. One car is tracked during the cloudy stage. As the clouds move out more light is present in the video, but no effect in the weather are present. After a while clouds are blocking the direct sun light again. During this transition no effects are present. One more car move into the scene, it is tracked without problems in the cloudy lighting.
Fragment three	The weather changes from sunny to cloudy. One van passes and it tracked correctly. Effects of the weather changing are present. Two parked cars – and thus non-moving – are marked as moved. This may be caused by reflection in the window caused by the sun glare and disappearing as the clouds move in.
Fragment four	Fragment starts with a transition from dark to bright. One car is tracked in the bright scene. No problems are present in this fragment.

However as read in Table 1 fragment three had false detections. The system detected movement on parked cars. As seen in Figure 39 the bright situation caused some reflection in the window. At exactly this position false motion is detected, due the sudden change of brightness and colours. The detected motion is moving fast enough and just enough distance for the current implementation to be tracked as a path. As the false path is detected on one parking spot, the any system using this information could invalidate the path as it never moved out of the parking spot. The duration of this effect is about 40 seconds, observed during this fragment and a video of Eindhoven. The same effect is in the Eindhoven video, as the tracker is initialised with moving traffic. As the background is updated with both moving vehicles and still images of the tarmac it takes more time to initialize. This effect lasted for about 50 seconds.

Figure 39: Two frames of fragment 3. (Left) The bright start situation. (Right) the detected van and wrong identified movement in the bottom.

6.2.3 Position of the camera

To test the effects of the camera position a heat map has been generated using the number of feature points detected in the videos. Two positions have been used to generate the map, first location was at the Apeldoorn Park showing the differences between using a transformation and not. The second location was at Eindhoven as it captures a long road. It clearly demonstrates how the number of features correlate to the distance of the camera.

6.2.3.1 Apeldoorn park

At the Apeldoorn – visible in Figure 40 – park two test have been run to indicate the differences of using the transformation as pre-processor and not. The first graph in Figure 41 shows the base capture of the park without transformation. It is clearly visible the centre of the map indicates the hotspots. This is caused by the position of the camera, as vehicles driving close to the camera – seen on the bottom 4 rows – are not entirely visible due cut off of the bottom. As soon as the vehicle enters the park itself the vehicle appears correctly on the image and is tracked using many points. Vehicles moving to the back of the park are losing points and are registered with less points than those in the front of the park.

Figure 40: Apeldoorn park as used in the experiment

The heat map has been converted using the perspective transformation matrix as calculated for the Apeldoorn park. This allows a comparison between untransformed – as seen in Figure 41 – and transformed – as seen in Figure 42.

Figure 41: Heat map of Apeldoorn park without transformation.

***Note**: The coordinates of this heat map have been converted to match the translated heat map. This allows easy comparison of both maps.*

Figure 42: Heat map of Apeldoorn park with transformation.

Adding a perspective correction as in Figure 42 shows a better distributed number of detected points in the map. Especially the left side has been improved, allowing detection were the untransformed map has zero points. Having this better distribution also adds to a better reliability as more points can be tracked and thus more detailed movement is tracked.

6.2.3.2 Eindhoven office

Figure 43: Beukenlaan in Eindhoven

Another heat map has been generated of the Beukenlaan – as seen in Figure 43 – visible from the Alten office in Eindhoven. This road is four lanes of two way traffic. This viewpoint allows a generation of a heat map in a scene where cars drive a few hundred metre from the camera point. As illustrated in Figure 44 the heat map indicates a better detection of the right lane. This is due to the implementation of MEVS. The current system checks for feature point quality upon detection. As the points are tracked it keeps following all initially detected points. As cars get smaller in the visible frame points start to overlap and merge – without being filtered by the system. The left lane indicates how a group is formed, as it shows a few feature points at the top left corner. This is the minimum displacement time, minimum number of points and clustering combined. As seen on this lane the close vehicles move to the camera the more points are located and tracked. This lane shows the relation between detail in the image – size of the vehicle – and the number of tracked points.

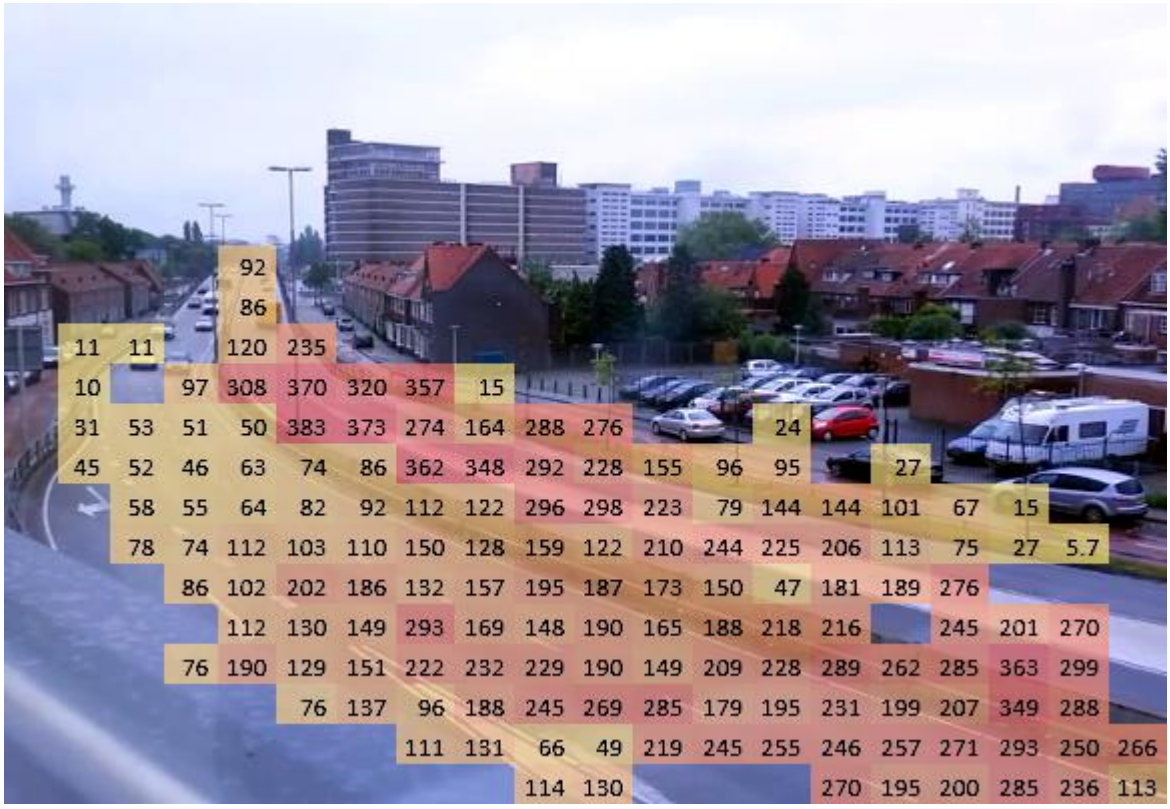


Figure 44: Heat map of Beukenlaan in Eindhoven

Note that the heat map in the top left corner is not completely valid. The system still required a minimum displacement per feature point which it can no longer detect due the distance between vehicle and camera. Some feature points were still visible but not tracked and thus not in the heat map.

Both the Apeldoorn and Eindhoven heat map indicate the requirement of an elevated camera position. The heat maps without transformation indicate much feature points can be located in the foreground of the image and less in the background, due the limitation of distance and displacement. Applying the transformation does help the tracker to detect vehicles, and results in a more distributed heat map.

6.2.4 Repeatability

The repeatability test is executed by tracking a fragment multiple times and drawing all tracks on one map. As seen in Figure 45 the start position – in the bottom left – and the first bend of the vehicle show more differences than the other parts of the track. This may be caused by the initial grouping of points. As the system needs a few frames to find all good feature points the centroid of the group – and thus vehicle – may not be completely stabilized.

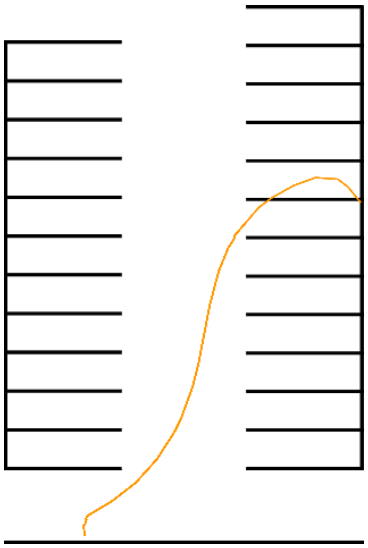


Figure 45: Drawing 5 recordings of fragment 1 captured in Apeldoorn.

7 Conclusion

MEVS is a flexible object tracker allowing one to follow people walking, vehicles moving and other objects passing by. This fulfils the assignment as written in Section 2.3. It tracks vehicles parking and does so reliably. As no filters are used to separate vehicle movements from human movements it tracks all people moving in a scene. This renders MEVS as a parking tracker less useful. Adding this filter is required in order to create a tracker for a specific purpose.

The tests of tracking in Section 6.2.1 indicate the improvement of combining the background remover and feature points. It allows more flexible detection as it clusters using the detected motion of the background remover and tracks using the feature points. This allows objects to move closely to each other while remain being detected. This shows the benefits of the new hybrid MEVS algorithm. However, the current implementation of MEVS only separates vehicles moving in opposite directions and having more than a certain distance. This results in tailing vehicles being tracked as one object.

The heat map as seen in 6.2.3 indicates the requirement for an elevated camera position. This allows better tracking due the more distributes number of feature points. Using a transformation to correct the perspective is improving the system much, but having an elevated position may add even more to the improvement.

The repeatability as tested in Section 6.2.4 shows that after 5 recordings of the same video the random initialization of feature points and grouping the points has minimal influence on the track. However, it must be noted that the experiment used a recorded video. This demonstrates the repeatability of the system in just one scene. More tests using live camera footage are required in order to completely test the repeatability and calculate the deviation per vehicle track recorded.

To answer the research question the sub questions are answered first.

How can displacement (velocity and direction) of an object be determined?

Using feature points the objects can be tracked in the videos used in the project. The displacement is determined by calculating the differences between two consecutive frames. Having the displacement the velocity and heading (direction) can be retrieved for all objects.

What is needed to avoid object identity switching?

MEVS uses the heading and distance between objects to separate them. The current implementation does not allow vehicles to tail each other within the maximum point distance. As this distance is not flexible it must compensate for larger vehicles to be tracked as one. This renders smaller vehicles to be grouped as one and tracked as one. To completely avoid object identity switching a new factor has to be added. This could be a flexible grouping based on velocity. Another factor could be disabling merging vehicles if they have been tracked for more than a predefined number of frames.

What is required to implement such a system?

The current implementation uses OpenCV for image processing. This framework allows use of all needed computer vision algorithms, such as the background segmentation, feature tracker and filtering options.

What is required to map motion in an environment using a camera system?

To map motion in an environment a motion tracker is required. MEVS implements such a system by combining two methods to create a new hybrid method. This method has improved initialisation versus the original Good Feature To Track method and better tracking than blob tracking. The system allows tracking of vehicles and has proved to do so in multiple tests, as written in Section 6.2. The current results were gathered using a proof-of-concept system. By further improving the system as written in the Recommendations in Chapter 8 the system may eventually be used as a tracker for the park in Eindhoven.

8 Recommendations

It is recommended to add filters to the output of MEVS. One could add the contours of the tracked object to each detected path. Applications using this data can filter based on the contour and separate vehicles and human. It is required to have some kind of filter prior to using this system as tracker for a parking lot.

A new method for grouping the feature points should allow the system to reliably track tailing vehicles. As written in 6.2.1 two vehicles tailing – moving closely behind each other – causes the system to fail and track them as one vehicle.

To improve the detection and tracking a better camera – allowing a wider angle and preferably less noise – should be used. The current used movies are captures with a mobile phone and contain a lot of noise. Having a wider angle allows tracking vehicles as they move into the park. Especially in Eindhoven a wider angle is necessary, as the gate is far to the side of the building. It is recommended to find an elevated location for the camera to have a better perspective for tracking. This is indicated by the position test in Section 6.2.3. The elevated position increases the reliability of each feature point and thus a better tracker.

It is recommended to improve the grouping by adding more factors to separate vehicles as they move on the same lane and same velocity. The current implementation cannot handle this case and tracks both vehicles as one.

More tests should be conducted to test the system in more various scenarios. Some more interesting tests could be:

Testing grey cars on the grey tarmac. As the implementation uses images converted to use the grey scale some coloured cars may be less visible. It is not tested to see how the system reacts on this kind of vehicles and whether or not they could be tracked.

Testing different camera systems. The current test cases are all using the same camera system. Using a better camera may allow one to use the MOG2 algorithm. This results in better detection of cars, as it clusters bigger groups and removes detected shades.

Testing on repeatability using more vehicles moving over the same path to calculate the mean deviation of each track. This allows calculating statistics on the reliability over time and the overall repeatability of the tracker.

9 Bibliography

AI-Absi, H. R., Sebastian, P., Dinesh, J., Devaraj, J., & Yoon, Y. Y. (2010). Vision-Based Automated Parking System. ISSPA: IEEE.

Alten Operations. (2014, May). Retrieved from Alten: http://www.alten.com/operations

Beymer, D., McLauchlan, P., Coifman, B., & Malik, J. (n.d.). *A Real-time Computer Vision System for Measuring Traffic Parameters.* Berkeley, California: University of California.

GSoC'14. (2014, May). *Project Proposals - RGB-D Flow*. Retrieved from PointClouds: http://pointclouds.org/gsoc/

KaewTraKulPong, P., & Bowden, R. (2001). An Improved Adaptive Background Mixture Model for Realtime Tracking with Shadow Detection. *2nd European Workshop on Advanced Video Based Surveillance Systems.* Brunel University, Middlesex: Kluwer Academic Publishers.

Modi, P., Morellas, V., & Papanikolopoulos, N. P. (2008). *Counting Empty Parking Spots at Truck Stops.* Department of Computer Science and Engineering, University of Minnesota.

Nourani-Vatani, N., Borges, P., & Roberts, J. (2012). A Study of Feature Extraction Algorithms for Optical Flow Tracking. Sydney: Australasian Conference on Robotics and Automation, University of Sydney.

OpenCV. (2014, March). *Feature Detection.* Retrieved from OpenCV Documentation: http://docs.opencv.org/modules/imgproc/doc/feature_detection.html

OpenCV Documentation. (2014, May). *Geometric Image Transformation*. Retrieved from OpenCV Docs: http://docs.opencv.org/modules/imgproc/doc/geometric_transformations.html#getperspectivetransform

True, N. (n.d.). *Vacant Parking Space Detection in Static Images.* San Diego: University of California.

Tschentscher, M., & Neuhause, M. (2012). *Video-based parking space detection.* Bochum: Institude for Neural Computation, Ruhr-Universität.

Welch, G., & Bishop, G. (2006). An Introduction to the Kalman Filter. Chapel Hill: University of North Carolina.

Wu, Q., & Yi, Z. (n.d.). *Parking Lots SPace Detection.* Carnegie Mellon University.

Zivkovic, Z. (2004). Improved Adaptive Gaussian Mixture Model for Background Subtraction. *ICPR.* Amsterdam: University of Amsterdam.