Department of Humanities
Institute of Cognitive Science

Bachelor's Thesis

An Evaluation Of Systems Comprising Sparse Lucas-Kanade Optical Flow For Motion Analysis In Head-Mounted Eye-Tracking Videos

by

Lars Berend Brandt
(larbrandt@uni-osnabrueck.de)

November 18, 2019

First Supervisor:
Dr. Matthias Temmen

Second Supervisor:
Prof. Dr. Gunther Heidemann

Acknowledgements

I would like to thank my first supervisor, Dr. Matthias Temmen, for the constructive, motivating discussions and my second supervisor, Prof. Dr. Heidemann, for the useful comments and remarks.

I would also like to thank in particular Inga Ibs for her never-ending support and for proofreading this thesis.

Additionally, I would like to thank Nadja Werner, Marlin Krüger and Paul Liebenow for proofreading parts of this thesis.

Abstract

In this thesis, an explorative study for extracting motion information from mobile eye-tracking videos is presented. Three key point detection algorithms, the sparse version of Lucas-Kanade optical flow and preprocessing steps are optimized and evaluated, to lay a basis for the classification of eye-movements without previously obtaining gaze information and a 3D-model of the eye. Data is provided by mindQ from the project eyeTrax, comprising 22 videos of 11 participants (one video per eye) and one video for the verification. The subjects were doing multiple, simple tasks in virtual reality in about 5 minutes. The comparison is done qualitatively, with a main focus on three properties. The quantity of the found key points, the robustness of the tracking by the Lucas-Kanade method over time and over different types of videos and the semantical relevance of the tracked regions in the videos. Finally, as a measure of verification, one of the found optimized systems is applied to detect blinks. The results show the ability of the optimized systems to extract motion information, for which however further refinement is needed.

Contents

1 Introduction

In this thesis, groundwork is acquired for accurate eye-movement classification. The focus lays on improving a medical application, precisely the project eyeTrax by mindQ (2018), in which the analysis of eye-movement is extracted to yield information of a person's possible malfunctions in oculomotor movement. These malfunctions are indicators of various impairments of the human brain, among others, mild traumatic brain injuries. So far, data is gathered by filming both eyes of a subject via two cameras built inside virtual reality (VR) glasses. The setup is assembled by adding the Pupil Labs Binocular Addon (Pupil Labs, Berlin) to Vive (HTC and Valve) VR glasses. Since the cameras are directly in front of the eyes, it is called mobile, or head-mounted eye-tracking. The subjects in the project eyeTrax complete minor visual tasks and by using the algorithms provided by the Pupil Labs' eye-tracker, a classification of eye-movements is obtained.
Eye-tracking is a large field, and various models and implementations exist. For example, extensive testing was done by Holmqvist (2017) for 12 mobile and stationary eye-trackers. In particular, the head-mounted Pupil Labs eye-tracker was compared to a stationary one, the Eyelink 1000, by Ehinger et al. (2019). Both studies show, that eye-trackers are not accurate enough in classifying eye-movement, e.g. blinks, to conduct a medical diagnosis. A possible reason for this is the majority of fields, which use eye-trackers for studies, are firstly interested in gaze-tracking and only incidentally in the movement itself (e.g. see Enriquez et al. (2010)). Three major issues can be derived from papers of classical gaze-tracking. One being the loss of accuracy concerning some changes in circumstances, as differing eye-color, the wearing of mascara or even the body-size (Holmqvist, 2017, pp. 19). This issue is especially problematic for the application for medical diagnosis, as in this project, since all people need to be treated equally. Also problematic is the categorization of blinks (Ehinger et al., 2019, p. 36). The software from the Pupil Labs eye-tracker for example relies on the absence of a fitted ellipsis for the tracked pupil in gaze-tracking to detect blinks. In Ehinger et al. (2019), blinks were not sufficiently detected by the Pupil Labs blink detection. Thirdly, many mobile gaze-trackers do not account for disruptions like head-movements and as a result, false data or none at all is returned (Enriquez et al., 2010, pp. 1). More on this below.
Other approaches to detect blinks exist without the reliance on gaze-tracking. For example Fogelton and Benesova (2016) developed an algorithm which proved well in detecting blinks in their own and other datasets of stationary eye-tracking. They manually annotate eye-regions and analyze motion obtained by Gunnar-Farneback optical flow in these regions with a state machine. The reasoning to test a different approach is the following. The ultimate goal of the project, to which this thesis is associated, is to detect general movement in eye regions and not only blinks, but also saccades or smooth pursuits. Also, the data in this project is recorded by mobile eye-trackers, and therefore different than the datasets tested in Fogelton and Benesova (2016). The characteristic of stationary eye-trackers, comprising those datasets, is their distance from the eye, usually the whole face or more of the subject is visible in the videos. On the contrary, cameras of mobile eye-trackers are positioned directly in front of the eye, or, like in this case, both eyes. Accordingly, the data gathered is differing to stationary eye-trackers. Among other things there are large

differences in processing of head-movements for stationary and mobile eye-trackers. On the one hand, the latter can be regarded as a better setup, since the cameras are mounted to the subject's head and head movement will not lead to a loss of visibility of the eyes in the recordings, which is an issue for stationary eye-trackers. If the subject's back is turned to the camera, eye-tracking is impossible. On the other hand, problems arise if the mobile eye-tracker itself is disrupted spatially, e.g. by a fast head-movement or a loosely fit strap. Many gaze-trackers do not account for disruptions like this and as a result, false data or none at all is returned (Enriquez et al., 2010, pp. 1). For stationary eye-trackers, this is a minor problem, since the risk to move it is minimal and if it does, it is more probable to see it while testing. A final reason to test a differing approach to the one by Fogelton and Benesova (2016), is that their data is manually annotated and samples, where the whole eye is not visible, are discarded. Even though it is imaginable to make improvements to their algorithm to tune it to the task at hand, the decision was made to come up with another system to analyze motion in eye-videos, because of the differences above. In addition to a series of computer vision algorithms, which are not included in this thesis, this can possibly yield a system to classify eye movements as accurate as is expected for the appliance of medical diagnosis.

Similar to the algorithm of Fogelton and Benesova (2016), this thesis will lay the ground work to an analysis for eye-movement classification. The central point is to optimize the sparse version of Lucas-Kanade optical flow, which can be used to extract information about movement in videos (Lucas et al., 1981). For the purpose of developing a system of algorithms that extract robust motion data from the videos taken by head-mounted eye-trackers, it is paired with three key point detection algorithms, which were chosen based on the evaluation by Gauglitz et al. (2011) for tracking of key point detectors and descriptors. While not applied to eye-movements, their comparison included six detectors and a large dataset with manually annotated recordings as a ground truth for different kinds of movement. The focus of their work was the score of repeatability, which reflects the probability to find a key point at the same location in consecutive frames after movement in the image sequence, i.e. rotation, translation or affine changes (Gauglitz et al., 2011, p. 346).

To extract motion information from the detected key points, their locations are tracked by applying the sparse version of Lucas-Kanade optical flow, which has been shown to be one of the best algorithms for motion analysis (see Galvin et al. (1998) and Barron et al. (1992)). The main part of this thesis is an explorative study of the optimization of these algorithms, their parameters and preprocessing steps. Due to time- and hardware-constraints, not all settings of parameters can be tested, and is accordingly done in four consecutive steps. At the end of each step, the evaluation of the system is done and the best setting is chosen for the next step of testing. The measures consist of how long the found points can be tracked and how many there are in these steps. Afterwards, a prototypical evaluation for blink detection is done. Additionally, an algorithm to remove flickering brightness, visible in some of the data, is presented and tested.

The structure of the thesis is as follows: First, in section 2 and subsection 2.2, an introduction to optical flow in general and in particular the Lucas-Kanade method is given. Following is an introduction to key point detection and the three detectors will be explained in comparison in subsection 3.2. Furthermore, the dataset and

steps for evaluation are presented in subsection 3.4, followed by the results and their discussion in section 4 and section 5.

2 Optical Flow

In this section, optical flow in the field of computer vision is defined and problems, which arise when trying to calculate it, are shown. Optical flow is the apparent velocity of everything visible in a scene caused by the relative motion of observer and the scene, and can therefore yield important information about movement, the location of objects and overall changes in the scene (Horn and Schunck, 1981, p. 185). For classifying eye-movement, it is important to receive more information about motion in the image sequence. To obtain this, the general way to calculate optical flow of consecutive frames in a video is explained and the aperture problem is shown. Followed by the method to solve these problems used in this thesis, the Lucas-Kanade method.

2.1 Calculating Optical Flow

To determine optical flow in image sequences in a video, one has to calculate the velocity of a 3D world on a 2D image plane. Take the human visual system as an example, in which the retina is the 2D image plane. By using different cues on this plane, one can infer the movement of the 3D world. Optical flow is part of these cues, namely the movement on the 2D plane, and can be adopted to computers in form of a mathematical formalism, which will be explained in the following in the style of Horn and Schunck (1981).
First, a constraint is assumed: The brightness of a pixel does not change when moving through time and space (Horn and Schunck, 1981, p.187). Under this assumption the movement of a pixel in a video can be viewed as follows:

$$E(x, y, t) = E(x + \delta x, y + \delta y, t + \delta t)$$

With E(x,y,t) being the brightness of a pixel at location (x,y) in the image plane at time t and δx, δy and δt referring to the change in these three dimensions.

Now, the first part of the Taylor series can be used to expand the right hand side:

$$E(x, y, t) = E(x, y, t) + \delta x \frac{\partial E}{\partial x} + \delta y \frac{\partial E}{\partial y} + \delta t \frac{\partial E}{\partial t} + \epsilon$$

This formula can be rearranged to

$$\delta x \frac{\partial E}{\partial x} + \delta y \frac{\partial E}{\partial y} + \delta t \frac{\partial E}{\partial t} + \epsilon = 0$$

Where the partial derivatives (∂) contain the gradient information in x, y and t direction, respectively and ϵ yields the error term, since only the first Taylor expansion is taken.

Division by δt yields:

$$\frac{\delta x}{\delta t} \frac{\partial E}{\partial x} + \frac{\delta y}{\delta t} \frac{\partial E}{\partial y} + \frac{\partial E}{\partial t} + \lim_{\frac{\epsilon}{\partial t} \to 0} = 0$$

Given $u = \frac{\delta x}{\delta t}$ and $v = \frac{\delta y}{\delta t}$ and the error term getting close to 0, the *optical flow equation* can be derived:

$$E_x u + E_y v + E_t = 0$$

The problem of determining optical flow becomes apparent here, a single equation with two unknown variables u and v is unsolvable. This is also called the aperture problem of optical flow. Semantically this problem is comparable to a human who lost one of his eyes and therefore struggles to derive motion information correctly. Several methods to work around this problem exist, one of which will be discussed in the following.

2.2 Lucas-Kanade Method

To solve the aperture problem Lucas et al. (1981) introduced an additional constraint: The pixels in the neighborhood of a pixel are assumed to move identically to it. In this way, the *optical flow equation* becomes solvable, since we can solve for u and v, with 5 equations, given a 4-neighborhood, or 9, given an 8-neighborhood or even larger number of equations for larger neighborhoods of a target pixel. Lucas et al. (1981) apply the least-squares method to solve this overdetermined system of equations for u and v. In other words, to calculate the location of a part of a 3D world, depicted by pixel A_t and its neighborhood at time t, their intensity is compared with the same pixel $A_{t+\delta t}$ and its neighborhood at the next timepoint $t+\delta t$, now depicting a different part of the world. Using the gradients in the neighborhood of the point it is possible to estimate the velocity of movement, u and v, and therefore the new location of the point in space at pixel $A'_{t+\delta t}$ (Rojas, 2010).
Problems with this method exist. Firstly, in a real-world scenario, pixel intensity can change because of differences in lighting or noise. This can lead to inconsistencies of the neighborhood constraint. Secondly, occlusion and therefore disappearing points in space do pose a problem in all movement detection applications. This problem occurs particularly in this project, since subjects blink during eye-tracking recordings. Last but not least, this method is purely local, which means that only small patches of the frames (neighborhood) are taken into account and not the whole image. Therefore, only small motions relative to the framerate are tracked.
Though these problems occur, with certain adaptions of the algorithm it proves to be one of the best performing optical flow methods (see Galvin et al. (1998) and Barron et al. (1992)). One of these adaptions is the removal of noise by a suppression of high spatial frequencies. Noise is changing from frame to frame and can lead to false movement detection. However, low-pass filtering also suppresses small details and a trade-off has to be found (Lucas et al., 1981, p. 123), more on that in subsection 3.3. To account for the ability to track large (relative to image size) or fast (relative to frame rate) motions, both starting image and consecutive images can be reduced by downsampling. This transform summarizes a portion of neighboring pixels via interpolation and yields an image with a smaller resolution. This step can be repeated to estimate every size of movement, without changing much of the actual Lucas-Kanade algorithm (Bouguet et al., 1999). As an example one can imagine two balls rolling over the ground, one fast and one slow. The slow one is rolling in the video at a speed of 5 pixels per frame. The fast one at 20 pixel per frame. To detect movement in the image, a point of both balls is tracked with

Lucas-Kanade optical flow. Because the detection in this example is only possible in a small neighborhood of 10 pixels, the slow ball can be tracked, but the fast ball is not in a region of search for the algorithm. Now, downsampling takes place. Since the balls are much brighter than the dark background in the videos, this reduction of resolution does not pose problems for the detection of the balls and the fast ball is now moving only with a speed of 8 pixels per second. Now the Lucas-Kanade method is able to track this ball as well.
One can choose to determine the optical flow of every pixel in the image or use a number of key points to determine it in salient regions of the image, also called dense and sparse optical flow, respectively. In this thesis, the latter is being used, because not every pixel is valuable or useful for tracking, ergo computational time can be saved (Shi and Tomasi, 1993). Key points are also called salient points, features, or regions of interest. The goal of their detection is to determine points in images which are 'important', differentiable or discriminative in an image patch and therefore can characterize the image as a whole or parts of it. Key point detection is an essential part of computer vision research and not only so for motion analysis, but also for object detection, camera calibration, 3D reconstruction and pose estimation (Tuytelaars and Mikolajczyk, 2008, p. 179). In order to achieve the best results with the sparse Lucas-Kanade method, the key point detection has to be optimized. For this reason, multiple key point detection algorithms are compared and optimized in this thesis. A vast amount of these detectors exist and can be grouped roughly in the following classes, as was done by (Gauglitz et al., 2011, pp. 337):

- *Corner detectors* compare candidates for interest points to their neighbors. It becomes apparent, that a strong local extremum in x-direction in the derivative image yields pixel laying on an edge. If this is accompanied by an extremum in y-direction as well, one can see this as a corner, which are regions that have been viewed as interesting in the past (Gauglitz et al., 2011, p. 337). One of the first of these detectors was the Moravec operator (Gauglitz et al., 2011, p. 337), which shifts a window around a key point candidate and measures the difference between the pixel values. An improvement is the widely used Harris corner detector (Harris et al., 1988), and, as a variation from it, the Shi-Tomasi corner detector (Shi and Tomasi, 1993). The latter will be used in this thesis.
- *Blob detectors* use extrema of various filters to determine if a region is of interest for further analysis. These regions can be ascribed to an anchor point, most often the center pixel, to yield key points (Gauglitz et al., 2011, p. 338). Examples are the fast Hessian (Bay et al., 2008) and the difference of Gaussian DoG) (Lowe, 2004) detectors, which use simple box filters to approximate the Hessian matrix, and Gaussian filters to find key points invariant to scale, respectively. Both of these will be part of this work.
- *Affine-Invariant detectors* have been proposed to yield key points, which are robust to affine changes. These changes include rotation, translation and scaling. Most of these detectors have a very high computational cost and will therefore not be represented in this work.

These groups are not strictly defined, some detectors use methods of more than one class to achieve more stable key points. The difference of Gaussian approach

for example is seen as a blob detector, but makes use of corner detectors to reject candidates found with the blob detection. More on that in the following section, where the detection algorithms used in this thesis are explained in further detail.

3 Sparse Lucas-Kanade Optical Flow for Analysis of Eye-Movements

In the following sections a method to search for the best setup for eye-movement analysis is carved out. The setups consist of preprocessing, key point detection and Lucas-Kanade optical flow. Firstly, these setups are explained in further detail. Secondly, in subsection 3.2, key point detection in general is presented and the algorithms used in this work are explained in further detail. Thirdly, in subsection 3.4, the structure for evaluation of these setups are defined. Lastly, the testing procedure is presented.

3.1 General Setup

The sparse version of the Lucas-Kanade method calculates the optical flow on a set of key points in an image. The whole sequence of the system is visible in Figure 1. Before the actual detection, preprocessing takes place to improve findings of key point detectors by removal of noise. The located key points found in the first frame function as input for the optical flow calculation. Based on the Lucas-Kanade method, the locations of these points are estimated in the next frame. If it is possible, the optical flow method can use the locations as new input for the next frame, repetitively. The computation of new locations of key points is called *tracking* in this thesis. The tracking of a point stops, if the Lucas-Kanade method can not estimate the new location of a point in the next frame. Furthermore, every 10 frames, additional key points are detected, with exactly the same method as in the first frame. To reduce false findings of the optical flow algorithm, a backwards check is implemented: The locations being estimated by the forward step serve as input for the estimation of original locations. If the estimation of the original location differ from the original location by more than 1 pixel, the tracking for this key point stops.

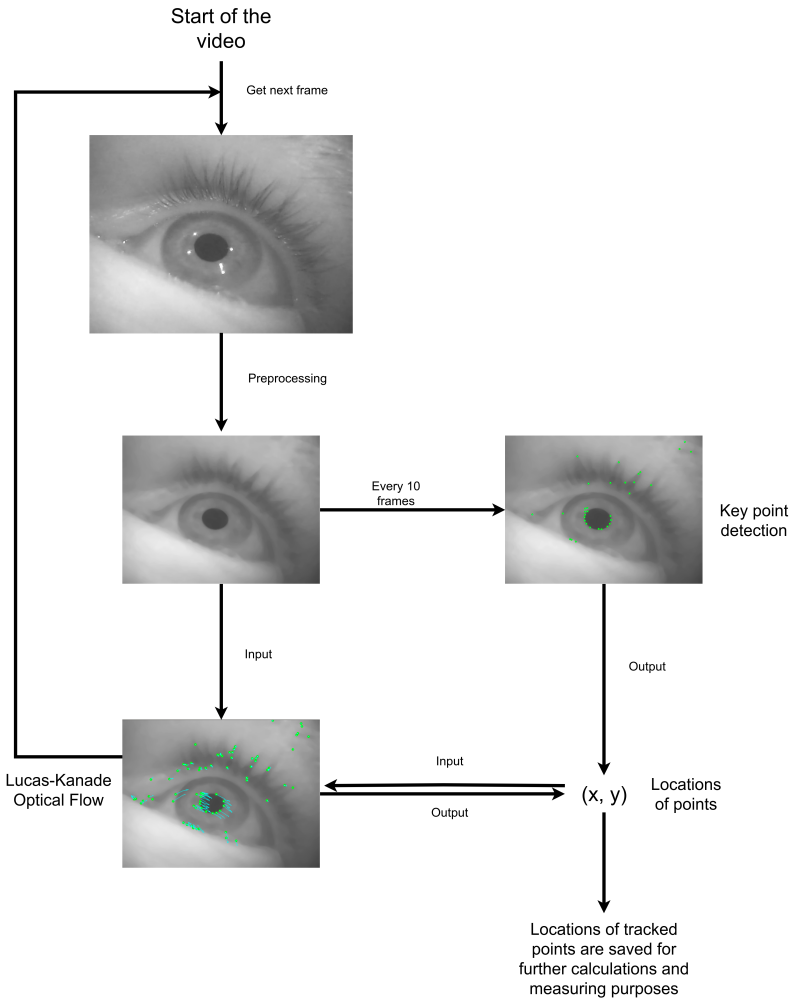


Figure 1: Sequence of the system. Every frame is preprocessed with resizing and filtering. Every ten frames, key points are detected with one of the methods. Key point locations are input for the Lucas-Kanade method, which calculates the locations of these key points in the next, preprocessed frame.

3.2 Key Point Detection

A comparison of key point detectors and feature descriptors is done by Gauglitz et al. (2011) for visual tracking. It includes six interest point detection algorithms, chosen because of their frequent use and previous tests. Based on their evaluation, three were chosen for this thesis.

3.2.1 Shi-Tomasi

The key point detector developed by Shi and Tomasi yields corners. It can be viewed as a variation of the Harris corner detector (Harris et al., 1988). Shi and Tomasi (1993) developed it by analyzing which key points are good for tracking. In general, the neighborhood is viewed and the change in intensity is calculated with help of a structure tensor, which comprises this information in form of a symmetric 2 x 2 matrix (Gauglitz et al., 2011, pp. 342):

$$M(x,y) =$$

$$\begin{bmatrix} \sum\limits_{u,v} w_{u,v} \cdot [I_x(x+u,y+v)]^2 & \sum\limits_{u,v} w_{u,v} \cdot I_x(x+u,y+v) I_y(x+u,y+v) \\ \sum\limits_{u,v} w_{u,v} \cdot I_x(x+u,y+v) I_y(x+u,y+v) & \sum\limits_{u,v} w_{u,v} \cdot [I_x(x+u,y+y)]^2 \end{bmatrix}$$

Where $M(x,y)$ is the structure tensor at location (x,y) in the image I. I_x and I_y denote derivatives of the image in x and y direction, respectively. $\sum\limits_{u,v} w_{u,v}$ is a window and Gaussian weighting function, so the pixels in the neighborhood of I(x,y) are taken less into account, the further they are from the center. Based on the eigenvalues λ_1 and λ_2 of $M(x,y)$, the image patch can be classified as uniform (both eigenvalues small), edge (λ_1 small and λ_2 large, or vice versa) or a corner (both values large.) Corners localized by this algorithm yield supposedly good features to track (Shi and Tomasi, 1993, p. 3).

3.2.2 Difference of Gaussian

Difference of Gaussian is the key point detection algorithm used in Scale Invariant Feature Transform (SIFT) (Lowe, 2004). This blob detector extracts key points which are invariant to scale. For this, an image is convolved with Gaussian kernels, having differing standard variations leading to various blurred versions of the image. These are also called octave layers, with octaves being down-sampled versions of the input image, in this case, by taking every second pixel in each row and column. The process of convolution with Gaussian kernels is repeated on these octaves. Next, differences of Gaussians are achieved, by subtracting each octave layer from the next higher layer. Figure 2 depicts this process.

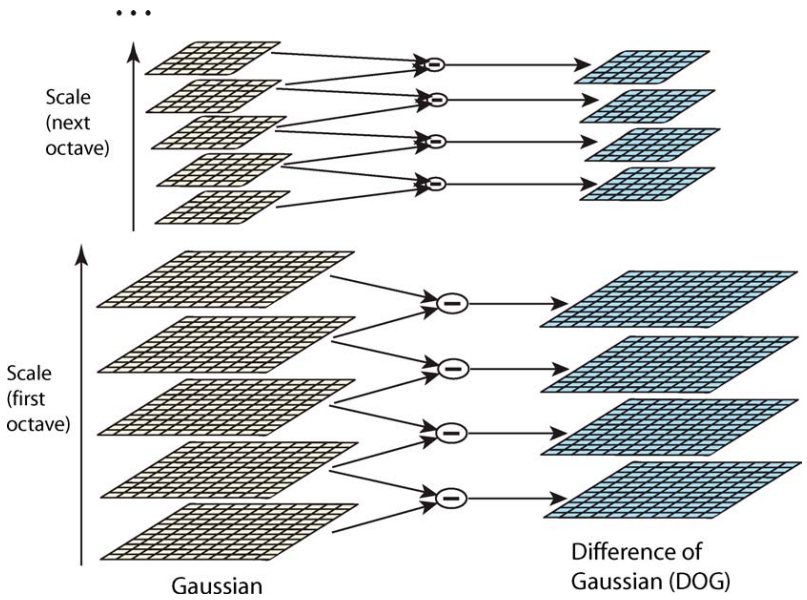


Figure 2: Difference of Gaussians is achieved by repeated convolution of the input image with Gaussian kernels and subsampling. Then, the octave layers are subtracted from each other (Lowe, 2004, p. 95).

Candidates for key points are those pixels, which yield local maxima compared to the 8 neighbors in its own difference of Gaussian and the 18 in the ones above and below it, as shown in Figure 3. In further analysis, candidates with low contrast and points along an edge in the image are discarded, with an approach similar to corner detection (Lowe, 2004, pp. 94-98).

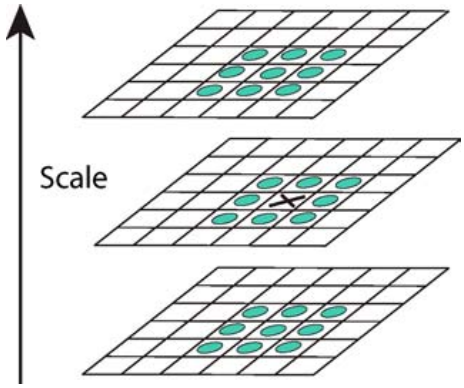


Figure 3: Candidates for key points are local extrema in comparison to their neighbors in the same, and neighboring octave layers (Lowe, 2004, p. 95).

3.2.3 Fast Hessian

Another Blob detector, the fast Hessian, was proposed by Bay et al. (2008) for Speeded Up Robust Features (SURF), which was developed to enhance SIFT. Hessian detectors select key points by use of the Hessian matrix, which is obtainable by convolution of the image with Gaussian second-order derivatives. Since this is very costly, the convolution in the fast Hessian detector is approximated by simple box filters, as shown in Figure 4, which can be computed in constant time using the integral image (Viola et al., 2001).

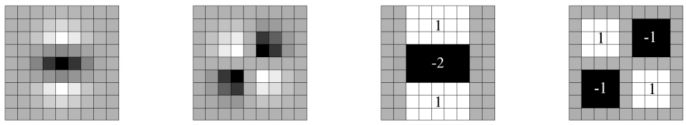


Figure 4: Second order Gaussian derivatives (left) are approximated as box filters (right) to receive an approximation of the Hessian matrix (Bay et al., 2008, p. 348).

The scale-invariance is achieved by upscaling these filters, as can be seen in Figure 5 and key points are chosen based on the highest determinant of the resulting, approximated Hessian matrices in their corresponding scale. Similar to the difference of Gaussian approach, candidates of low contrast are rejected (Bay et al., 2008).

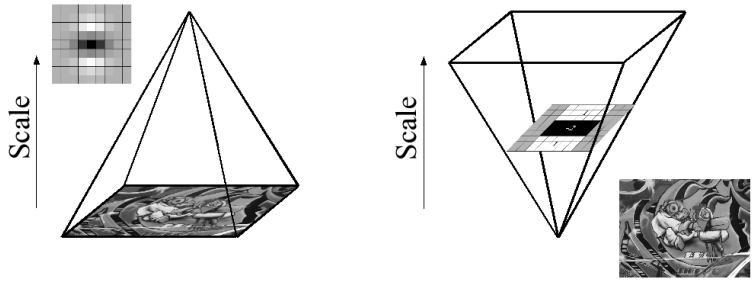


Figure 5: As opposed to the upscaling being done in key point detection with the difference of Gaussian approach (left), the image is not subsambled, but the box-filter is upscaled to include different scales (right) (Bay et al., 2008, p. 349).

3.3 Preprocessing

To improve the accuracy of movement detection it is of importance to remove irregularities from the data. Techniques to process the videos before detecting key points alter the findings significantly and their parameters are therefore a part of testing. The frames are filtered with Gaussian or median kernels and resized via interpolation. This reduces noise and also possibly destroys larger structures seen in the image, depending on the kernel size. This could lead to a better measurement of motion, because these are unmoving. The smaller reduction of resolution and filtering leads to a larger difference of pixels beside each other and therefore possibly more stable key points to find (Bouguet et al., 1999, p. 2). To test if the flicker, as explained in subsubsection 3.4.1, has notable impact on the performance of the tested motion analysis, a procedure to remove this flicker is applied. Midway equalization, as shown by Delon (2004) takes the histogram of gray values from two or more consecutive frames and calculates an equalized version, changing the values of the single frames.

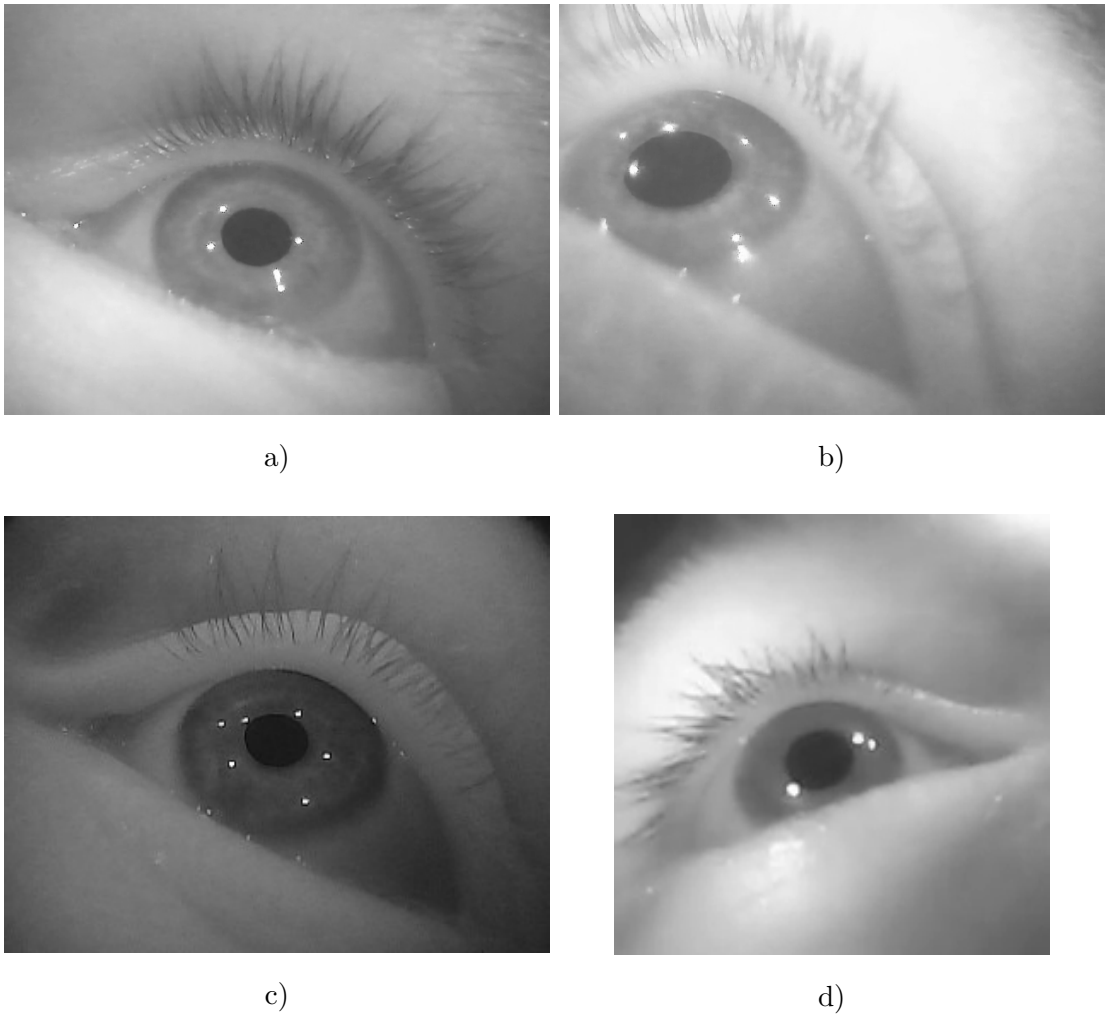
3.4 Setup for Evaluation

In the following subsections, a setup to optimize the analysis of eye-movements using preprocessing, the three methods to detect key points and tracking of these by Lucas-Kanade optical flow is presented. In the beginning, the dataset is explained, followed by the three qualitative measures used. Lastly, one quantitative measure for evaluation is presented.

3.4.1 Dataset

A dataset was taken in which subjects are filmed solving small tasks in virtual reality. Two cameras are located in the virtual reality glasses and pointed at the subject's left and right eye. Three different sets of videos exist. 10 subjects were filmed doing the newest version of the test, with an average video length of 5 minutes. The frame-rate corresponds to 120 fps and the videos have a resolution of 640 x 320 pixel. One subject was recorded doing the same test, but at a frame-rate of 200 fps and a resolution of 192 x 192 pixel. Eight subjects were recorded doing an older version of the test in virtual reality, which takes about 3 minutes, also at a frame-rate of

120 fps and a resolution of 320 x 240 pixel. The last dataset was annotated with timestamps of blinks by Gundler (2018). Examples for each dataset can be seen in Figure 6. Even though the cameras are mounted to the subject's head, videos contain not only eyes, and not always every part of the eye. As seen in Figure 6 b), both canthi, the corners of eyes where the eyelids meet, are not visible in every video. Furthermore, unmoving, strong structures can be seen in Figure 6 a) in the upper right corner.



a) b) c) d)

Figure 6: Variations of data: **a)** and **b)** 640 x 320 @ 120 fps. **c)** 320 x 240 @ 120 fps and **d)** 192 x 192 @ 200 fps. Rotation and camera angle are not identical, e.g. in **b)** only one canthi is part of the image.

Beside this and noise in the recordings, another disruption of videos is visible. One infrared light appears to be turning on and off, leading to a so-called flicker in large parts of the images, as can be seen in Figure 7. Even if it is not a very large dataset, it enables a lot of research with a wide variation of data. To reduce computational time, not all frames in the videos were used for the analysis. Only the first 10.000 frames were taken, which also simplifies the need for accounting of differing length of videos for relation.

a) b)

Figure 7: Flashing of light leads to a 'flicker'. Consecutive frames vary in brightness.

3.4.2 Measures

In the following section, the qualitative measures used to validate the setups for calculating optical flow are compared. In this thesis, qualitative measures are those, which do not rely on ground truth, e.g., it is not determined, whether the optical flow of the eye being calculated is the correct one, by measuring it in another way in or via human validation. As opposed to such quantitative measures, qualitative ones, like the number of key points, the length of tracking of these and their distribution can be helpful to compare these methods in relation to each other.

Number of Key Points

To have a basic measure to compare detection algorithms, found key points are counted. This basic measure is important, because a very small number of key points leads to small computational effort for the optical flow method, but one can argue that a small change in the setting (lighting, shift of camera etc.) leads to problems for the optical flow algorithm to track these points. This is also true, if a lot of points are being tracked, but in that case, the chance is higher to have additionally a few traceable points. For one, a large number of key points is computationally suboptimal. Also, it is most often paired with a short lifespan of key points and therefore not desirable. Accordingly, a medium number of key points is desired and a threshold of a minimum of 500 tracked points on average per 10000 frames of each video was set. Settings with large numbers of key points are automatically rejected with the following measure.

Lifespan

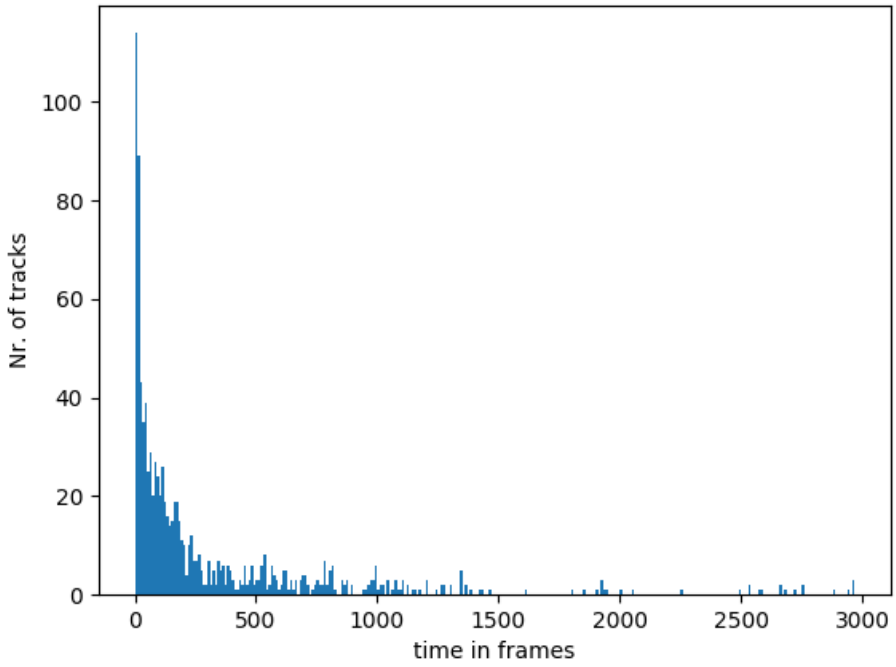


Figure 8: Histogram of quantity of key points being tracked over time in frames on one 120 fps video. Sum of tracked points = 960. Average lifespan = 303.525. Shi-Tomasi parameters: qualityLevel = 0.9, minDistance = 1, maxCorners = 260, blockSize = 5. Lucas-Kanade parameters: winSize = (31,31), maxLevel = 3, terminationCriteria=(3, 4, 0.5).

A second measure for comparison is how long sparse Lucas-Kanade optical flow can track the found key points, the so-called lifespan. In particular, it supplies the sum of how often the Lucas-Kanade optical flow can calculate the location of a point in the following frame. This can easily be counted and is essential for comparison. First in each step, the lifespan for all key points is averaged over all first 10000 frames of the videos with 120 fps and 200 fps, respectively. Then, after thresholding with the number of key points, the one setting with the highest average lifespan is selected, and proceeds to the following steps of testing. In Figure 8 the data is visible for a part of a video comprising 10000 frames and using the Shi-Tomasi corner detector. The data is rounded to decades. in this example many key points can no longer be tracked after a few frames, hence the peak around 0.

Spatial Distribution of Key Points

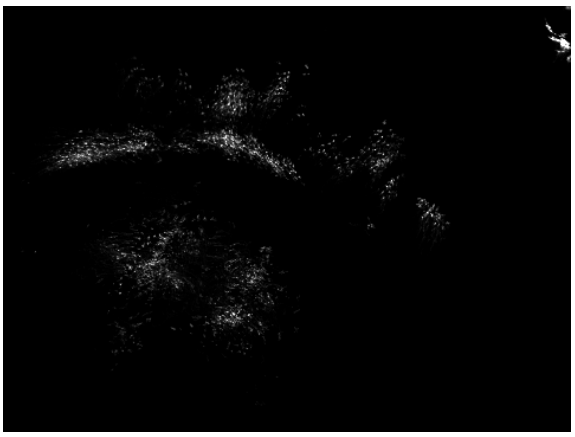


Figure 9: Heatmap of duration of tracking in image plane. The brighter the pixel, the more often it was a key point location. Shi-Tomasi parameters: qualityLevel: 0.5, minDistance = 61, maxCorners = 260, blockSize = 5. Lucas-Kanade parameters: winSize = (31, 31), maxLevel = 3, terminationCriteria=(3, 4, 0.5)

A third qualitative measure for the detection algorithms is where key points are located. Included is the location of those, which are found by a detector, but also the locations which have been estimated by the optical flow algorithm. Figure 9 shows an example heatmap which presents this information. By comparison with the actual images in the video it is possible to subjectively decide whether the locations are semantically relevant. Distributions over the whole image are desired, especially so over moving parts, as this is information directly valuable for motion detection. But also key points tracked at generally unmoving parts are important because if movement is measured there, a disruption of world parameters is likely, e.g. a change in the location of the camera or lighting conditions.

It is important to carefully examine the measures above with respect to each other. A very high lifespan is a generally desirable result, but if there are just a few key points found, which are all located in unmoving parts of the video it is still not a good outcome. The same can be said about a desirable distribution of key points over the images with a very small average lifespan, since it signifies that key points can not be tracked well with the optical flow method.

3.4.3 Blink Detection

The only quantitative measure is to test, if one of the setups can yield insight to motion. Accordingly, the location of every tracked point is noted and additionally the average magnitude and direction of all key points between one frame and the next is measured. Visualization for interpretation can be accomplished by a heatmap histogram of directions (HoD). In it, every frame is a slice with 8 bins of directions, measured in degrees. Direction is calculated by drawing an imaginary circle, the origin being the point in the first frame. If the location of the estimated point in the next frame is to the below of this origin, it counts towards the average of 0°. Same for other directions, running counter-clockwise on the circle, 90° implies the new location is to the right, 180° upwards and by 270° the movement took place in left

direction, as can be seen in Figure 10. This calculation yields a heatmap, in which for every two frames, movement of the key points is visible in direction and magnitude. Therefore, interpretation and even a simple classification of eye movement could be possible.

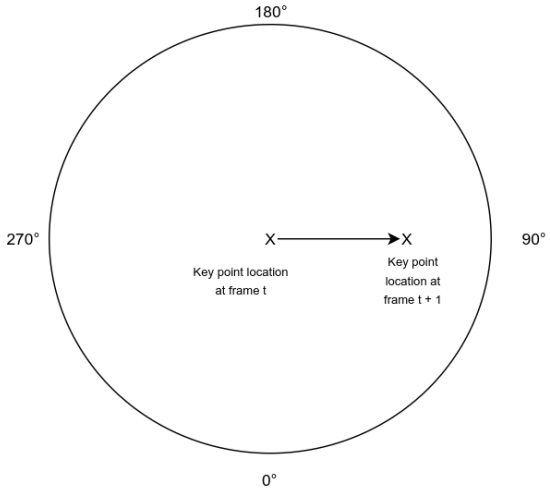


Figure 10: Diagram to demonstrate the instrument to calculate the direction of the movement of tracked points. A motion of point x to the right direction is visible.

3.5 Testing

In this section, implementations to gain information to measure, as explained in subsection 3.4 are presented. The machine is working with Python 3.7.4 and OpenCV 4.1.1. To be tested is the following: parameters for detection algorithms, parameters for Lucas-Kanade optical flow and preprocessing. Because the computational load would explode by testing all combinations at once, not every configuration of parameters is going to be tested. Rather, three steps are applied: First, detector parameters are analyzed with a fixed framework of optical flow and preprocessing steps. Second, the framework is set with fixed parameters for the detection algorithms proven to be best and fixed preprocessing and optical flow is tested with these. In a third step, preprocessing is evaluated with fixed optical flow and key point detection. This should approximate the best setup for this application.
For verification, a Hheatmap histogram of directions (HoD) is calculated with this setup. The strongest up and down movement, visible in it, should yield blinks at that point in time. This is performed by the author. Furthermore, the removal of flicker is tested in two videos, one with a frame-rate of 120 fps and one with 200 fps. Due to time constraints, this prototypical test in form of a direct comparison with the same videos without histogram equalization must serve enough data for an evaluation of this algorithm.
What follows is a listing of parameters for the applied algorithms with a short explanation. The tested values for parameters are given in parentheses, and educated guessed values are printed in **bold**.

3.5.1 Shi-Tomasi Corner Detector

The Shi-Tomasi detector is implemented in the version of OpenCV in use as cv2.goodfeaturestotrack() and users can change it with the following parameters (Itseez, 2019):

- maxCorners. (10, 260, 510) This is the maximum number of returned key points.
- qualityLevel (0.1, 0.5, 0.9). A number between 0 and 1, which determines how strong the measure in a point has to be. All candidates are compared with the best one found. To be considered a key point, the corner measure has to be at least 90 percent as good as the best one found, if the qualityLevel parameter is set to a value of 0.9.
- minDistance. (1, 31, 61) The minimal Euclidian distance between points to be considered a key point.
- blockSize. (2, 3, 5) Size of the neighborhood included by the structure tensor.
- mask. This parameter is not tested due to time constraints. It specifies a region of interest for detection. No key points are detected for 5 pixels around the locations of tracked points in the last frame.

3.5.2 Fast Hessian

The fast Hessian detector is bound to SURF in OpenCV. As only the detection of key points is needed, a SURF object is created and key points are detected with cv2.xfeatures2d.SURF_create().detect() (Itseez, 2019). The following parameters were tested:

- hessianThreshold (50, 100, 200, 500). The main part of key point detection with this detector is the threshold for the determinant of the approximated Hessian matrix.
- nOctaves (1, 3, 5, 7). How often the first filter applied to the image is upscaled to yield different octaves.
- nOctaveLayers (1, 3, 5, 7). How often the filter is upscaled at each octave.

3.5.3 Difference of Gaussian

The Differences of Gaussian detector is bound to SIFT in OpenCV. Same as for SURF, only the key point detection is needed, it is evoked by calling cv2.xfeatures2d.SIFT.create().detect() (Itseez, 2019). Parameters concerning key point detection are listed here:

- nfeatures (10, 250). The number of best key points to hold on to, ranked by local contrast.
- Octaves (1, 5, 3). The number of times the input image is downsampled.

- nOctaveLayers (1, 3, 5). The number of times the Gaussian convolution is applied to each octave with higher standard deviation to yield more and more blurred versions of the octave. Returns difference of Gaussian by subtraction to the function. Differing standard deviations are calculated automatically by the resolution.
- contrastThreshold (0.01, 0.04, 0.1). This threshold rejects key point candidates which have a low contrast.
- edgeThreshold (2, 10, 15). This threshold rejects key point candidates which are located on edges, and not corners.
- sigma (1.0, 1.6, 2.5). The standard deviation applied to the first octave of the input image.

3.5.4 Lucas-Kanade Optical Flow

The Lucas-Kanade optical flow method is implemented in the OpenCV library as cv2.calcOpticalFlowPyrLK() and uses the following parameters:

- winSize ((5, 5), **(15, 15)**, (31, 31)). The window in which the algorithm searches in the new frame for the new location of the key points, on all levels.
- maxLevel (0, 3, **10**). The maximum number of levels of reduction to use.
- criteria: Identifier, cv.TERM_CRITERIA_EPS | cv.TERM_CRITERIA_COUNT ((3, 4, 0.5), **(3, 10, 0.03)**, (3, 30, 0.03)). The number of the identifier specifies the termination criteria to stop the search algorithm, either by number of iterations (COUNT) or after the search window moves by less than epsilon (EPS). Both of these criteria are used, when setting the identifier to 3.
- minEigThreshold (**0.0001**). To account for noise, the minimal eigenvalue of the system of equations for optical flow calculation is set with this value. Due to time constraints and since noise is also reduced with tests in the following section, this value is not tested and the default value is used.

3.5.5 Preprocessing

To test the improvement of precision by detection and tracking of key points by altering the preprocessing steps, the following steps are taken into account:

- Downsampling by bilinear interpolation, as is used by cv2.resize() to a resolution of the original images' width and height to a fraction of them. ((0.5, 0.5), **(0.75, 0.75)**, (1, 1))
- Noise-removal: The influence of Gaussian ((5, 5), (15, 15), (21, 21)), as well as median filtering ((5, 5), (15, 15), **(21, 21)**) is being tested. The latter mainly to destruct medium-sized, unmoving structures (i.e. eyebrows) to focus key points on moving, large objects (pupil, iris, eye-lids).

- Deflickering (Midway equalization). Key point detectors are tested prototypically on two videos with removed global flicker, to examine possible improvements for the optimized system.

4 Results

In the following section, the test results are presented. First, an evaluation of each step, the parameters for each key point detection algorithm, Lucas-Kanade optical flow and preprocessing are analyzed. Here, the main focus lays on the low level measures, average lifespan in frames and number of tracked points over the first 10.000 frames of each video. A threshold for the latter is set at 500 points and the parameter-settings with the highest average lifespan over this threshold are chosen for further testing. Initially, an educated guess for parameters of Lucas-Kanade (winSize = (15, 15), maxLevel = 10, criteria = (3, 10, 0.03)) and preprocessing (resize = (0.75, 0.75), filterType = median, filterSize = (21, 21)) were chosen. Thereafter, having compared settings on these simple measures, insight is given into more sophisticated measures explained in subsection 3.4.

4.1 Comparison by Average Lifespan and Quantity

In the following section, the measures of average lifespan and number of tracked points is analyzed. With these, comparison of the systems with different settings can be set in relation to each other, without subjectively comparing heatmaps or videos. The gathered data can be viewed in full in the appendix.

4.1.1 Key Point Detection Algorithms

When comparing the average lifespan of various settings, the most remarkable result is the difference between the videos taken by cameras with a frame-rate of 120 fps and those with 200 fps. All key point detectors show a notable higher lifespan and a lower number of tracked points for this increase in frame rate and a decrease in resolution. This becomes particularly apparent by comparing the key point parameter settings with the highest average lifespan, as can be seen in Figure 11. The Shi-Tomasi detector shows an average lifespan of 307.09 frames for its 740.45 tracked points in videos with 120 fps, while the 573.50 tracked points in videos with 200 fps yield an average lifespan of 2105.74 frames. The results of the fast Hessian detector are similar: 874.25 points tracked with an average lifespan of 166.50 frames for 120 fps videos and 1166.50 frames for 715.50 tracked points in videos with 200 fps. Comparatively, but in a more severe relation are the values found for the DoG detector: 2941 tracked points with an average lifespan of 87.69 frames were the results of the tests with 120 fps, while it shows 545 tracked points with an average lifespan of 1226.61 frames for videos with 200 fps.
Furthermore, it is visible that the Shi-Tomasi corner detector shows the highest average lifespan of the three detectors, for both classes of videos. The second highest yields the fast Hessian detector and the DoG method comes third. The average number of tracked points, on the other hand, shows a vast maximum for the DoG method in videos with 120 fps, it is more than double the amounts of the Shi-Tomasi and fast Hessian detectors. The latter had the most tracked points on average for the 200 fps videos, while Shi-Tomasi is in the middle, with DoG being the lowest in this measure.

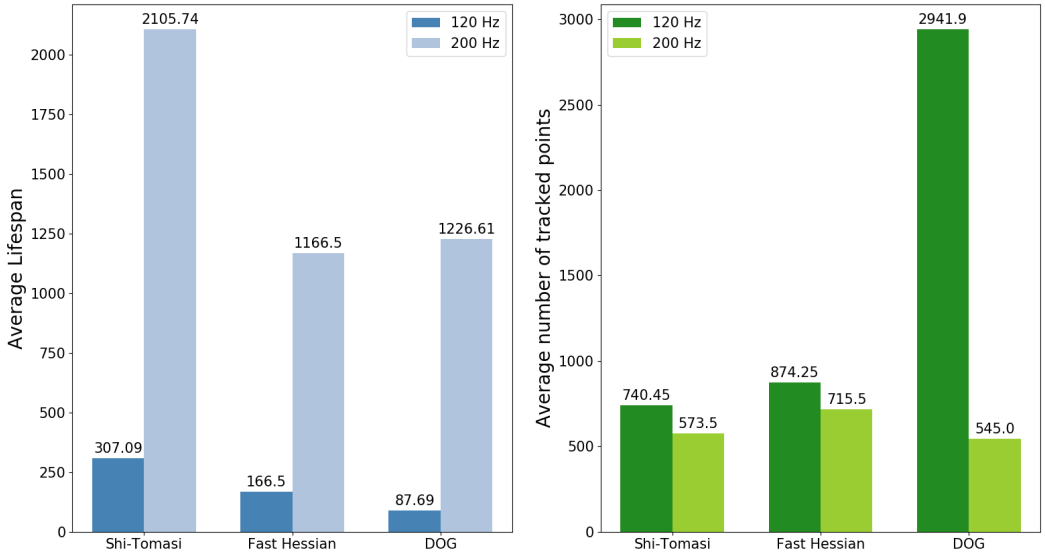


Figure 11: Values of best settings for each key point detector, averaged over all videos in the respective frame-rate

When looking at different values for parameter-settings of each algorithm, visible in Table 1, differences between videos with 120 and 200 fps persist. Concerning the Shi-Tomasi detector, medium quality level yields high values for average lifespan as does a large distance between corner points for those recorded with 120 fps. Also, large neighborhoods to calculate the structure tensor prove to be more successful than smaller ones. For videos recorded at 200 fps, a high quality level, medium to large minimal distance between key points and a small size for the structure tensor turn out to be more successful. Furthermore, the restriction by a maximum of detected corners does not seem to change a lot in average lifespan in both classes of videos, but does so in the number of tracked points. Also the quality level is of importance here - the setting with the lowest tested quality level and the highest maximum of returned corners yields an average of 168206.45 and 15893 key points per video, accompanied with the lowest average lifespans of this test, 65.02 and 408.14 frames, for videos with 120 fps and 200 fps, respectively.
For the fast Hessian detector, low numbers for octaves, octave-layers and the Hessian threshold yield a high lifespan for videos with 120 fps. Quite the opposite is the case for videos with a higher frame-rate: high numbers of octave layers and a medium threshold prove to be superior. For settings with various numbers of octaves, the same results concerning average lifespan and number of tracked points are visible, as is the case for higher numbers of octaves in videos with 120 fps. The highest quantity of key points is found with a setting of highest numbers of octaves and octave layers and the lowest tested value for the Hessian threshold, yielding an average of 88509.65 for videos with 120 fps and 2950.0 for those with 200 fps. These settings both yield the lowest average lifespan for this detector, 43.33 and 719.87, respectively.
The DoG detector shows the highest average lifespan in 120 fps videos with a setting of a high value for sigma and low values for number of features, octave layers and

both thresholds. What is visible overall for this class of videos, is that a low threshold for edges and a high one for contrast perform best in terms of average lifespan. The settings with the highest value for the contrast threshold yields not enough key points to prevail the thresholding of average number of tracked points in videos with 120 or 200 fps. In videos with a higher frame-rate, a low value for sigma is superior. A low number of features seem to improve the average lifespan, but lessen the number of key points, which can be observed for both classes of videos. Furthermore, a setting with many octave layers yields better results in both measures, average lifespan and number of tracked points, over both classes of videos. The highest number of tracked points is returned, when setting sigma and contrast threshold to low values, with high values for the returned number of features and edge threshold. The best settings for the two classes of videos differ only in the setting for the number of octave layers, which is the lowest for 200 fps videos and highest for those with 120 fps. The general maximum of tracked points on average, 24081.25 for 120 fps and 1615.5 for 200 fps, is reached by setting the contrast threshold and sigma very low, additional to a high edge threshold and number of features. Also the number of octave layers is high for 120 hz, but low for 200 hz videos. These setting reaches a moderate value for average lifespan, 54.83 for the videos with 120 fps, compared to the other parameter settings of DoG.

Framerate	120 fps		
Detector	Shi-Tomasi	Fast Hessian	DoG

Parameters	qualityLevel = 0.5;
	minDistance = 61;
	maxCorners = 510,
	blockSize = 5
		nOctaves = 1;
		nOctaveLayers = 1;
		hessianThreshold = 50
			sigma = 2.5;
			nfeatures = 10;
			nOctaveLayers = 1;
			edgeThreshold = 2;
			contrastThreshold = 0.01

Framerate	200 fps		
Detector	Shi-Tomasi	Fast Hessian	DoG

Parameters	qualityLevel = 0.5;
	minDistance = 1;
	maxCorners = 260,
	blockSize = 2
		nOctaves = 1;
		nOctaveLayers = 5;
		hessianThreshold = 100
			sigma = 1.0;
			nfeatures = 10;
			nOctaveLayers = 5;
			edgeThreshold = 10;
			contrastThreshold = 0.01

Table 1: Best parameters for both classes of videos and the three tested key point detection algorithms. Values for average lifespan and quantity can be viewed in Figure 11.

4.1.2 Parameters for the Lucas-Kanade Algorithm

For the evaluation of parameters for the Lucas-Kanade optical flow algorithm, the parameter-settings for each key point detector with the highest average lifespan were chosen in respect to the class of videos, coming to a total of 6 settings, as shown in Table 1. The testing is done over the first 10000 frames of all videos, and a threshold of 500 tracked points on average was set.

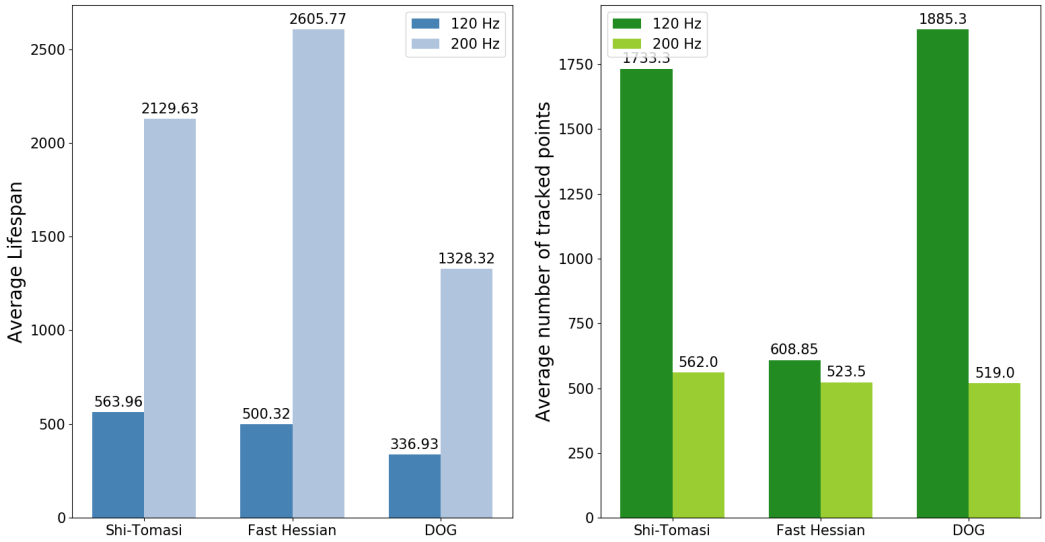


Figure 12: Values of the best settings for each key point detector and the Lucas-Kanade algorithm, averaged over all videos in the respective frame-rate.

As visible in Figure 12, the differences for the two classes of videos persist. For those with a frame-rate of 120 fps, the best setting of parameters for Lucas-Kanade optical flow and the Shi-Tomasi detector yields an average lifespan of 563.96 frames, while the result for 200 fps videos is nearly four times as much, 2129.63. The number of tracked points, on the other hand, is much lower for videos with the higher frame-rate than for the lower one, 562 and 1733.3, respectively. The behavior for the best setting of Lucas-Kanade parameters in respect to the best settings for the DoG method is similar. For 120 fps videos, the resulting values of average lifespan are 336.93 and 1328.32 frames for those taken with 200 fps. Also the number of tracked points are on average per video 1885.3 and 519.0 for lower and higher frame-rate, respectively. While the fast Hessian detector shows similar results for the average lifespan as the other two detectors, 500.32 (120 fps) and 2605.77 (200 fps) frames, the number of tracked points shows a difference: It returns only 608.85 on average for videos with 120 fps, which is only less than 20 percent more than for videos with a higher frame-rate (523.5).

The best parameter settings for Lucas-Kanade optical flow are equal in all key point detectors for videos with a frame-rate of 120 and can be seen in Table 2. A large window size, a high value for the epsilon and a low one for the maximal number of iterations prove to be the best setting for Lucas-Kanade concerning average lifespans over all 120 fps videos. It yields an average lifespan of 563.96 for the Shi-Tomasi, 500.32 for the fast Hessian and 336.93 frames for the DoG detector. Their respective number of tracked points are 1733.3, 608.84 and 1885.3.

The setting of differing values for the maximum number of levels of reduction (maxLevel) yields only a difference between low and medium/high numbers. The outcome is very similar or equal for a setting of 3 and 10 for this value, which also holds true concerning the videos with a higher frame-rate.

Framerate	120 fps		
Detector	Shi-Tomasi	Fast Hessian	DoG
Lucas-Kanade Parameters	winSize = (31,31); maxLevel = 3; criteria = (3, 4, 0.5)		

Framerate	200 fps		
Detector	Shi-Tomasi	Fast Hessian	DoG
Lucas-Kanade Parameters	winSize = (15,15); maxLevel = 3; criteria = (3, 30, 0.03)	winSize = (31,31); maxLevel = 3; criteria = (3, 30, 0.03)	winSize = (15,15); maxLevel = 0; criteria = (3, 4, 0.5)

Table 2: The best Lucas-Kanade parameters for the best parameter-setting of each of the tested key point detectors are shown. Note that the parameters for 120 fps videos are the same for all three.

However, the parameter-settings for the Lucas-Kanade algorithm differ more for the highest outcome of average lifespan per detector over the videos with a frame-rate of 200 fps. The returned value of 2605.77 is the highest for the fast Hessian detector, with a medium window size, a medium/high number of maximum level of reduction and a low value for epsilon. The only difference in the setting for the Shi-Tomasi detector is the window size, which yields a higher lifespan of 2129.63 frames with a medium value compared to lower or higher settings of this parameter. The same, medium value proves to be superior for the DoG method for key point detection, in addition to a high value for epsilon, the lowest tested setting for the number of iterations and the maximum level of reduction. The average number of tracked points are very similar in this class of videos, 562.0, 523.5 and 519.0 for the setups with the Shi-Tomasi, fast Hessian and DoG detectors, respectively.
In general, it can be said that a large window size yields higher average lifespans, as does multiple levels of reduction, excluding the results of the DoG method for 200 fps videos. The termination criteria show a lower value of average lifespan for a high number of maximum iterations and a low value for epsilon and a high one vice versa. A medium average lifespan is accounted for medium numbers of termination criteria. The number of tracked points is highest for the DoG (120 fps) and the fast Hessian (200 fps) detectors. This can be observed when the parameters are set to small window size, large levels of reduction, a medium value for the termination criteria of iteration and a low one for epsilon. It is accompanied by the lowest average lifespan for 120 hz videos, 20.99 frames, and a very low one for those with 200 fps, 173.88 frames. It declines for a larger window size and higher values for both termination criteria. This holds true for both classes of videos and all detectors.

4.1.3 Parameters for Preprocessing

As a last step of optimization the preprocessing steps are evaluated on the three key point detectors and their best respective parameter-settings for Lucas-Kanade, depending on the frame-rate. The setting with the second highest returned value of average lifespan in videos with 120 fps is less than 1% worse in this measure, but yields an increase of about 50% in the number of tracked points, compared to the

best. For this reason it is chosen as the best setting, and not, as beforehand, the one with strictly the highest lifespan. The other settings were chosen, as before, by choosing the one with the highest average lifespan over the threshold of 500 tracked points on average per video.

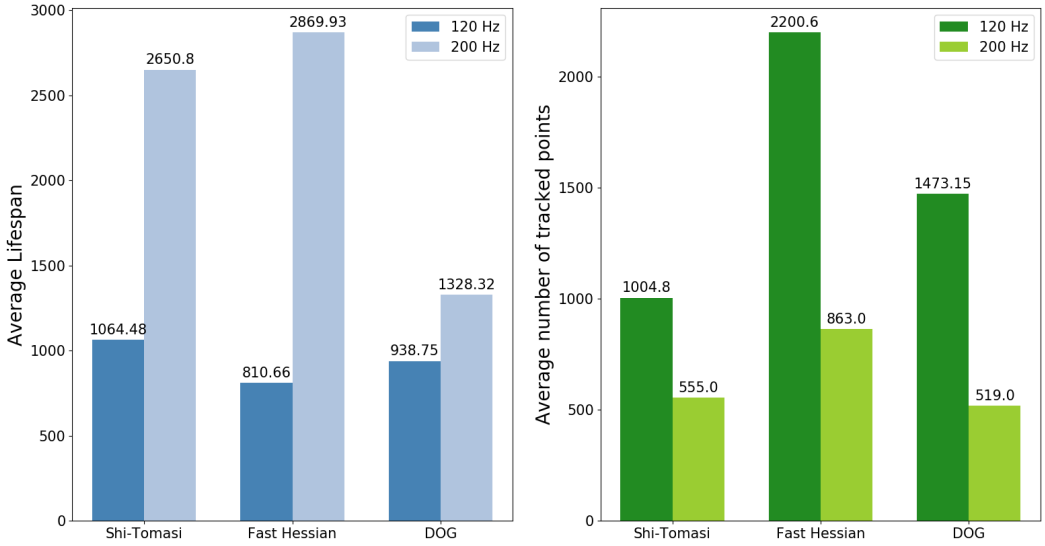


Figure 13: Values of the best settings for each key point detector, Lucas-Kanade algorithm and preprocessing, averaged over all videos in the respective framerate

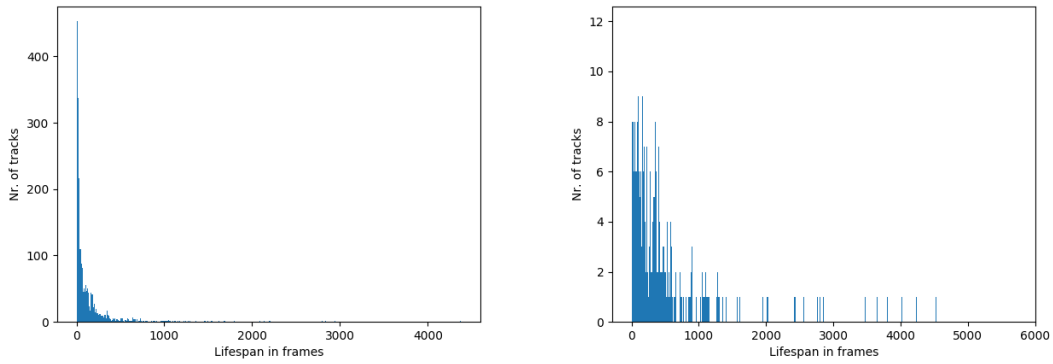
What is visible in Figure 13 is similar to the preceding steps of optimization. A large difference can be seen between the values for average lifespan by comparing the best settings of the system for the two classes of videos. The setting returning the highest value for it in 120 hz videos is the setting with the Shi-Tomasi corner detector (1064.48 frames), followed by the DoG method (938.75 frames) and the fast Hessian (810.66 frames). The returned values for fast Hessian and Shi-Tomasi detectors are nearly tripled for videos with a higher frame-rate. The maximum in this class yields the fast Hessian detector with 2869.93, the Shi-Tomasi detector comes second with 2650.8 and the DoG approach yields the lowest result of 1328.32. The difference is also high for the average number of tracked points. The fast Hessian yields 2200.60 for videos with 120 fps and only 863.00 for those with 200 fps. The setup with the DoG detector returns a value of 1473.15 average key points per video for 120 fps and 519.00 for the ones with 200 fps. The Shi-Tomasi corner detector returns the lowest average number of points for videos with a frame-rate of 120 fps, namely 1004.8, and 555 for videos with 200 fps.
What can be seen in general for 120 fps videos, is that a resize to the lowest tested resolution $(0.5 * width, 0.5 * height)$ returns the highest values of average lifespan. The other preprocessing steps differ more, as can be seen in Table 3. For the Shi-Tomasi corner detector, reducing both width and height to a half and filtering with a median is in both classes of videos superior, though the latter is differing in size, large for 120 and small for 200 fps videos. The fast Hessian returns best results with a large Gaussian kernel and halving width and height for 120 fps videos.

Framerate	120 fps		
Detector	Shi-Tomasi	Fast Hessian	DoG
Lucas-Kanade Parameters	winSize = (31,31); maxLevel = 3; criteria = (3, 4, 0.5)		
Preprocessing	resize = (0.5, 0.5); filter-type = median; filter-size = (15, 15) / (21, 21)	resize = (0.5, 0.5); filter-type = Gauss; filter-size = (21, 21)	resize = (0.5, 0.5) filter-type = None

Framerate	200 fps		
Detector	Shi-Tomasi	Fast Hessian	DoG
Lucas-Kanade Parameters	winSize = (15,15); maxLevel = 3; criteria = (3, 30, 0.03)	winSize = (31,31); maxLevel = 3; criteria = (3, 30, 0.03)	winSize = (15,15); maxLevel = 0; criteria = (3, 4, 0.5)
Preprocessing	resize = (0.5, 0.5); filter-type = median; filter-size = (5, 5)	resize = (1, 1); filter-type = None;	resize = (0.75, 0.75); filter-type = median; filter-size = (21, 21)

Table 3: The best pre-processing steps found to optimize average lifespan and number of tracked points with regard to the initially found parameters for the three tested key point detecion algorithms and Lucas-Kanade. Note the exception to choose a different parameter marked with a "/"

Remarkably, for videos with a higher frame-rate, no preprocessing yields best results. No filtering is best for the DoG detector concerning 120 fps videos, but applying a large median-kernel and a resize to three-quarters of width and height are necessary for the highest average lifespan. These correspond to the values initially chosen for parameter testing in the initial steps.
Preprocessing with no resizing and no or little filtering yield highest numbers of tracked points (max. 12727.4 for fast Hessian in 120 fps videos) and are accompanied by low average lifespans (< 800). The difference is visible in Figure 14, showing the setup with the highest returned number of tracked points, a), and the one, with the highest average lifespan, b). A vast majority of points cannot be tracked longer than 500 frames in a), while this portion is relatively small in b). Still, only a minority of points can be tracked longer than 1000 frames.



a) b)

Figure 14: Histograms for the same video recorded at a frame-rate of 120 fps. a) shows the setup with the lowest returned average lifespan in preprocessing, using the DoG detector, no resizing and a median filter of size (21, 21). b) shows the best rated setup of paramters in this last step, using the Shi-Tomasi detector, a resize of (0.5, 0.5) and also a median filter of size (21, 21).

4.2 Comparison by Spatial Distribution

As described in section 3.4.2, the spatial distribution of tracked points are visualized using heatmaps, shown in Figure 15 and Figure 16, for videos with 120 and 200 fps, respectively. Used for this are the best setups after optimization with parameter-settings for key point detection, the Lukas-Kanade method and preprocessing. First of all, most key point locations are situated on or directly around semantically relevant regions, for example eye-lids or the iris. Furthermore, the measure of quantity becomes apparent: The general brightness is higher in the heatmaps with a higher number of tracked points, namely the fast Hessian and the DoG detector. In most cases, contours of the filmed eye are visible, but also random points, located in semantically irrelevant parts. This is visible in the first three heatmaps, at the lower middle parts, as well as in the upper and lower right corner of the Shi-Tomasi detector. Also, a lot of seemingly random parts of the image are tracked for short amount of times, as can be seen in Figure 15 h), where bright pixels are distributed over parts which appear dark in f) and g). Also semantically irrelevant points, tracked over long periods of time, are visible, for example, the heatmap in Figure 16 b) in the upper right corner shows seemingly one tracked point, which wanders off in random directions. Some lack of brightness holds information, too: In Figure 15, the bottom right canthus is visible in the heatmap, but in the sclera beside it, it is dark.

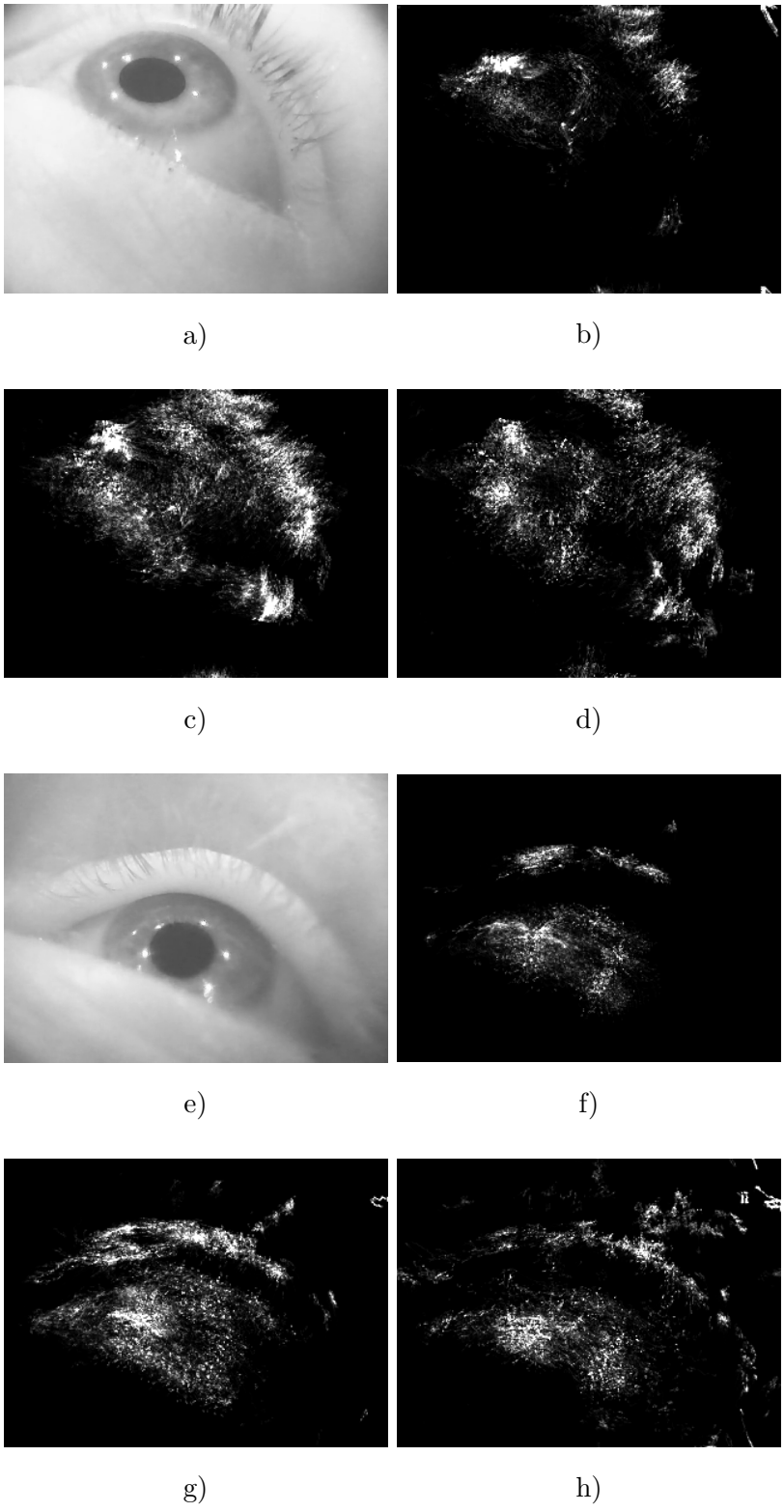


Figure 15: Spatial distribution of key points, visualized in form of heatmaps for two videos with 120 fps. The brighter a pixel, the more often a tracked point was on this location. b) and f) show the distribution of the optimized Shi-Tomasi, c) and g) the fast Hessian, d) and h) the DoG tracker. a) and e) show a snapshot of the video, from which the heatmaps are shown: a) corresponds to b), c) and d); e) corresponds to f), g) and h))

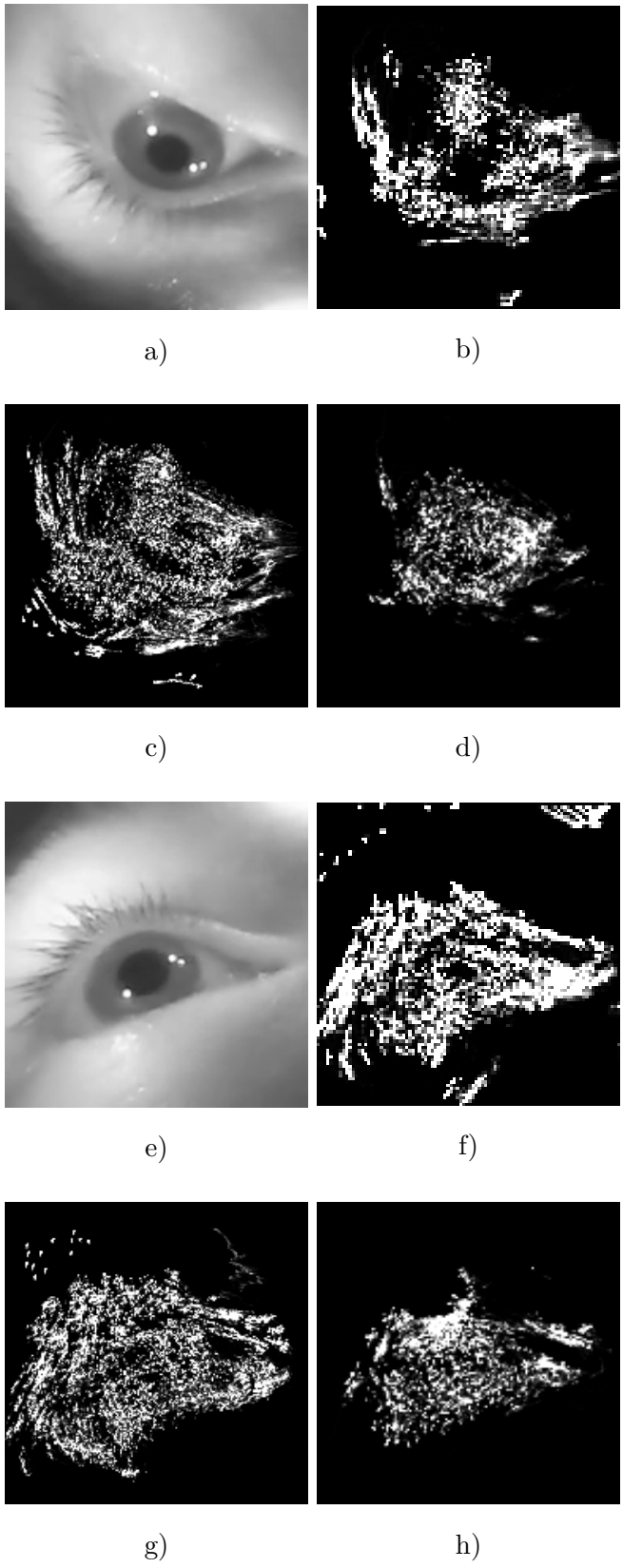


Figure 16: Spatial distribution of key points, visualized in form of heatmaps for both videos with a framerate of 200 fps. The brighter a pixel, the more often tracked point was on this location. b) and f) show the distribution of the optimized Shi-Tomasi, c) and g) the fast Hessian, d) and h) the DoG tracker. a) and e) show a snapshot of the video, from which the heatmaps are shown (a) correspnds to b), c) and d); e corresponds to f), g) and h))

4.3 Validation Measurse: Blink Detection

To obtain information, whether the optimized optical flow system is viable for classification of eye-movement, one video of the dataset with annotated blinks was chosen. It is similar to the videos with 120 fps used for optimization (see subsubsection 3.4.1) and yields the same frame-rate. The first 10000 frames are tested and summarized in a heat histogram of directions (HoD). An extract for the frames between 5800 and 6050 can be seen in Figure 17. A lot of motion is visible. Firstly, considerable findings are the small, repetitive motions visible throughout the HoD, which can also be seen in the snippet. In the tested frames, only one of the 5 true blinks was undetected, which was in the first few hundred frames. At the other time-steps, the HoD shows differentiable magnitudes (black rectangle in the figure). Furthermore, additional to the 4 correctly identified blinks, 5 locations in the HoD could be interpreted as blinks as well (false positives), one of which is also visible in the extract, marked with a red rectangle.

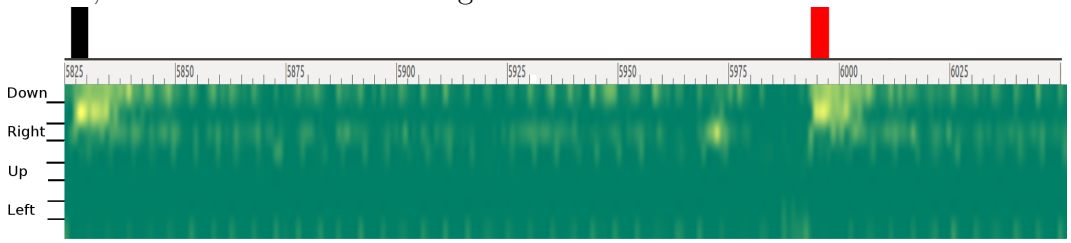


Figure 17: Extract of heat histogram of directions. For each frame, direction and magnitude of tracked points between these frames are visible. Black and red bars show manually annotated true and false detections of blinks, respectively.

4.4 Removal of Global Flicker

Because of the fact that a special kind of noise, flickering, is visible in some of the videos, prototypical testing was done for global flicker removal for one of each of the videos with 120 and 200 fps and their respective optimal settings for movement analysis. The results can be viewed in Table 4. It is visible that midway equalization does change the outcome of the measures average lifespan and quantity, but there is not a large difference to the outcome for the videos which were not treated with histogram equalization. For the video with a frame-rate of 120 fps, the number of tracked points decreases by less than 10%, while the average lifespan increases by approx. 4%. In contrast to that are the results of the video with 200 fps. The average lifespan decreases by less than 10%, while the number of tracked points increases by approximately 10%.

Frame-rate	120 fps		200 fps	
Detector	Shi-Tomasi		Fast Hessian	
Midway Equalization	No	Yes	No	Yes
Average lifespan	2508.52	2588.67	5245.85	4814.42
Number of tracked points	535	498	403	438

Table 4: Results for 10.000 frames for the optimized settings for one 120 and 200 fps video, each with and without global flicker removal via midway equalization.

5 Discussion

In order to examine the applicability of systems with sparse Lucas-Kanade optical flow on the analysis of eye-movement in mobile eye-tracking, three key point detection algorithms were chosen, tested and optimized in addition to the previous processing of the data. This was done in four steps:

- Optimization of parameters for the key point detection algorithms.
- Optimization of parameters for the Lucas-Kanade algorithm.
- Optimization of preprocessing steps.
- Validation on one video with annotated time-points for blinks.

Furthermore, an algorithm to remove global flicker was applied to two videos and the results were put in direct comparison to those videos without the removal. All of these results are presented in section 4. In this section, they are discussed, beginning with the general differences of videos with higher and lower frame-rate. It is followed by detailed deductions based on the testing and optimization. Lastly, the validation measure is examined and final conclusions are drawn.
When viewing the results of the first step of optimization, large differences became apparent between 120 fps and 200 fps videos. Thus, the decision was made to preform the testing steps for both classes separately. The differences are due to various reasons. Firstly, since only the first 10000 frames of the videos were taken, the real time captured is approximately 83.33 seconds in videos with 120 fps and only 50 seconds for those with 200 fps. Accordingly, the speed of motion, relative to frames, is lower with a higher frame-rate. Since it is known, that problems arise for the Lucas-Kanade method to capture optical flow of fast motions, modifications have been found in the past and are applied in this thesis, as can be seen in subsection 2.2. Nevertheless, differences persist. Secondly, not only the frame-rate of the videos is different, but also the resolution, which is much smaller for videos with a higher frame-rate. Therefore, motions in the real world are larger in terms of pixel per frame when captured with a larger resolution and pose similar problems for the Lucas-Kanade optical flow as a difference in frame-rate. Furthermore, the space to detect and place key points is limited, not only by the amount of pixels, but also by the Shi-Tomasi detector and the Lucas-Kanade algorithm, both can be set with an explicit minimum distance between points being tracked, as explained in subsection 3.5. Moreover, a smaller resolution is similar to resizing the images, which leads to more information being contained per pixel and a higher contrast. All three key point detectors behave differently if resizing is applied, which indicates that the differences in resolution of both classes of videos influences the outcomes of the evaluation. Thirdly, the datasets are unequal in the amount of videos they contain. Only one subject did the test in VR with the cameras capturing 200 fps. It is much less than the other dataset comprising 10 subjects and therefore a much higher variance is probable. In particular, only one blink is captured in the first 10000 frames of the recordings with 200 fps. Since blinking is the most disruptive movement for this setup, this is a crucial difference to the videos in the other dataset, where blinks happen much more often.

Concerning the interpretation of parameters, the first focus lies on the Shi-Tomasi corner detector. The quality level is a parameter to reject found key points and can be directly linked to the robustness of the returned points. Robust points yield a higher lifespan, but too many rejections result in a smaller portion of the image being tracked and since both of these account for the ratings of a parameter setting, a medium value is superior. The minimal distance between found key points is important for the distribution over the image. Since it is calculated in units of pixels, a high value is best for processing of images with a larger resolution and a low one for small ones. Nevertheless, since key points are detected every 10 frames, the change of this parameter is not of much consequence, similar to the maximum of returned corners. This parameter rejects most of the found key point candidates and leaves only a certain amount. The circumstances of consecutive detection and similarity to the quality level lead to very few consequences in change of this parameter. Since corners are directly linked to a strong contrast, the size of the structure tensor (blockSize) must be larger for a higher resolution, because less information and possibly noise are contained in each pixel. Accordingly, it must be small for a lower resolution, also because large neighborhoods can possibly include another corner, which is undesirable. The Shi-Tomasi key point detector yields overall the best results for videos captured at 120 Hz. The found key points can be tracked better by the Lucas-Kanade algorithm than the other detectors. This could be due to the fact, that it is the only one which is getting information of previously tracked points as input in form of a mask.
To explain the optimal parameter settings for the fast Hessian detector, one has to note that the information of scale of the returned key points is not conveyed to the Lucas-Kanade algorithm, but is only contained when building the SURF-descriptor. Additionally, the key points on various scales are obtained by upscaling the box filter and therefore differ crucially from the down-sampling method used by the Lucas-Kanade optical flow to estimate locations on different scales. For these reasons, key points found by the fast Hessian detector on various scales can only be interpreted on the original image, thus resulting only in robust key points for tracking by chance. Accordingly, the optimal setting for this key point detection algorithm uses only the basic scale and a medium Hessian threshold for rejection in videos with a low frame-rate. For the videos tested with the higher frame-rate on the other hand, a lot of key points can be tracked for a long time, even if they are not as robust, since only one blink occurs in the whole tested frames and the average lifespan over all tracked points is measured. Therefore, key points on a lot of scales with a medium threshold, to account for a bit of robustness, are superior. Generally, the fast Hessian detector in addition to Lucas-Kanade optical flow yields good results for motion analysis.
The DoG detector suffers from the same problems as the fast Hessian, even though the approach to yield different scales of the original image is more similar to the one of Lucas-Kanade. Still, the DoG detector yields the lowest results in the first step of optimization. In the steps thereafter, it scores better, which implies a suboptimal setting of educated guesses for parameters of preprocessing and the Lucas-Kanade algorithm. The found key point candidates are thresholded threefold: by contrast, edges and an absolute maximum number of points. It can be inferred, that these thresholds do not interact well with each other, because in most settings of parameters, a lot of key points are returned, but the lifespan is very low. For other preprocessing

steps which were tested, this detector yields good results concerning the ability of the Lucas-Kanade algorithm to track the found key points.
In the following, the second step of optimization is discussed. It is concerned with the best Lucas-Kanade parameters for each detector, which yields the same for all three of them for videos recorded with 120 Hz cameras: a medium window size to estimate the new locations, three or more pyramidal levels and a low maximum of iteration and a high value for epsilon, the two latter specifying termination criteria. These criteria reduce computational time when set to higher values, since not all key points can be tracked. The pyramidal levels are important for image sequences with a high resolution and a low frame-rate, which counts to the ability to track fast and large motion. For the reasons mentioned above, it is not as important for the videos with 200 fps. The window size correlates to the neighborhood, in which the optical flow is calculated. It is at a medium level, because most of the information can be inferred from these pixels, without taking into account other, contradicting motion in other regions of the video.
Lastly, different preprocessing steps were tested with regards to the optimal settings for key point detection algorithms and Lucas-Kanade parameters. The reduction to a lower resolution proves to be most important for videos with a higher frame-rate and for the Shi-Tomasi detector. The other detectors implement reduction and are filtering themselves, so it yields less importance for them. Similarly, the Shi-Tomasi detector yields best result with large-sized filters to improve contrast and edges.
The direct comparison of the two videos treated with midway equalization to remove flicker and those without it, shows only a small and mixed difference in comparison to not de-flickered videos. While the average lifespan improved slightly in one of the videos after removal, the number of tracked points declined. For the other video, the number of tracked points increased, while the lifespan was reduced. It is difficult, to draw final conclusions from this, but it seems, that the flicker contained in the videos is not of global nature (concerning the whole image), but only resides in a large, local part of it. Therefore, the global method of midway equalization does not suffice to remove it and other methods, which can be applied locally have to be found.
Finally, the validation measure to manually classify blinks based on the pattern of movement presented as a HoD yields mixed results, too. While most of the blinks could be detected correctly, one was not clearly visible. This is based on an insufficient number of initialized key points for the relevant time frame. Since the HoD is equalized over all measured frames, the average magnitude at this point was not as big as in later stages, when more key points were tracked. Also, even though optimized preprocessing has been applied to the video, consistent, repetitive motion, probably due to noise, is visible throughout the histogram of directions. Unfortunately, the number of false detected blinks is higher than the one of correct detection which could be linked to the fact that even though the algorithm to build the hod was tested in all directions, no motion is visible to the left and up direction. Furthermore, strong motion in the videos like saccades can be accounted for the false detections.
In conclusion, a system to extract motion information from mobile eye-tracking data was optimized, using algorithms for preprocessing of the data and three methods for key point detection in addition to the sparse version of Lucas-Kanade optical flow. The resulting system yields a long lifespan and a good distribution over

the semantically relevant parts of the data. The validation measure showed the applicability to the task and it is possible to obtain information about eye-movement in this system. Nevertheless, due to a large amount of false positives, the classification of blinks does not yield positive results. Furthermore, the lifespans of the key points have improved to more than 1000 on average and the key points are distributed spatially over semantically important regions of the eyes. Accordingly, a major problem, the vanishing of key points located at the eye-lids at blinks, was overcome and their detection seems possible by further refinement of the algorithm.

Future research

To improve the developed system, some steps can be taken, which were not possible to implement in this thesis. First of all, the optimization can be improved by testing all parameter-settings in all variations at once, including the minimum Eigen-threshold of the Lucas-Kanade algorithm. Guesses had to be taken for preprocessing and the optical flow method, which leads to a bias for the optimization. Also, a practice to convey the information of previously found and tracked key points not only to the Shi-Tomasi detector, but also to the others would be interesting. Additionally, other values for the threshold of tracked points are important to take into account, since the threshold chosen in this thesis proved to be low in comparison to the values actually found. Another improvement could be to measure the lifespan not in frames, but in real-time, to be able to compare the system on videos with differing frame-rates. Accordingly, the resolution in different classes of videos can be equalized. Furthermore, not only more videos with a higher frame-rate should be taken into account, but also other datasets to increase the variance of the data and make the results of the optimization of the parameters more robust. Moreover, other key point detection algorithms can be tested and a local method has to be implemented to remove flicker in the videos or it should be completely avoided by utilizing a different hardware-setting. Finally, the heat-histogram of directions can be improved by taking the average of magnitudes of key points over certain smaller time-frames, than the whole measure. This could improve the findings in the beginning, when not so many key points have been initialized. The results indicate that with a proper optimization an automatic classification of eye-movements could be possible.

6 Appendix

6.1 Detector Testing

Shi-Tomasi

Rank	Average lifespan	Number of keypoints	Shi-Tomasi parameters			
			quality-level	min-Distance	max-Corners	block-Size
1	307.09	740.45	0.5	61	510	5
2	307.09	740.45	0.5	61	260	5
3	307.07	738.25	0.5	61	10	5
4	305.23	782.75	0.5	31	10	5
5	304.97	789.60	0.5	31	510	5
6	304.97	789.60	0.5	31	260	5
7	297.59	877.50	0.5	1	10	5
8	297.24	915.65	0.5	1	260	5
9	297.24	915.65	0.5	1	510	5
10	281.62	948.70	0.5	61	10	3
11	281.26	959.45	0.5	61	510	3
12	281.26	959.45	0.5	61	260	3
13	278.92	1009.20	0.5	31	10	3
14	278.10	1059.00	0.5	31	260	3
15	278.10	1059.00	0.5	31	510	3
16	274.22	618.20	0.5	61	510	2
17	273.11	644.20	0.5	31	510	2
18	270.64	1114.35	0.5	1	10	3
19	270.47	611.50	0.9	61	260	2
20	270.47	611.50	0.9	61	10	2
21	269.52	636.90	0.9	31	10	2
22	269.52	636.90	0.9	31	260	2
23	268.85	1269.75	0.5	1	260	3
24	268.85	1269.75	0.5	1	510	3
25	262.47	689.00	0.9	1	10	2
26	262.47	689.00	0.9	1	260	2
27	262.47	689.00	0.9	1	510	2
28	251.99	2095.90	0.1	61	10	5
29	249.95	2563.70	0.1	61	510	5
30	249.95	2563.70	0.1	61	260	5
31	240.96	2475.35	0.1	31	10	5
32	236.70	4169.75	0.1	31	260	5
33	236.70	4169.75	0.1	31	510	5
34	235.29	2810.05	0.1	61	10	3
35	232.84	3460.15	0.1	61	260	3
36	232.84	3460.15	0.1	61	510	3
37	227.05	2889.90	0.1	61	510	2
38	226.51	1914.90	0.5	61	10	2
39	226.32	1933.95	0.5	61	260	2
40	219.21	2371.10	0.5	31	10	2
41	218.74	3494.85	0.1	31	10	3
42	218.50	2516.20	0.5	31	260	2
43	218.29	4526.90	0.1	31	510	2
44	213.90	3165.95	0.1	1	10	5
45	213.81	6119.45	0.1	31	510	3
46	213.81	6119.45	0.1	31	260	3

Rank	Average lifespan	Number of keypoints	Shi-Tomasi parameters			
			quality-level	min-Distance	max-Corners	block-Size
47	205.49	8856.30	0.1	1	260	5
48	205.35	10008.25	0.1	1	510	5
49	201.64	3154.70	0.5	1	10	2
50	198.20	3997.90	0.5	1	510	2
51	198.20	3997.90	0.5	1	260	2
52	197.55	7042.50	0.9	31	10	5
53	192.69	4579.70	0.1	1	10	3
54	188.96	7749.55	0.9	31	10	3
55	182.42	7710.70	0.9	1	10	5
56	178.90	8190.65	0.9	1	10	3
57	178.82	9756.00	0.9	31	260	5
58	178.82	9756.00	0.9	31	510	5
59	177.59	12671.00	0.1	1	260	3
60	177.15	16117.10	0.1	1	510	3
61	165.01	8015.55	0.9	61	510	3
62	161.85	7994.15	0.1	61	10	2
63	159.52	14322.85	0.9	31	510	3
64	159.52	14322.85	0.9	31	260	3
65	157.21	8326.55	0.1	31	10	2
66	152.19	8266.90	0.9	61	510	2
67	143.99	8473.60	0.1	1	10	2
68	141.67	19763.85	0.9	1	260	5
69	141.67	19763.85	0.9	1	510	5
70	140.40	16651.30	0.1	61	260	2
71	135.98	15657.25	0.9	31	510	2
72	126.65	33258.05	0.9	1	260	3
73	126.65	33258.05	0.9	1	510	3
74	115.06	39062.65	0.1	31	510	2
75	115.06	39062.65	0.1	31	260	2
76	90.37	58635.35	0.9	61	10	5
77	89.28	63775.15	0.9	61	260	5
78	76.65	111770.85	0.9	61	10	3
79	73.15	145249.95	0.9	61	260	3
80	71.01	119068.70	0.1	1	260	2
81	65.02	168206.45	0.1	1	510	2

Table 5: The results for the testing of the Shi-Tomasi parameters on 120 Hz videos. For the Lucas-Kanade parameters {'winSize': (15, 15), 'maxLevel': 10, 'criteria': (3, 10, 0.03)} were used. For the preprocessing resize with (3/4, 3/4) und a median filter with (21,21) was used.

Rank	Average lifespan	Number of keypoints	Shi-Tomasi parameters			
			quality-level	min-Distance	max-Corners	block-Size
1	2105.74	573.50	0.5	1	260	2
2	2105.74	573.50	0.5	1	510	2
3	2012.67	554.00	0.5	31	510	2
4	2012.67	554.00	0.5	31	260	2
5	2012.67	554.00	0.5	31	10	2
6	2000.79	610.50	0.5	1	10	2
7	1825.79	558.50	0.5	31	510	5

Rank	Average lifespan	Number of keypoints	Shi-Tomasi parameters			
			quality-level	min-Distance	max-Corners	block-Size
8	1825.79	558.50	0.5	31	10	5
9	1825.79	558.50	0.5	31	260	5
10	1774.72	533.00	0.5	61	510	5
11	1774.72	533.00	0.5	61	260	5
12	1774.72	533.00	0.5	61	10	5
13	1760.69	890.00	0.5	1	510	5
14	1760.69	890.00	0.5	1	260	5
15	1752.69	886.00	0.5	1	10	5
16	1461.45	889.00	0.5	61	260	3
17	1461.45	889.00	0.5	61	10	3
18	1461.45	889.00	0.5	61	510	3
19	1403.96	1041.00	0.5	31	510	3
20	1403.96	1041.00	0.5	31	260	3
21	1403.96	1041.00	0.5	31	10	3
22	1395.62	1699.50	0.1	61	510	5
23	1395.62	1699.50	0.1	61	10	5
24	1395.62	1699.50	0.1	61	260	5
25	1339.03	1480.50	0.5	1	10	3
26	1336.56	1525.00	0.5	1	260	3
27	1336.56	1525.00	0.5	1	510	3
28	1307.02	2167.50	0.1	61	260	2
29	1307.02	2167.50	0.1	61	510	2
30	1307.02	2167.50	0.1	61	10	2
31	1288.76	2576.50	0.1	61	10	3
32	1288.76	2576.50	0.1	61	260	3
33	1288.76	2576.50	0.1	61	510	3
34	1207.52	2489.50	0.1	31	260	5
35	1207.52	2489.50	0.1	31	510	5
36	1206.94	2488.50	0.1	31	10	5
37	980.38	3638.00	0.1	31	260	2
38	980.38	3638.00	0.1	31	510	2
39	975.30	3624.50	0.1	31	10	2
40	941.21	4575.00	0.1	31	510	3
41	941.21	4575.00	0.1	31	260	3
42	933.05	4581.00	0.1	31	10	3
43	863.99	7672.00	0.1	1	510	5
44	863.99	7672.00	0.1	1	260	5
45	849.86	5099.00	0.1	1	10	5
46	587.76	6055.50	0.1	1	10	2
47	535.06	7612.00	0.1	1	10	3
48	515.22	9242.00	0.1	1	260	2
49	505.14	9682.50	0.1	1	510	2
50	408.14	15893.00	0.1	1	260	3
51	408.14	15893.00	0.1	1	510	3

Table 6: The results for the testing of the Shi-Tomasi parameters on 200 Hz videos. For the Lucas-Kanade parameters {'winSize': (15, 15), 'maxLevel': 10, 'criteria': (3, 10, 0.03)} were used. For the preprocessing resize with (3/4, 3/4) und a median filter with (21,21) was used.

Fast Hessian

Rank	Average lifespan	Number of keypoints	Fast Hessian parameters		
			nOctaves	nOctave-Layers	hessian-Threshold
1	166.50	874.25	1	1	50
2	126.02	1316.45	1	3	100
3	114.14	540.25	1	7	500
4	109.27	3215.70	1	3	50
5	108.96	1567.05	1	5	200
6	87.86	9695.25	1	1	10
7	84.77	4182.35	1	5	100
8	82.81	2771.50	1	7	200
9	79.43	818.05	3	1	500
10	70.58	2808.50	3	1	200
11	69.94	9082.20	1	5	50
12	66.41	20364.30	1	3	10
13	66.32	6933.95	1	7	100
14	65.67	1270.95	7	1	500
15	65.67	1270.95	5	1	500
16	65.63	5594.55	3	1	100
17	65.18	9434.80	3	1	50
18	64.26	3628.60	5	1	200
19	64.26	3628.60	7	1	200
20	62.44	10593.15	7	1	50
21	62.44	10593.15	5	1	50
22	61.95	6619.75	7	1	100
23	61.95	6619.75	5	1	100
24	59.28	13863.90	1	7	50
25	58.11	28587.35	3	1	10
26	58.07	2552.45	3	3	500
27	57.56	3786.80	3	5	500
28	57.38	29789.40	5	1	10
29	57.38	29789.40	7	1	10
30	56.80	3110.05	5	3	500
31	56.80	3110.05	7	3	500
32	56.66	4302.85	7	5	500
33	56.66	4302.85	5	5	500
34	55.52	7719.30	3	3	200
35	55.30	36950.05	1	5	10
36	54.84	4717.45	3	7	500
37	54.48	8613.80	5	3	200
38	54.48	8613.80	7	3	200
39	54.17	5197.70	5	7	500
40	54.17	5197.70	7	7	500
41	53.67	11097.25	3	5	200
42	53.05	11924.35	7	5	200
43	53.05	11924.35	5	5	200
44	52.13	46246.45	1	7	10
45	51.88	14283.35	3	3	100
46	51.27	15362.50	7	3	100
47	51.27	15362.50	5	3	100
48	51.11	13587.50	3	7	200
49	50.83	22736.75	3	3	50
50	50.68	14364.70	7	7	200

Rank	Average lifespan	Number of keypoints	Fast Hessian parameters		
			nOctaves	nOctave-Layers	hessian-Threshold
51	50.68	14364.70	5	7	200
52	50.38	23930.95	5	3	50
53	50.38	23930.95	7	3	50
54	49.60	20424.85	3	5	100
55	49.26	21419.35	7	5	100
56	49.26	21419.35	5	5	100
57	47.89	32445.25	3	5	50
58	47.70	56624.35	3	3	10
59	47.66	33547.40	7	5	50
60	47.66	33547.40	5	5	50
61	47.55	57848.25	7	3	10
62	47.55	57848.25	5	3	10
63	47.34	24636.00	3	7	100
64	47.10	25569.20	5	7	100
65	47.10	25569.20	7	7	100
66	45.87	38609.60	3	7	50
67	45.70	39647.40	7	7	50
68	45.70	39647.40	5	7	50
69	44.64	77078.30	3	5	10
70	44.57	78198.50	7	5	10
71	44.57	78198.50	5	5	10
72	43.39	87451.65	3	7	10
73	43.33	88509.65	7	7	10
74	43.33	88509.65	5	7	10

Table 7: The results for the testing of the fast Hessian parameters on 120 Hz videos. For the Lucas-Kanade parameters {'winSize': (15, 15), 'maxLevel': 10, 'criteria': (3, 10, 0.03)} were used. For the preprocessing resize with (3/4, 3/4) und a median filter with (21,21) was used.

Rank	Average lifespan	Number of keypoints	Fast Hessian parameters		
			nOctaves	nOctave-Layers	hessian-Threshold
1	1166.50	715.50	1	5	100
2	1151.17	735.00	7	3	100
3	1151.17	735.00	5	3	100
4	1151.17	735.00	3	3	100
5	1035.95	517.50	5	5	200
6	1035.95	517.50	3	5	200
7	1035.95	517.50	7	5	200
8	1022.01	798.50	1	7	100
9	998.14	934.50	5	5	100
10	998.14	934.50	7	5	100
11	998.14	934.50	3	5	100
12	970.61	825.50	1	3	50
13	947.37	599.50	3	7	200
14	947.37	599.50	5	7	200
15	947.37	599.50	7	7	200
16	930.48	1017.50	3	7	100
17	930.48	1017.50	5	7	100
18	930.48	1017.50	7	7	100

Rank	Average lifespan	Number of keypoints	Fast Hessian parameters		
			nOctaves	nOctave-Layers	hessian-Threshold
19	922.12	1132.50	1	5	50
20	921.95	628.00	7	1	50
21	921.95	628.00	3	1	50
22	921.95	628.00	5	1	50
23	905.04	1186.00	7	3	50
24	905.04	1186.00	5	3	50
25	905.04	1186.00	3	3	50
26	862.00	1424.50	7	5	50
27	862.00	1424.50	5	5	50
28	862.00	1424.50	3	5	50
29	855.28	1219.50	1	7	50
30	824.69	1507.50	5	7	50
31	824.69	1507.50	3	7	50
32	824.69	1507.50	7	7	50
33	771.90	2396.00	1	5	10
34	751.86	2113.00	1	3	10
35	746.31	2472.00	1	7	10
36	741.60	2632.50	3	3	10
37	741.60	2632.50	5	3	10
38	741.60	2632.50	7	3	10
39	732.91	2875.00	7	5	10
40	732.91	2875.00	3	5	10
41	732.91	2875.00	5	5	10
42	727.66	1685.50	5	1	10
43	727.66	1685.50	7	1	10
44	727.66	1685.50	3	1	10
45	721.75	1220.50	1	1	10
46	719.87	2950.00	3	7	10
47	719.87	2950.00	5	7	10
48	719.87	2950.00	7	7	10

Table 8: The results for the testing of the fast Hessian parameters on 200 Hz videos. For the Lucas-Kanade parameters {'winSize': (15, 15), 'maxLevel': 10, 'criteria': (3, 10, 0.03)} were used. For the preprocessing resize with (3/4, 3/4) und a median filter with (21,21) was used.

Difference of Gaussians

Rank	Average lifespan	Number of keypoints	DoG parameters				
			sigma ures	nfeat-Layers	nOctave-hold	edgeThres-Threshold	contrast-
1	87.69	2941.90	2.5	10	1	2	0.01
2	85.18	2900.75	1	10	5	2	0.01
3	84.37	2871.85	1	10	3	2	0.01
4	84.22	2942.00	1	10	1	2	0.01
5	81.27	4184.05	2.5	250	1	2	0.01
6	80.41	3984.60	1	250	1	2	0.01
7	79.71	3001.30	1.6	10	5	2	0.01
8	79.63	4687.75	1	250	5	2	0.01
9	79.42	2976.15	1.6	10	3	2	0.01
10	79.31	3848.60	1	250	3	2	0.01

Rank	Average lifespan	Number of keypoints	DoG parameters				
			sigma ures	nfeat- Layers	nOctave- hold	edgeThres- Threshold	contrast-
11	79.24	3107.00	1.6	10	1	2	0.01
12	77.42	943.95	2.5	250	1	2	0.04
13	77.41	943.55	2.5	10	1	2	0.04
14	77.25	2994.90	2.5	10	3	2	0.01
15	76.45	3016.20	2.5	10	5	2	0.01
16	76.09	890.80	1	10	1	2	0.04
17	76.01	892.90	1	250	1	2	0.04
18	75.84	4332.40	1.6	250	1	2	0.01
19	74.8	4392.80	1.6	250	5	2	0.01
20	74.79	904.00	1	10	3	2	0.04
21	74.74	905.65	1	250	3	2	0.04
22	74.41	4272.85	1.6	250	3	2	0.01
23	73.12	4419.35	2.5	250	3	2	0.01
24	72.66	1022.45	1.6	10	3	2	0.04
25	72.62	1026.15	1.6	250	3	2	0.04
26	72.4	1090.60	1	10	5	2	0.04
27	72.07	1098.25	1	250	5	2	0.04
28	72.05	4533.65	2.5	250	5	2	0.01
29	71.42	926.95	1.6	250	1	2	0.04
30	71.4	925.95	1.6	10	1	2	0.04
31	69.91	1064.95	1.6	10	5	2	0.04
32	69.89	1067.30	1.6	250	5	2	0.04
33	69.18	1072.10	2.5	10	3	2	0.04
34	69.16	1076.50	2.5	250	3	2	0.04
35	69.07	1120.85	2.5	10	5	2	0.04
36	69.01	1124.00	2.5	250	5	2	0.04
37	58.99	18722.05	1	250	1	10	0.01
38	57.79	17813.95	1	250	3	10	0.01
39	57.48	21075.80	1	250	1	15	0.01
40	56.41	21902.60	1	250	5	10	0.01
41	56.15	19688.35	1	250	3	15	0.01
42	55.97	19810.35	1.6	250	1	10	0.01
43	54.87	20097.25	1.6	250	5	10	0.01
44	54.83	24081.25	1	250	5	15	0.01
45	54.78	19657.95	2.5	250	1	10	0.01
46	54.76	19668.20	1.6	250	3	10	0.01
47	54.7	22305.45	1.6	250	1	15	0.01
48	53.5	22250.50	1.6	250	5	15	0.01
49	53.48	21804.40	1.6	250	3	15	0.01
50	53.37	20450.95	2.5	250	3	10	0.01
51	53.09	21990.85	2.5	250	1	15	0.01
52	53.02	20792.40	2.5	250	5	10	0.01
53	52.42	3724.85	1	10	3	10	0.01
54	52.18	3541.80	1	10	5	10	0.01
55	52.14	22636.85	2.5	250	3	15	0.01
56	51.9	22975.00	2.5	250	5	15	0.01
57	51.35	3735.90	1	10	1	10	0.01
58	51.14	3172.00	1	10	5	10	0.04
59	50.74	3026.95	1	10	3	10	0.04
60	50.44	3596.30	1	10	5	15	0.01
61	50.16	3793.00	1	10	3	15	0.01
62	49.99	2909.80	1	10	1	10	0.04
63	49.93	3787.10	2.5	10	1	10	0.01

Rank	Average lifespan	Number of keypoints	DoG parameters				
			sigma ures	nfeat- Layers	nOctave- hold	edgeThres- Threshold	contrast-
64	49.81	3571.05	1	250	3	10	0.04
65	49.76	3717.40	1.6	10	3	10	0.01
66	49.67	3287.95	1	10	5	15	0.04
67	49.52	4403.25	1	250	5	10	0.04
68	49.31	3341.85	1	250	1	10	0.04
69	49.31	3746.95	1.6	10	1	10	0.01
70	49.2	3797.25	1	10	1	15	0.01
71	49.11	3203.05	1	10	3	15	0.04
72	48.97	3732.25	1.6	10	5	10	0.01
73	48.56	3314.55	1.6	10	3	10	0.04
74	48.54	3124.90	2.5	10	1	10	0.04
75	48.39	3723.40	2.5	10	5	10	0.01
76	48.27	3702.05	2.5	10	3	10	0.01
77	48.18	3121.55	1.6	10	1	10	0.04
78	48.06	4853.25	1	250	5	15	0.04
79	48.05	3858.60	2.5	10	1	15	0.01
80	48.01	3780.90	1.6	10	3	15	0.01
81	47.92	3977.80	1	250	3	15	0.04
82	47.86	3701.40	2.5	250	1	10	0.04
83	47.7	3108.00	1	10	1	15	0.04
84	47.65	3337.35	1.6	10	5	10	0.04
85	47.6	3817.05	1.6	10	1	15	0.01
86	47.25	4179.85	1.6	250	3	10	0.04
87	47.22	3741.65	1	250	1	15	0.04
88	47.21	3800.00	1.6	10	5	15	0.01
89	47.05	3321.35	2.5	10	1	15	0.04
90	47.01	3426.00	2.5	10	3	10	0.04
91	46.86	3462.00	1.6	10	3	15	0.04
92	46.73	3487.60	2.5	10	5	10	0.04
93	46.69	3779.50	2.5	10	5	15	0.01
94	46.56	3294.60	1.6	10	1	15	0.04
95	46.53	4262.30	1.6	250	5	10	0.04
96	46.53	3695.25	1.6	250	1	10	0.04
97	46.48	4524.60	2.5	250	3	10	0.04
98	46.07	4127.75	2.5	250	1	15	0.04
99	46.01	3761.80	2.5	10	3	15	0.01
100	45.88	4676.80	2.5	250	5	10	0.04
101	45.85	3482.25	1.6	10	5	15	0.04
102	45.46	4665.45	1.6	250	3	15	0.04
103	45.33	3603.90	2.5	10	5	15	0.04
104	45.2	3555.35	2.5	10	3	15	0.04
105	44.88	4085.20	1.6	250	1	15	0.04
106	44.72	4744.60	1.6	250	5	15	0.04
107	44.65	5016.45	2.5	250	3	15	0.04
108	44.15	5164.05	2.5	250	5	15	0.04

Table 9: The results for the testing of the DoG parameters on 120 Hz videos. For the Lucas-Kanade parameters {'winSize': (15, 15), 'maxLevel': 10, 'criteria': (3, 10, 0.03)} were used. For the preprocessing resize with (3/4, 3/4) und a median filter with (21,21) was used.

Rank	Average lifespan	Number of keypoints	DoG parameters				
			sigma ures	nfeat- Layers	nOctave- hold	edgeThres- Threshold	contrast-
1	1226.61	545.00	1	10	5	10	0.01
2	1207.66	521.00	1	10	5	15	0.01
3	1197.45	983.00	1	250	5	10	0.01
4	1196.41	743.00	1	10	3	10	0.01
5	1193.68	659.00	1	10	3	15	0.01
6	1192.51	960.00	1	250	3	10	0.01
7	1186.09	1101.00	1	250	3	15	0.01
8	1177.75	1126.00	1	250	5	15	0.01
9	1107.33	1253.00	1.6	250	3	10	0.01
10	1077.32	1245.50	2.5	250	3	10	0.01
11	1073.13	947.50	1.6	10	3	10	0.01
12	1055.05	1353.00	1	250	1	10	0.01
13	1049.62	826.00	2.5	10	5	10	0.01
14	1048.24	862.00	2.5	10	3	10	0.01
15	1047.52	1228.50	2.5	250	5	10	0.01
16	1043.95	1023.00	1	10	1	10	0.01
17	1011.86	963.50	1.6	10	1	10	0.01
18	1009.7	1509.50	2.5	250	3	15	0.01
19	1002.57	1494.50	1.6	250	3	15	0.01
20	989.01	1486.00	2.5	250	5	15	0.01
21	983.16	933.00	1.6	10	3	15	0.01
22	972.35	1615.50	1	250	1	15	0.01
23	969.79	1324.00	1.6	250	1	10	0.01
24	968.24	1263.50	1.6	250	5	10	0.01
25	960.76	853.50	2.5	10	3	15	0.01
26	945.5	941.50	1.6	10	5	10	0.01
27	944.79	1467.00	1.6	250	5	15	0.01
28	927.03	820.50	2.5	10	5	15	0.01
29	891.41	1027.00	1	10	1	15	0.01
30	880.54	992.00	1.6	10	1	15	0.01
31	863.96	896.50	1.6	10	5	15	0.01
32	837.87	1603.50	1.6	250	1	15	0.01
33	813.83	947.50	2.5	10	1	10	0.01
34	772.08	1248.00	2.5	250	1	10	0.01
35	714.87	560.00	2.5	250	3	10	0.04
36	714.87	560.00	2.5	10	3	10	0.04
37	669.99	530.00	2.5	250	5	10	0.04
38	669.99	530.00	2.5	10	5	10	0.04
39	658.9	1023.00	2.5	10	1	15	0.01
40	658.36	615.00	1.6	250	1	10	0.04
41	657.9	614.50	1.6	10	1	10	0.04
42	628.25	601.50	2.5	10	3	15	0.04
43	626.36	544.00	1.6	250	3	10	0.04
44	626.36	544.00	1.6	10	3	10	0.04
45	621.03	601.50	2.5	250	3	15	0.04
46	615.79	1505.50	2.5	250	1	15	0.01
47	593.38	569.50	2.5	250	5	15	0.04
48	593.31	569.50	2.5	10	5	15	0.04
49	555.15	640.00	2.5	250	1	10	0.04
50	555.14	640.50	2.5	10	1	10	0.04
51	541.76	591.00	1.6	250	3	15	0.04
52	541.76	591.00	1.6	10	3	15	0.04

Rank	Average lifespan	Number of keypoints	DoG parameters				
			sigma ures	nfeat- Layers	nOctave- hold	edgeThres- Threshold	contrast-
53	527.8	537.00	1.6	250	5	10	0.04
54	527.8	537.00	1.6	10	5	10	0.04
55	517.36	660.00	1	250	1	10	0.04
56	517.36	660.00	1	10	1	10	0.04
57	489.85	567.50	1.6	10	5	15	0.04
58	489.85	567.50	1.6	250	5	15	0.04
59	477.93	699.50	1.6	250	1	15	0.04
60	477.9	699.50	1.6	10	1	15	0.04
61	446.39	703.50	1	10	1	15	0.04
62	446.39	703.50	1	250	1	15	0.04
63	377.76	730.00	2.5	250	1	15	0.04
64	377.75	730.50	2.5	10	1	15	0.04

Table 10: The results for the testing of the DoG parameters on 200 Hz videos. For the Lucas-Kanade parameters {'winSize': (15, 15), 'maxLevel': 10, 'criteria': (3, 10, 0.03)} were used. For the preprocessing resize with (3/4, 3/4) und a median filter with (21,21) was used.

6.2 Lucas-Kanade Testing

Rank	Average lifespan	Number of keypoints	Detector	Lucas-Kanade Parameters		
				winSize	maxLevel	criteria
1	563.96	1733.3	Shi-Tomasi	(31, 31)	3	(3, 4, 0.5)
2	563.96	1733.3	Shi-Tomasi	(31, 31)	10	(3, 4, 0.5)
3	547.1	1747.6	Shi-Tomasi	(31, 31)	0	(3, 4, 0.5)
4	541.12	1751.05	Shi-Tomasi	(31, 31)	10	(3, 10, 0.03)
5	541.12	1751.05	Shi-Tomasi	(31, 31)	3	(3, 10, 0.03)
6	538.38	1768.55	Shi-Tomasi	(31, 31)	0	(3, 10, 0.03)
7	520.58	1744.4	Shi-Tomasi	(15, 15)	3	(3, 30, 0.03)
8	520.22	1752.8	Shi-Tomasi	(15, 15)	10	(3, 30, 0.03)
9	517.47	1763.8	Shi-Tomasi	(15, 15)	0	(3, 30, 0.03)
10	500.32	608.85	Fast Hessian	(31, 31)	3	(3, 4, 0.5)
11	500.32	608.85	Fast Hessian	(31, 31)	10	(3, 4, 0.5)
12	494.02	602.45	Fast Hessian	(31, 31)	0	(3, 4, 0.5)
13	482.73	613.95	Fast Hessian	(31, 31)	3	(3, 10, 0.03)
14	482.73	613.95	Fast Hessian	(31, 31)	10	(3, 10, 0.03)
15	479.66	615.25	Fast Hessian	(31, 31)	0	(3, 10, 0.03)
16	465.11	616.35	Fast Hessian	(15, 15)	3	(3, 30, 0.03)
17	464.99	616.4	Fast Hessian	(15, 15)	10	(3, 30, 0.03)
18	462.93	616.85	Fast Hessian	(15, 15)	0	(3, 30, 0.03)
19	336.93	1885.3	DoG	(31, 31)	3	(3, 4, 0.5)
20	336.93	1885.3	DoG	(31, 31)	10	(3, 4, 0.5)
21	323.89	1915.25	DoG	(31, 31)	3	(3, 30, 0.03)
22	323.84	1915.6	DoG	(31, 31)	3	(3, 10, 0.03)
23	323.84	1915.6	DoG	(31, 31)	10	(3, 10, 0.03)
24	322.13	1958.5	DoG	(31, 31)	0	(3, 4, 0.5)
25	312.97	1989.3	DoG	(31, 31)	0	(3, 30, 0.03)
26	312.78	1988.5	DoG	(31, 31)	0	(3, 10, 0.03)
27	275.95	1807.5	Shi-Tomasi	(15, 15)	10	(3, 4, 0.5)
28	274.61	1804.65	Shi-Tomasi	(15, 15)	3	(3, 4, 0.5)

Rank	Average lifespan	Number of keypoints	Detector	Lucas-Kanade Parameters		
				winSize	maxLevel	criteria
29	267.92	1791.95	Shi-Tomasi	(15, 15)	0	(3, 4, 0.5)
30	255.98	1816.7	Shi-Tomasi	(15, 15)	3	(3, 10, 0.03)
31	255.45	1810.7	Shi-Tomasi	(15, 15)	10	(3, 10, 0.03)
32	254.23	1800.15	Shi-Tomasi	(15, 15)	0	(3, 10, 0.03)
33	237.31	1836.5	Shi-Tomasi	(5, 5)	3	(3, 30, 0.03)
34	237.24	1827.75	Shi-Tomasi	(5, 5)	10	(3, 30, 0.03)
35	235.75	1840.15	Shi-Tomasi	(5, 5)	0	(3, 30, 0.03)
36	174.71	868.8	Fast Hessian	(15, 15)	10	(3, 4, 0.5)
37	174.36	869.15	Fast Hessian	(15, 15)	3	(3, 4, 0.5)
38	166.63	851.6	Fast Hessian	(15, 15)	0	(3, 4, 0.5)
39	166.5	874.25	Fast Hessian	(15, 15)	10	(3, 10, 0.03)
40	165.96	874.45	Fast Hessian	(15, 15)	3	(3, 10, 0.03)
41	164.63	873.05	Fast Hessian	(15, 15)	0	(3, 10, 0.03)
42	153.33	884.1	Fast Hessian	(5, 5)	10	(3, 30, 0.03)
43	152.76	883.65	Fast Hessian	(5, 5)	3	(3, 30, 0.03)
44	150.35	883.65	Fast Hessian	(5, 5)	0	(3, 30, 0.03)
45	139.93	2079.7	Shi-Tomasi	(31, 31)	0	(3, 30, 0.03)
46	136.5	2059.4	Shi-Tomasi	(31, 31)	3	(3, 30, 0.03)
47	135.33	2619.3	Shi-Tomasi	(31, 31)	10	(3, 30, 0.03)
48	129.01	3423.35	DoG	(5, 5)	0	(3, 4, 0.5)
49	125.48	3455.9	DoG	(5, 5)	0	(3, 30, 0.03)
50	125.03	3462.35	DoG	(5, 5)	0	(3, 10, 0.03)
51	118.63	2708.3	DoG	(15, 15)	0	(3, 4, 0.5)
52	113.2	2771.25	DoG	(15, 15)	0	(3, 30, 0.03)
53	113.12	2773.3	DoG	(15, 15)	0	(3, 10, 0.03)
54	101.11	2082.25	Shi-Tomasi	(5, 5)	0	(3, 4, 0.5)
55	100.4	2081.3	Shi-Tomasi	(5, 5)	3	(3, 10, 0.03)
56	100.04	2061.35	Shi-Tomasi	(5, 5)	3	(3, 4, 0.5)
57	99.63	2124.15	Shi-Tomasi	(5, 5)	0	(3, 10, 0.03)
58	96.41	2088.85	Shi-Tomasi	(5, 5)	10	(3, 10, 0.03)
59	96.28	2105.3	Shi-Tomasi	(5, 5)	10	(3, 4, 0.5)
60	95.83	2884.9	DoG	(15, 15)	3	(3, 4, 0.5)
61	95.69	2882.25	DoG	(15, 15)	10	(3, 4, 0.5)
62	87.96	2938.95	DoG	(15, 15)	3	(3, 10, 0.03)
63	87.91	2938	DoG	(15, 15)	3	(3, 30, 0.03)
64	87.67	2939.35	DoG	(15, 15)	10	(3, 30, 0.03)
65	87.65	2941.15	DoG	(15, 15)	10	(3, 10, 0.03)
66	63.9	1495.7	Fast Hessian	(31, 31)	0	(3, 30, 0.03)
67	62.84	1505.25	Fast Hessian	(31, 31)	3	(3, 30, 0.03)
68	60.35	2677.35	Fast Hessian	(31, 31)	10	(3, 30, 0.03)
69	42.32	2810.8	Fast Hessian	(31, 31)	3	(3, 30, 0.03)
70	37.09	1507.6	Fast Hessian	(5, 5)	3	(3, 4, 0.5)
71	36.99	1503.5	Fast Hessian	(5, 5)	10	(3, 4, 0.5)
72	35.28	1504.05	Fast Hessian	(5, 5)	0	(3, 4, 0.5)
73	34.29	1517.35	Fast Hessian	(5, 5)	0	(3, 10, 0.03)
74	32.92	1518.7	Fast Hessian	(5, 5)	3	(3, 10, 0.03)
75	32.71	1527.25	Fast Hessian	(5, 5)	10	(3, 10, 0.03)
76	29.98	2728.35	Fast Hessian	(5, 5)	0	(3, 4, 0.5)
77	24.13	4456.85	DoG	(5, 5)	3	(3, 4, 0.5)
78	23.62	4474.55	DoG	(5, 5)	10	(3, 4, 0.5)
79	21.99	4520.15	DoG	(5, 5)	3	(3, 30, 0.03)
80	21.72	4529.8	DoG	(5, 5)	3	(3, 10, 0.03)
81	21.27	4532.3	DoG	(5, 5)	10	(3, 30, 0.03)
82	20.99	4547.35	DoG	(5, 5)	10	(3, 10, 0.03)

Rank	Average lifespan	Number of keypoints	Detector	Lucas-Kanade Parameters		
				winSize	maxLevel	criteria

Table 11: The results for the testing of the Lucas-Kanade parameters on 120 Hz videos. For each keypoint detector the best setting from the previous testing was used: Shi-Tomasi with {'qualityLevel': 0.5, 'minDistance': 61, 'maxCorners': 510, 'blockSize': 5}, DoG with {'sigma': 2.5, 'nfeatures': 10, 'nOctaveLayers': 1, 'edgeThreshold': 2, 'contrastThreshold': 0.01} and fast Hessian with {'nOctaves': 1, 'nOctaveLayers': 1, 'hessianThreshold': 50}. For the preprocessing resize with (3/4, 3/4) und a median filter with (21,21) was used.

Rank	Average lifespan	Number of keypoints	Detector	Lucas-Kanade Parameters		
				winSize	maxLevel	criteria
1	2605.77	523.5	Fast Hessian	(31, 31)	10	(3, 30, 0.03)
2	2605.77	523.5	Fast Hessian	(31, 31)	3	(3, 30, 0.03)
3	2605.77	523.5	Fast Hessian	(31, 31)	3	(3, 10, 0.03)
4	2605.77	523.5	Fast Hessian	(31, 31)	10	(3, 10, 0.03)
5	2566.14	524.5	Fast Hessian	(31, 31)	0	(3, 30, 0.03)
6	2566.14	524.5	Fast Hessian	(31, 31)	0	(3, 10, 0.03)
7	2129.63	562	Shi-Tomasi	(15, 15)	3	(3, 30, 0.03)
8	2129.63	562	Shi-Tomasi	(15, 15)	10	(3, 30, 0.03)
9	2121.36	572.5	Shi-Tomasi	(15, 15)	10	(3, 10, 0.03)
10	2121.36	572.5	Shi-Tomasi	(15, 15)	3	(3, 10, 0.03)
11	2115.73	659	Shi-Tomasi	(15, 15)	0	(3, 4, 0.5)
12	1940.98	658	Shi-Tomasi	(15, 15)	0	(3, 10, 0.03)
13	1938.87	654.5	Shi-Tomasi	(15, 15)	0	(3, 30, 0.03)
14	1328.32	519	DoG	(15, 15)	0	(3, 4, 0.5)
15	1314.65	649	Fast Hessian	(15, 15)	10	(3, 4, 0.5)
16	1314.65	649	Fast Hessian	(15, 15)	3	(3, 4, 0.5)
17	1275.17	544	DoG	(15, 15)	0	(3, 10, 0.03)
18	1273.9	544.5	DoG	(15, 15)	0	(3, 30, 0.03)
19	1264.87	537	DoG	(15, 15)	3	(3, 4, 0.5)
20	1264.87	537	DoG	(15, 15)	10	(3, 4, 0.5)
21	1245.89	665	Fast Hessian	(15, 15)	0	(3, 4, 0.5)
22	1229.21	541	DoG	(15, 15)	3	(3, 30, 0.03)
23	1229.21	541	DoG	(15, 15)	10	(3, 30, 0.03)
24	1221.98	543	DoG	(15, 15)	10	(3, 10, 0.03)
25	1221.98	543	DoG	(15, 15)	3	(3, 10, 0.03)
26	1166.5	715.5	Fast Hessian	(15, 15)	3	(3, 10, 0.03)
27	1166.5	715.5	Fast Hessian	(15, 15)	10	(3, 10, 0.03)
28	1165.36	713	Fast Hessian	(15, 15)	10	(3, 30, 0.03)
29	1165.36	713	Fast Hessian	(15, 15)	3	(3, 30, 0.03)
30	1165.19	722	Fast Hessian	(15, 15)	0	(3, 30, 0.03)
31	1148.57	724	Fast Hessian	(15, 15)	0	(3, 10, 0.03)
32	756.88	1035	Shi-Tomasi	(5, 5)	0	(3, 4, 0.5)
33	708.56	837.5	Shi-Tomasi	(5, 5)	0	(3, 30, 0.03)
34	705.72	1020.5	Shi-Tomasi	(5, 5)	3	(3, 10, 0.03)
35	699.27	1074.5	Shi-Tomasi	(5, 5)	3	(3, 30, 0.03)
36	697.37	1088	Shi-Tomasi	(5, 5)	3	(3, 4, 0.5)
37	696.48	1024.5	Shi-Tomasi	(5, 5)	10	(3, 10, 0.03)
38	686.91	1175	Shi-Tomasi	(5, 5)	10	(3, 4, 0.5)
39	678.79	1048.5	Shi-Tomasi	(5, 5)	10	(3, 30, 0.03)
40	668.96	954	Shi-Tomasi	(5, 5)	0	(3, 10, 0.03)
41	234.52	1872	Fast Hessian	(5, 5)	10	(3, 4, 0.5)

Rank	Average lifespan	Number of keypoints	Detector	Lucas-Kanade Parameters		
				winSize	maxLevel	criteria
42	226.74	1880.5	Fast Hessian	(5, 5)	3	(3, 4, 0.5)
43	191.89	1707	Fast Hessian	(5, 5)	0	(3, 4, 0.5)
44	178.23	1993.5	Fast Hessian	(5, 5)	10	(3, 10, 0.03)
45	177.73	1984	Fast Hessian	(5, 5)	10	(3, 30, 0.03)
46	175.7	1986	Fast Hessian	(5, 5)	3	(3, 30, 0.03)
47	173.88	2014	Fast Hessian	(5, 5)	3	(3, 10, 0.03)
48	173.31	1736	DoG	(5, 5)	3	(3, 4, 0.5)
49	168.44	1749	DoG	(5, 5)	10	(3, 4, 0.5)
50	165.64	1598.5	DoG	(5, 5)	0	(3, 4, 0.5)
51	151.69	1784	Fast Hessian	(5, 5)	0	(3, 30, 0.03)
52	149.69	1802	Fast Hessian	(5, 5)	0	(3, 10, 0.03)
53	147.31	1650	DoG	(5, 5)	0	(3, 30, 0.03)
54	146.73	1668	DoG	(5, 5)	0	(3, 10, 0.03)
55	144.36	1839	DoG	(5, 5)	3	(3, 30, 0.03)
56	139.03	1856	DoG	(5, 5)	3	(3, 10, 0.03)
57	133.63	1842	DoG	(5, 5)	10	(3, 30, 0.03)
58	132.63	1861.5	DoG	(5, 5)	10	(3, 10, 0.03)

Table 12: The results for the testing of the Lucas-Kanade parameters on 200 Hz videos. For each keypoint detector the best setting from the previous testing was used: Shi-Tomasi with {'qualityLevel': 0.5, 'minDistance': 1, 'maxCorners': 260, 'blockSize': 2}, DoG with {'sigma': 1.0, 'nfeatures': 10, 'nOctaveLayers': 5, 'edgeThreshold': 10, 'contrastThreshold': 0.01} and fast Hessian with {'nOctaves': 1, 'nOctaveLayers': 5, 'hessianThreshold': 100}. For the preprocessing resize with (3/4, 3/4) und a median filter with (21,21) was used.

6.3 Preprocessing

Rank	Average lifespan	Number of keypoints	Detector	Resize	Filter (filtersize)
1	1071.6	657.25	Shi-Tomasi	(0.5, 0.5)	Median (15)
2	1064.49	1004.8	Shi-Tomasi	(0.5, 0.5)	Median (21)
3	938.75	1473.15	DoG	(0.5, 0.5)	None ()
4	933.88	1454.85	DoG	(0.5, 0.5)	Gauss (5)
5	932.44	1446.7	DoG	(0.5, 0.5)	Median (5)
6	910.15	1371.05	DoG	(0.5, 0.5)	Gauss (15)
7	891.49	1270.15	DoG	(0.5, 0.5)	Gauss (21)
8	870.97	1319.8	DoG	(0.5, 0.5)	Median (15)
9	810.66	2200.6	Fast Hessian	(0.5, 0.5)	Gauss (21)
10	809.95	999	Fast Hessian	(0.5, 0.5)	Median (21)
11	799.59	2922.9	Fast Hessian	(0.5, 0.5)	Gauss (15)
12	787.99	1974.55	Fast Hessian	(0.5, 0.5)	Median (15)
13	774.66	1261.85	DoG	(0.5, 0.5)	Median (21)
14	763.09	3958.2	Fast Hessian	(0.5, 0.5)	Median (5)
15	762.33	4088.9	Fast Hessian	(0.5, 0.5)	Gauss (5)
16	742.68	4439.5	Fast Hessian	(0.5, 0.5)	None ()
17	665.65	502.75	Shi-Tomasi	(0.75, 0.75)	'gauss' (21)
18	642.33	951.85	Shi-Tomasi	(0.75, 0.75)	Median (15)
19	563.96	1733.3	Shi-Tomasi	(0.75, 0.75)	Median (21)
20	561.67	1249.95	Fast Hessian	(0.75, 0.75)	Gauss (21)
21	552.41	6340.25	Fast Hessian	(0.75, 0.75)	Median (5)
22	551.78	593.9	Shi-Tomasi	(0.75, 0.75)	Gauss (5)

Rank	Average lifespan	Number of keypoints	Detector	Resize	Filter (filtersize)
23	547.79	2975	Fast Hessian	(0.75, 0.75)	Gauss (15)
24	547.32	6822.65	Fast Hessian	(0.75, 0.75)	Gauss (5)
25	542.27	1906.85	DoG	(0.75, 0.75)	None ()
26	533.89	1374.25	Fast Hessian	(0.75, 0.75)	Median (15)
27	528.58	8234.5	Fast Hessian	(0.75, 0.75)	None ()
28	525.81	1897.45	DoG	(0.75, 0.75)	Gauss (5)
29	516.37	1917.95	DoG	(0.75, 0.75)	Median (5)
30	506.63	623.5	Shi-Tomasi	(0.75, 0.75)	None ()
31	500.32	608.85	Fast Hessian	(0.75, 0.75)	Median (21)
32	481.22	576.95	Shi-Tomasi	(1, 1)	Gauss (21)
33	462.73	588.55	Shi-Tomasi	(1, 1)	Gauss (15)
34	460.7	1498.45	DoG	(0.75, 0.75)	Gauss (15)
35	442.81	1406.6	Shi-Tomasi	(1, 1)	Median (15)
36	436.09	1431	DoG	(0.75, 0.75)	Gauss (21)
37	432.45	1008.8	Fast Hessian	(1, 1)	Median (15)
38	421.94	616.9	Shi-Tomasi	(1, 1)	Median (5)
39	417.32	1936.1	Fast Hessian	(1, 1)	Gauss (15)
40	415.57	8175.7	Fast Hessian	(1, 1)	Median (5)
41	412.37	9044.25	Fast Hessian	(1, 1)	Gauss (5)
42	399.93	1835.1	DoG	(0.75, 0.75)	Median (15)
43	388.86	902.95	Shi-Tomasi	(1, 1)	None ()
44	388.4	12727.4	Fast Hessian	(1, 1)	None ()
45	374.16	878.5	Shi-Tomasi	(1, 1)	Gauss (5)
46	346.29	2509.1	Shi-Tomasi	(1, 1)	Median (21)
47	340.55	2171.4	DoG	(1, 1)	None ()
48	337.07	1884.6	DoG	(0.75, 0.75)	Median (21)
49	328.38	2027.6	DoG	(1, 1)	Gauss (5)
50	320.17	2034.85	DoG	(1, 1)	Median (5)
51	304.51	1703.8	DoG	(1, 1)	Gauss (15)
52	269.55	1877.8	DoG	(1, 1)	Gauss (21)
53	234.29	2414.3	DoG	(1, 1)	Median (15)
54	173.85	2534.25	DoG	(1, 1)	Median (21)

Table 13: The results for the testing of the preprocessing algorithms on 120 Hz videos. For each keypoint detector the best setting from the previous testing was used: Shi-Tomasi with {'qualityLevel': 0.5, 'minDistance': 61, 'maxCorners': 510, 'blockSize': 5}, DoG with {'sigma': 2.5, 'nfeatures': 10, 'nOctaveLayers': 1, 'edgeThreshold': 2, 'contrastThreshold': 0.01} and fast Hessian with {'nOctaves': 1, 'nOctaveLayers': 1, 'hessianThreshold': 50}. For the Lucas-Kanade algorithm the best parameters from previous testing {'winSize': (31, 31), 'maxLevel': 3, 'criteria': (3, 4, 0.5)} were used.

Rank	Average lifespan	Number of keypoints	Detector	Resize	Filter (filtersize)
1	2869.93	863.00	Fast Hessian	(1, 1)	None ()
2	2683.58	921.50	Fast Hessian	(1, 1)	Gauss (5)
3	2650.80	555.00	Shi-Tomasi	(0.5, 0.5)	Median (5)
4	2605.77	523.50	Fast Hessian	(0.75, 0.75)	Median (21)
5	2593.13	1019.00	Fast Hessian	(1, 1)	Median (5)
6	2374.17	665.50	Shi-Tomasi	(0.5, 0.5)	Gauss (15)
7	2286.93	914.50	Shi-Tomasi	(0.5, 0.5)	Gauss (21)
8	2241.20	928.00	Fast Hessian	(1, 1)	Gauss (15)

Rank	Average lifespan	Number of keypoints	Detector	Resize	Filter (filtersize)
9	2228.59	654.50	Shi-Tomasi	(0.5, 0.5)	None ()
10	2118.86	576.00	Shi-Tomasi	(0.75, 0.75)	Median (21)
11	2065.14	811.00	Fast Hessian	(1, 1)	Gauss (21)
12	1985.94	918.00	Fast Hessian	(1, 1)	Median (15)
13	1959.97	542.50	Fast Hessian	(1, 1)	Median (21)
14	1939.53	622.50	Shi-Tomasi	(0.75, 0.75)	Median (15)
15	1866.99	553.50	Shi-Tomasi	(0.75, 0.75)	Gauss (5)
16	1856.85	736.00	Shi-Tomasi	(0.75, 0.75)	Median (5)
17	1730.41	915.00	Shi-Tomasi	(0.75, 0.75)	None ()
18	1698.85	696.00	Shi-Tomasi	(0.75, 0.75)	Gauss (15)
19	1541.18	953.50	Shi-Tomasi	(1, 1)	Median (5)
20	1525.34	1328.00	Shi-Tomasi	(1, 1)	None ()
21	1328.32	519.00	DoG	(0.75, 0.75)	Median (21)
22	1312.39	1327.00	Shi-Tomasi	(1, 1)	Median (21)
23	1189.23	1566.50	Shi-Tomasi	(1, 1)	Median (15)
24	885.86	1018.00	DoG	(1, 1)	Median (21)
25	878.38	2364.00	Shi-Tomasi	(0.75, 0.75)	Gauss (21)
26	868.88	687.00	DoG	(0.75, 0.75)	Median (15)
27	838.26	573.50	DoG	(0.75, 0.75)	Gauss (15)
28	797.31	796.50	DoG	(1, 1)	Median (15)
29	754.15	751.50	DoG	(0.75, 0.75)	Gauss (21)
30	667.01	3353.50	Shi-Tomasi	(1, 1)	Gauss (21)
31	603.65	699.50	DoG	(1, 1)	Gauss (21)
32	580.22	724.50	DoG	(1, 1)	Gauss (15)

Table 14: The results for the testing of the preprocessing algorithms on 200 Hz videos. For each keypoint detector the best setting from the previous testing was used: Shi-Tomasi with {'qualityLevel': 0.5, 'minDistance': 1, 'maxCorners': 260, 'blockSize': 2}, DoG with {'sigma': 1.0, 'nfeatures': 10, 'nOctaveLayers': 5, 'edgeThreshold': 10, 'contrastThreshold': 0.01} and fast Hessian with {'nOctaves': 1, 'nOctaveLayers': 5, 'hessianThreshold': 100}. For each detector different Lucas-Kanade parameters were used. They were set to {'winSize': (15, 15), 'maxLevel': 3, 'criteria': (3, 10, 0.03)} for Shi-Tomasi, {'winSize': (15, 15), 'maxLevel': 0, 'criteria': (3, 4, 0.5)} for Dog and {'winSize': (31, 31), 'maxLevel': 3, 'criteria': (3, 10, 0.03)} for fast Hessian.

References

John L Barron, David J Fleet, Steven S Beauchemin, and TA Burkitt. Performance of optical flow techniques. In *Proceedings 1992 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 236–242. IEEE, 1992.

Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer vision and image understanding*, 110(3):346–359, 2008.

Jean-Yves Bouguet et al. Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm. Technical report, Intel Corporation, Microprocessor ResearchLabs, 1999.

Julie Delon. Midway image equalization. *Journal of Mathematical Imaging and Vision*, 21(2):119–134, 2004.

Benedikt V Ehinger, Katharina Gross, Inga Ibs, and Peter Koenig. A new comprehensive eye-tracking test battery concurrently evaluating the pupil labs glasses and the eyelink 1000. *PeerJ*, 7:e7086, 2019.

Mario Enriquez, Colin Swindells, and Ricardo Pedrosa. Method and apparatus for tracking eye movement, 2010. US Patent 7,736,000.

Andrej Fogelton and Wanda Benesova. Eye blink detection based on motion vectors analysis. *Computer Vision and Image Understanding*, 148:23–33, 2016.

Ben Galvin, Brendan McCane, Kevin Novins, David Mason, and Steven Mills. Recovering motion fields: An evaluation of eight optical flow algorithms. In *British Machine Vision Conference*, pages 195–204, 1998.

Steffen Gauglitz, Tobias Höllerer, and Matthew Turk. Evaluation of interest point detectors and feature descriptors for visual tracking. *International journal of computer vision*, 94(3):335–360, 2011.

Christopher Gundler. Beyond pixels: Semantic representations in the context of video-oculography. *Bachelor's thesis at University of Osnabrück*, 2018.

Christopher G Harris, Mike Stephens, et al. A combined corner and edge detector. In *Alvey vision conference*, pages 147–151, 1988.

Kenneth Holmqvist. Common predictors of accuracy, precision and data loss in 12 eye-trackers. `https://www.researchgate.net/publication/321678981_Common_predictors_of_accuracy_precision_and_data_loss_in_12_eye-trackers/citations`, 2017. Accessed:2019-11-17.

Berthold KP Horn and Brian G Schunck. Determining optical flow. *Artificial intelligence*, 17(1-3):185–203, 1981.

Itseez. The opencv reference manual. `https://docs.opencv.org/4.1.1/`, 2019. Accessed: 2019-17-11.

David G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

Bruce D. Lucas, Takeo Kanade, et al. An iterative image registration technique with an application to stereo vision. *Proceedings of Imaging Understanding Workshop*, pages 121–130, 1981.

mindQ. eyetrax. `https://www.eyetrax.de/index_en.html`, 2018. Accessed: 2019-11-08.

Raúl Rojas. Lucas-kanade in a nutshell. Technical report, Freie Universit at Berlin, Dept. of Computer Science, 2010.

Jianbo Shi and Carlo Tomasi. Good features to track. Technical report, Cornell University, 1993.

Tinne Tuytelaars and Krystian Mikolajczyk. Local invariant feature detectors: A survey. *Foundations and trends in computer graphics and vision*, 3(3):177–280, 2008.

Paul Viola, Michael Jones, et al. Rapid object detection using a boosted cascade of simple features. *CVPR (1)*, 1(511-518):3, 2001.

List of Figures

List of Tables

Abbreviations

DoG Difference of Gaussian

HoD Heatmap histogram of Directions

SIFT Scale-Invariant Feature Transform

SURF Speeded-Up Robust Features

Declaration of Authorship

I, Lars Berend Brandt, hereby certify that the work presented here is, to the best of my knowledge and belief, original and the result of my own investigations, except as acknowledged, and has not been submitted, either in part or whole, for a degree at this or any other university.

.....

signature

.....

city, date