Department of Humanities
Institute of Cognitive Science

Bachelor Thesis

An evaluation of different motion descriptors in head-mounted eye-tracking videos for movement identification

by

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

November 16, 2019

First Supervisor:
Dr. Matthias Temmen

Second Supervisor:
Prof. Dr. Gunther Heidemann

Acknowledgements

I would like to thank

I would also like to thank in particular .

Additionally, I would like to thank

Abstract

In this thesis, various approaches for generating motion descriptors are evaluated, including pixel-based methods, i.e. SSIM and dense optical flow, and feature-based methods using algorithms like Lucas-Kanade, Gabor or Pyramid Histograms of Oriented Gradients. This thesis will provide the basis for the identification of eye movements in videos without previous calculations of gaze and a 3d model of the eye, since they are costly and not easy to come by. Approaches to find features in eye tracking videos are compared qualitatively, with a main focus on three properties. First, the quantity and distribution of features over time is calculated and compared. Second, the semantical relevance of features is evaluated, i.e. if features are positioned at the edge of the pupil or eyelids. Third, the robustness over time and different videos of the found features is analyzed, in particular if they are located in the same part of the eye even when it is moving or after a blink happens. To evaluate over time, sparse optical flow is used to track the found features. Data is provided by mindQ from the project eyeTrax, comprising 20 videos of 10 participants (one video per eye), which were doing multiple, simple tasks in virtual reality in about 5 minutes.

Contents

1 Introduction

Even though

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. Following is the method to solve these problems used in this thesis, the Lucas-Kanade method. It includes the neighborhood of a pixel in the calculation.

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 of the 3D world humans exist in, first and foremost stereo vision, computation of optical flow and therefore the movement in the 3D world, is possible. Optical flow can be expressed in a mathematical formalism, which will be explained in the following. First, the following constraint is assumed: The brightness of a pixel does not change when moving through time an space (Horn and Schunck, 1981, p.187). Under this assumption the movement of a pixel in a video can be viewed as follows:

$$E(x, y, t) = E(x + \delta x, y + \delta y, t + \delta t) \tag{1}$$

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 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.

Division by δt yields:

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

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

$$E_x u + E_y v + E_t = 0 \tag{2}$$

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, Equation 2 becomes solvable, since we can solve for u and v, with 5 equations given a 4-neighborhood or 9 equations given a 8-neighborhood for 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 difference 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. In this project this problem occurs particularly, 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.
blinks?
Though these problems occur, with certain adaptions of the algorithm it proves to be the one 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, this 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. (2001). 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 10 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 can 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 their neighborhood 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 et al., 2008, p. 179). In order to achieve the best results with the Lucas-Kanade algorithm, the key point detection has to be optimized. For this reason, multiple key point detection algorithms can be compared and optimised.A vast amount of detection algorithms 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 extrema in x-direction yields pixel laying on an edge. If this is accompanied by an extrema in y-direction as well, one can see this as a corner, which are regions that have been viewed as interesting for a long time(Gauglitz et al., 2011, p. 337).
- 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).
- 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 high computational cost.

3 Sparse Lucas Kanade Optical Flow for Eye Movement Detection

In the following sections a method to search for the best setup for eye-movement analysis is carved out. These setups consisting of preprocessing, key point detection and Lucas-Kanade optical flow. Firstly, these setups are explained. Secondly, in subsection 3.1, key point detection in general is presented and the algorithms used in this work are explained in further detail. Thirdly, in subsection 3.2, the structure for evaluation of these setups are defined. Lastly, the testing is presented in general on the scripts being used.

The sparse version of Lucas' and Kanade's method calculates the optical flow on a set of key points in the image. 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 optical flow. Based on the Lucas-Kanade method, the locations of these points are estimated on the next frame. If it is possible, the optical flow method can use the locations as new input for the next frame, repetitively. To calculate 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.

3.1 Keypoint Detection

A comparison of key point detectors and feature descriptors was 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.

3.1.1 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:

$$M(x,y) = \begin{bmatrix} \sum\limits_{u,v} w_{u,v} \cdot [I_x(x+u,y+v)]^2 & \sum\limits_{u,v} w_{u,v} \cdot I_x(x+u,y+v) I_y(x+u,y+v) \\ \sum\limits_{u,v} w_{u,v} \cdot I_x(x+u,y+v) I_y(x+u,y+v) & \sum\limits_{u,v} w_{u,v} \cdot [I_x(x+u,y+y)]^2 \end{bmatrix} \tag{3}$$

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 location, respectively. $w_{u,v}$ is a window and Gaussian

weighting function, so the summations take place over the neighborhood of I(x,y). Based on the eigenvalues λ_1 and λ_2 of M, the image patch can be classified as uniform (both eigenvalues small), edge ($\lambda_1 small and \lambda_2 large, or vice versa$) or a corner (both values large.) Corners localized by this algorithm yield supposedly good features to track (Shi and Tomasi, 1993, p. 3).

3.1.2 Difference of Gaussian

Difference of Gaussian (DoG), 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, an image is convolved with Gaussian kernels, having differing standard variations leading to various blurred versions of it. 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. 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. In further analysis, candidates with low contrast and points along an edge are discarded, with an approach similar to corner detection (Lowe, 2004, pp. 94-98).

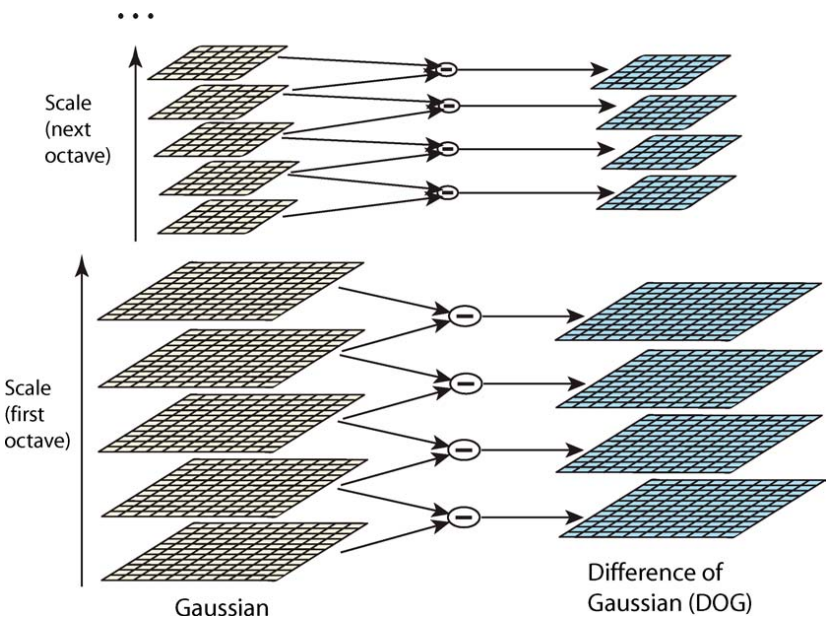


Figure 1: Difference of Gaussians is achieved by repeated convolution of the input image with Gaussian kernels and subsampling. Then, the octave layers are subtracted from each other.

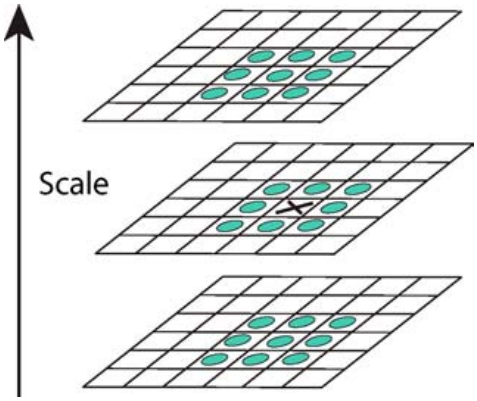


Figure 2: Candidates for key points are local extrema in comparison to their neighbors in the same, and neighboring octave layers.

3.1.3 Fast Hessian

Another Blob detector, the fast Hessian detector, was proposed by Bay et al. (2006) for Speeded Up Robust Features (SURF), which was developed to enhance SIFT. Fast Hessian detects key points by approximation of the Hessian matrix, which is obtainable by convolution of the image with a Gaussian second-order derivatives. Since this is very costly, the convolution is approximated by simple box filters, which can be computed in constant time using the integral image (Viola et al., 2001). The scale-invariance is achieved by upscaling these filters and key points are chosen based on the highest determinant of the resulting, (fast) approximated, Hessian matrices in their corresponding scale. Similar to the difference of Gaussian approach, candidates of low contrast are rejected(Bay et al., 2006).

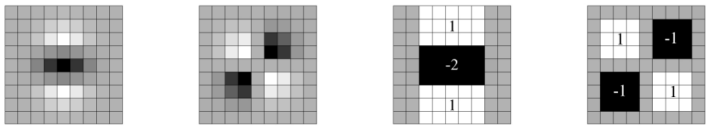


Figure 3: Second order Gaussian derivatives (left) are approximated as box filters (right) to receive an approximation of the Hessian matrix.

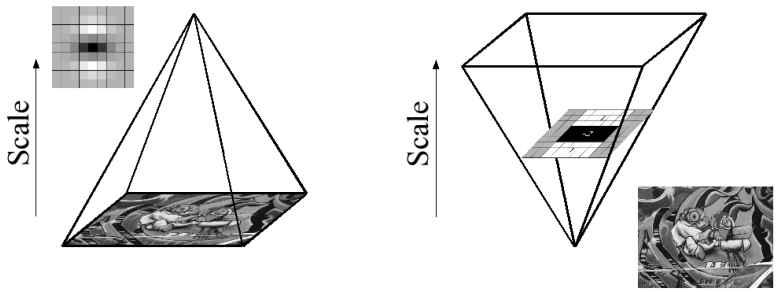


Figure 4: As opposed to the upscaling being done in key point detection with the difference of Gaussian approach (left), the image is not supsambled, but the box-filter is upscaled to include different scales.

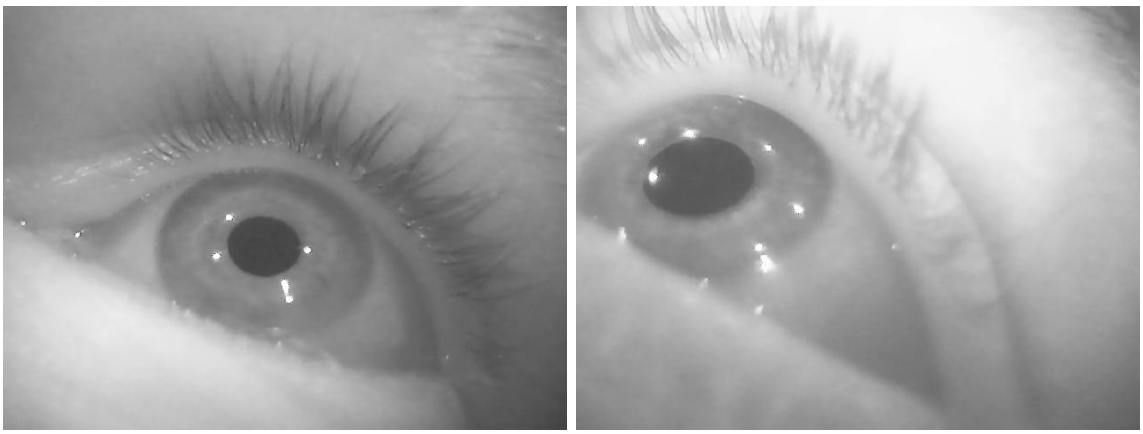
3.2 Setup for Evaluation

In the following subsections, a setup to evaluate three methods to detect key points in eye-videos which can be tracked by the method of Lucas and Kanade. In the beginning, the dataset is explained, followed by qualitative measures to evaluate different setups to calculate optical flow with the Lucas-Kanade method, each including preprocessing of the data and key point detection.

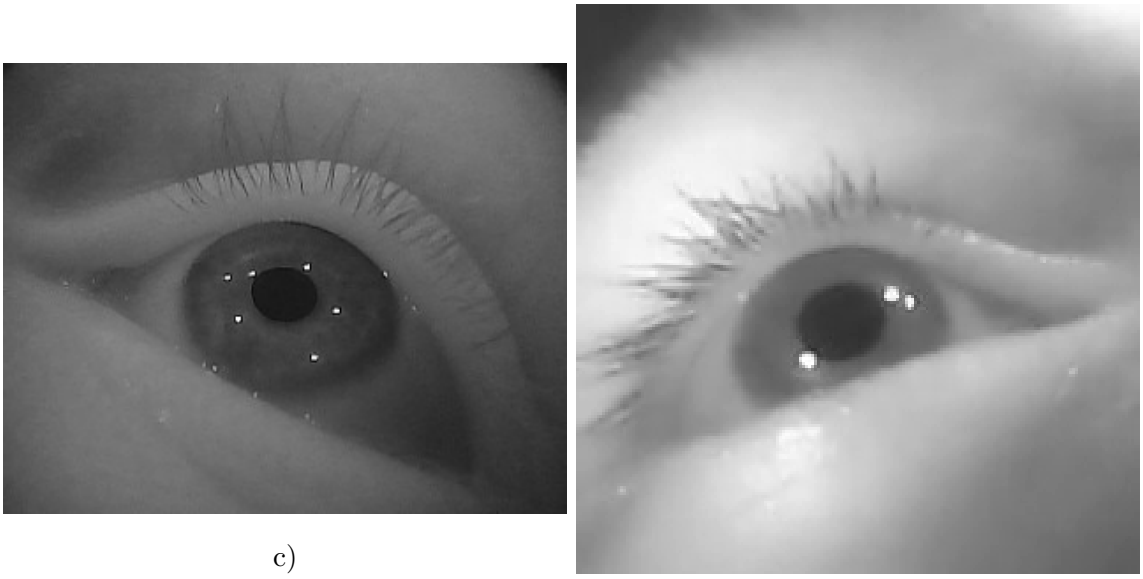
noch me intro

3.2.1 Dataset

The data was acquired by filming a subject 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 Hz and the videos have a resolution of 640 x 320 pixel. 1 subject was recorded doing the same test, but at a frame-rate of 200 Hz and a resolution of 192 x 192 pixel. 8 subjects were recorded doing an older version of the test in virtual reality, which takes about 3 minutes, also at a framerate of 120 Hz and a resolution of 640 x 320 pixel. This dataset was annotated with timestamps of blinks. Examples for each dataset can be seen in Figure 5. 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 5b), both canthi, the corners of eyes where the eyelids meet, are not visible in every video. Furthermore, unmoving, strong structures can be viewed in Figure 5a) in the upper right corner. Beside this and noise in the recordings, another disruption of videos is visible. One infrared light can be seen turning on and off, leading to a so-called flicker in large parts of the images, as can be seen in Figure 6. Even if it is not a very large dataset, it enables a lot of research with a wide variation of data.

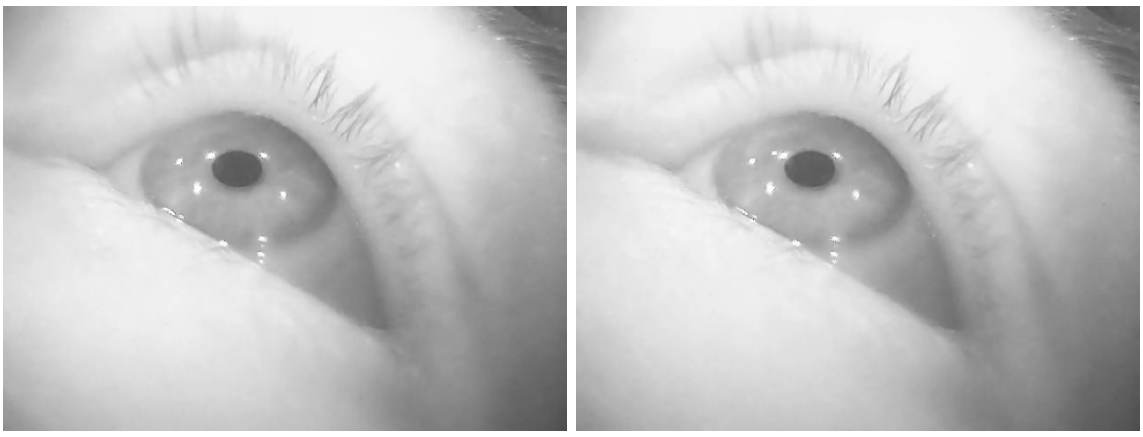


a) b)



c) d)

Figure 5: Variations of data: **a)** and **b)** 640 x 320 @ 120 fps. **c)** 320 x 240 @ 120 frames 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.



a) b)

Figure 6: Flicker
Flashing of light leads to a 'flicker'. Consecutive frames vary in brightness.

3.2.2 Measures

In the following section, the qualitative measures taken 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, measured in another way or via human validation. As opposed to such quantitative measures, qualitative ones can be helpful to compare these methods in relation to each other.

Quantity and Runtime

To have a basic measure to compare detection algorithms, found key points are counted. Additionally, parts in the video, in which no key points can be found or have been tracked are noted. To have 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 trackable points. On the other hand, 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.
Another simple measure is the runtime of the overall algorithm. Even though the application is not supposed to be in real time, a faster algorithm which yield the same results as another is superior to it.

Lifespan

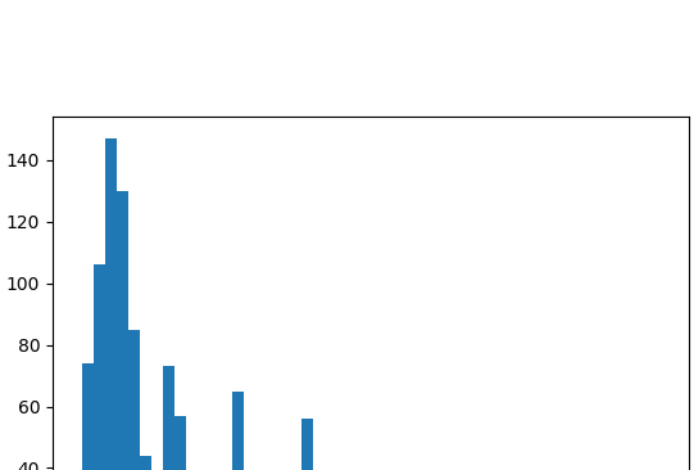


Figure 7: Histogram of quantity of keypoints being tracked over time in frames

beschrift

A second measure for comparison is how long 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, the average lifespan over all videos is calculated and the best few are selected. In further analysis, a histogram is helpful, see section 7. The data is visible for a part of a video comprising 1000 frames and using the Shi-Tomasi corner detector. The data is rounded to decades. A peak is visible at a lifespan of 80 frames with over 140 tracks. A lot of

- how long can optical flow track? - important to see with distribution

aktuelles
histogra
beschrei
dafür, in
params,
lifespan,

Spatial Distribution of Key Points

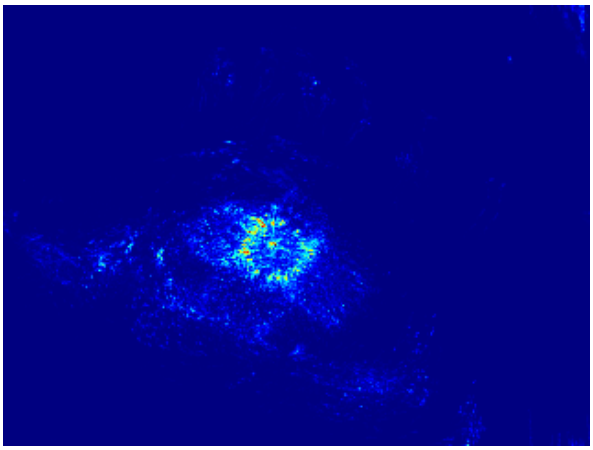


Figure 8: Heatmap of duration of tracking in image plane.

A third qualitative measure for the detection algorithms is where key points are located. Included is the location of points that are found by a detection algorithm, but also the locations which have been estimated by the optical flow algorithm. Figure 4 shows a heatmap which presents this information. By comparison with the actual image in the video it is possible to see if the locations are semantically relevant. Distribution over the whole image are desired. Especially so over moving parts, as this is information directly valuable for the task at hand, being 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 these measures 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 well 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.

3.3 Preprocessing

To improve the accuracy of movement detection it is of importance to remove irregularities from the data. The techniques to process the videos before detecting key point 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. Destruction of those could lead to a better measurment, because they are unmoving.The smaller sizes and (median-) filtering leads to a larger difference of pixels beside each other and therefore more key points to find To test if the flicker, as explained in subsubsection 3.2.1, has significant 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.

entschei
ob du di
stören w
oder nic

ref

bilder ei
gen

3.4 Testing

In this section, implementations to gain information to measure, as explained in subsection 3.2 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 tested 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 tested with fixed optical flow and key point detection. Afterwards, key point detectors will be tested again to validate the results. What follows is a listing of parameters with a short explanation.

vlt eine nung?

3.4.1 Shi-Tomasi Corner Detector

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

- maxCorners. This is the maximum number of returned key points.
- qualityLevel. 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. The minimal Euclidian distance between points to be considered a key point.
- blockSize. Size of the neighborhood included by the structure tensor.

3.4.2 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 with and key points are detected with cv2.xfeatures2d.SURF_create().detect() (Itseez, 2019). The following parameters were tested:

- hessianThreshold. The main part of key point detection with this detector is the threshold for the determinant of the approximated Hessian matrix.
- nOctaves. How often the filter is upscaled to yield different scales.
- nOctaveLayers. How often the filter is upscaled at each octave.

3.4.3 Differences of Gaussian

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

- nfeatures. The number of best key points to hold on to, ranked by local contrast.
- nOctaveLayers. The number of times the Gaussian convolution is applied to each octave with higher standard deviation to yield blurred versions of the octave. Returns difference of Gaussian by subtraction to the function. Differing standard deviations are calculated automatically by the resolution.
- Octaves. The number of times, the input image is downsampled.
- contrastThreshold. This threshold rejects key point candidates which have a low contrast.
- edgeThreshold. This threshold rejects key point candidates which are located on edges, and not corners.
- sigma. The standard deviation applied to the first octave of the input image.

3.4.4 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. The window in which the algorithm searches in the new frame for the new location of the key points, on all levels.
- maxLevel. The maximum number of levels of reduction to use.
- criteria (cv.TERM_CRITERIA_EPS | cv.TERM_CRITERIA_COUNT) Specifies the criterion to stop the search algorithm, either by number of iterations (COUNT) and/or after the search window moves by less than epsilon (EPS).

3.4.5 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:

wie mac
ich das
abkürzu
opt flow

- Downsampling by bilinear interpolation, as is used by cv2.resize()
- Noise-removal: Gaussian kernels, as well as median filtering is being used. 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. Keypoint detectors are tested on the videos with removed global flicker, to improve the optical flow algorithm.

- explain scripts: testing.py, gfeattesting.py etc cycle: testing detectors with fixed values for lkparams und preprocessing dann fixed detectors, lk params dann fixed detectors, preprocessing dann noch einmal keypoint detectors

4 Results

In the following section, results of testing are presented. First, an evaluation of each step, the comparison of parameters for each key point detection algorithm, Lucas-Kanade optical flow and preprocessing is analyzed, with a main focus on low level measures like average lifespan and runtime. Thereafter, having compared each setting on these simple measures, insight is given for more sophisticated measures explained in subsection 3.2.

4.1 Comparison by average lifespan and runtime

When comparing the average lifespan of various settings, the most remarkable result is the difference between the two videos taken by cameras with a frame-rate of 120 Hz and those with 200 Hz. All key point detectors show a notable higher lifespan and runtime for this increase in frame rate and lower resolution. Furthermore, The Shi-Tomasi detector shows the highest performance in these measures, concerning the videos with 120 fps. It is second for the average lifespan in videos with 200 fps after the difference of Gaussian detector, but is the fastest one of all three. The Fast Hessian key point detector performs worst in the videos with 200 fps, but is in second place for videos with the lower frame-rate. Interestingly, the difference of Gaussian approach shows the worst performance concerning average lifespan in 120 fps videos, but the best in 200 fps videos. The runtime is much lower in both classes of videos for the Shi-Tomasi detector than the other two, which show no significant difference in the one with lower frame-rate. The difference of Gaussian detector is also a bit faster in detection in videos with 200 fps compared to the fast Hessian approach.

When looking at different values for parameters of each algorithm, beginning with the Shi-Tomasi corner detector. A medium quality level seems to perform better than higher or lower values. A large distance between corner points proves to be a good approach, since higher values have a better performance. Also, larger neighborhood to calculate the covariation matrix proves to be more successful than smaller ones. Furthermore, the restriction by a maximum of detected corners does not seem to change a lot in average lifespan, and only slightly in runtime. For the fast Hessian detector, the longest average lifespan yields a combination of parameters with just one octave and only one octave layer and a moderately small Hessian threshold of 100. In general, combinations with a low number of octaves seem to prove superior. Good results yield low to moderate numbers of octave layers and the Hessian threshold offers a high variance. For the difference of Gaussian approach, a low contrast threshold seems to be most important for a good average lifespan, followed by the edge threshold. High and moderate values for sigma and octave layers yield better results than lower ones.

5 Discussion

Since midway equalization is a strictly global function, and does not take into account flickering in only small parts of the videos, the result is not as perfect as one would imagine, but it would be too much of work to extend this to local flicker (see Delon and Desolneux (2010)).

Future research

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, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.

Jean-Yves Bouguet et al. Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm. *Intel Corporation*, 5(1-10):4, 2001.

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

Julie Delon and Agnes Desolneux. Stabilization of flicker-like effects in image sequences through local contrast correction. *SIAM Journal on Imaging Sciences*, 3(4):703–734, 2010.

Ben Galvin, Brendan McCane, Kevin Novins, David Mason, Steven Mills, et al. Recovering motion fields: An evaluation of eight optical flow algorithms. In *BMVC*, volume 98, 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, 2011.

Christopher G Harris, Mike Stephens, et al. A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 10–5244. Citeseer, 1988.

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

Itseez. *The OpenCV Reference Manual*, 4.1.1 edition, July 2019.

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. 1981.

Raúl Rojas. Lucas-kanade in a nutshell. *Freie Universit at Berlinn, Dept. of Computer Science, Tech. Rep*, 2010.

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

Tinne Tuytelaars, Krystian Mikolajczyk, et al. 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

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