Computer Vision 1 - Assignment 3

Ruben-Laurentiu Grab
ruben.grab@student.uva.nl 11609923

Andrei Marius Sili
andrei.sili@student.uva.nl 11659416

1 Introduction

In this report we investigate different algorithms for optical flow detection, corner points detection, and also investigate how we can combine them in order to apply them on short video clips. We take an empirical approach, experimenting with different setups and parameters, reporting our results. We based our decisions on the knowledge gained during the lectures and individual study. Whenever possible, we provide theoretical grounding for our findings.

2 Harris Corner Detector

Question 1

In implementing the Harris Corner Detector, some choices had to be made regarding some of the parameters, namely the spread of the Gaussian filter σ, the size of the sliding window n, and the threshold for recognising corners, which we denote by γ. According to some example implementations we have seen, we chose to set the window size proportional to the spread of the Gaussian filter, $n = 6\sigma$. We hold σ constant at the value 1.5 and we vary γ in the range $20, 100, 1000, 10000$. The choice of this range was mostly empirical as there does not seem to be a consensus on what this value should be. Some implementations start with a low threshold of 20, while some others recommend a threshold of 10000 [1].

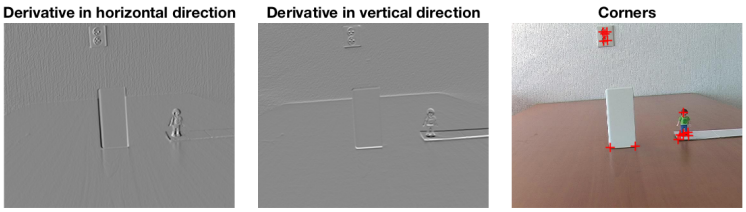


Figure 1: Corner detection on Person Toy image with threshold 10000

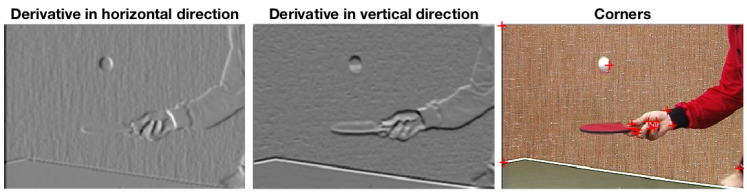
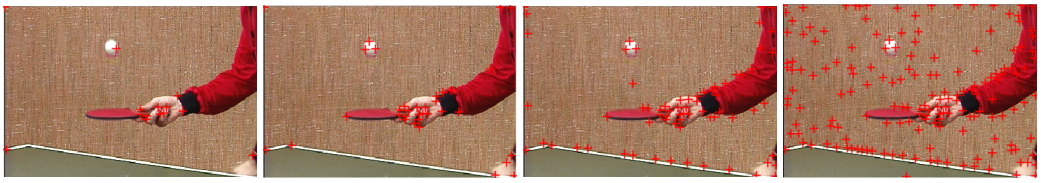


Figure 2: Corner detection on Ping Pong image with threshold 10000



(a) $\gamma = 10000$ (b) $\gamma = 1000$ (c) $\gamma = 100$ (d) $\gamma = 20$

Figure 3: Threshold experiment

Figures 1 and 2 show the results for a threshold of 10000. It can be seen that most corners are well detected, while some others are not. For example, the corner of the ping-pong table and the top corners of the white object are not detected at this threshold. Varying the threshold in a sense, gives a trade-off between precision and recall. Of course, the lower the threshold, the more corners one will detected, but most of these might be false positives. An interesting suggestion was to incorporate some sort of prior into the algorithm that estimates the amount of corners expected in an image, which could be a learned parameter. A threshold can then be dynamically adjusted to output the desired number of corners. Yet another way is by using the statistics of the cornerness matrix, such as mean and standard variation of the matrix values. In this way, open could make the detector adapt to the information in the image instead of taking a hard line approach. Our experiments show that for the the ping-pong image, a good threshold value is 1000. This achieves a good balance between precision and recall, since, for example, the edge of the table is detected, but the wall texture does not falsely show any corners. Figures 3a 3b 3c 3d depict the experiment.

The Harris Corner Detector is a rotation invariant algorithm. This is because the eigenvalues of the auto-correlation matrix remain the same. Only the orientation of the derivatives change, but the magnitude stays constant. Figures 5a 5b 5c 5d show the results of rotating the image successively by 90 degrees. Because the way MATLAB generates 0-padded images for non-square angles of rotation, the detector will no longer provide identical results, especially around the corner boundaries of the image. This effect can be seen in figure 5e. Figure 5f is the original non-rotated version and is shown for reference.

Question 2

The Shi and Tomasi corner detector is almost identical to the Harris corner detector, with the only difference being in how the cornerness score is calculated. That is, the Shi and Tomasi detector calculates the cornerness response as

$$R = \min(\lambda_1, \lambda_2)$$

where λ_1 and λ_2 are the eigenvalues of the auto-correlation matrix for each point (x, y). This criteria simply states that both eigenvalues need to be higher that a certain threshold value, which is still user defined, in order for the point to be considered a corner.

This cornerness function seems a lot simpler than Harris' initial proposal, mainly because this function does not depend on any parameters (i.e. the k constant). However, it should be noted that because the criteria applies directly to each individual eigenvalue, those need to be determined first. So one needs to find the eigen decomposition of the matrix. Generally, finding the eigenvalues of a matrix is resource intensive, but this is mitigated by the fact that we are only concerned with 2x2 matrices.

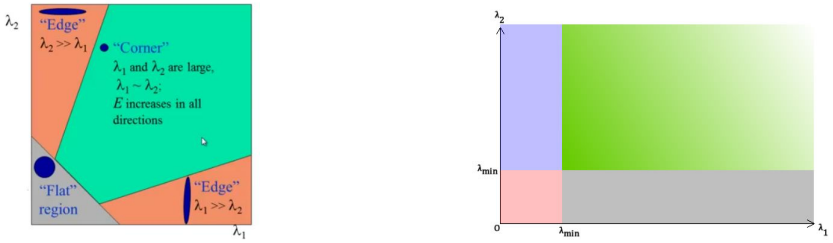
Using this criteria, it is still the case that both eigen values need to be large in order for the cornerness value to be high. Considering the 3 scenarios presented in the assignment:

a if both eigenvalues are near 0, then the the cornerness response will be low.

b if only one of the two eigenvalue is big, then the cornerness response will still be low, since the function picks the lowest of the two.

c if both eigenvalues are big, then the cornerness response will be high.

In essence, the intuition is the same, with the only difference being that the boundaries defined by the threshold are vertical or horizontal, rather than diagonal. Both algorithms will detect corners if both



(a) Harris

(b) Shi and Tomasi

Figure 4: Corner detector regions.

eigenvalues are high, but Shi and Tomasi experimentally show that their response function performs better [2]. Figure 4 shows the two cornerness regions for Harris and Shi and Tomasi detectors.

3 Optical Flow with Lucas Kanade Algorithm

Question 1

The results of running the Lucas-Kanade algorithm that we implemented on the two data-sets can be seen in figure 6.

Question 2

Operation scale Lucas-Kanade algorithm is a local optical flow algorithm. For a given pixel, it takes into consideration only the pixels that are in its neighbourhood. Because of this property, it suffers from what is called in the literature as "*The aperture problem*"(see fig. 7). It has to be mentioned that the size of the neighbourhood can be set by a parameter. However, one must carefully choose the window size, as a too large window size may cause a point does not move like its neighbours.

Horn-Schunck algorithm assumes smoothness in the flow over the whole image. Pixels should move together as much as possible, and after introducing a constraint over the flow in the image, we compute the solutions for v and u that satisfied the more constrained equation. The flow of pixels is formulated as a global energy functional,

$$E = \int\int \left[(I_x u + I_y v + I_t)^2 + \alpha^2(\nabla u^2 + \nabla v^2)\right] \mathrm{d}x\mathrm{d}y$$

in which the second term is the regularisation term, and α controls the smoothness.

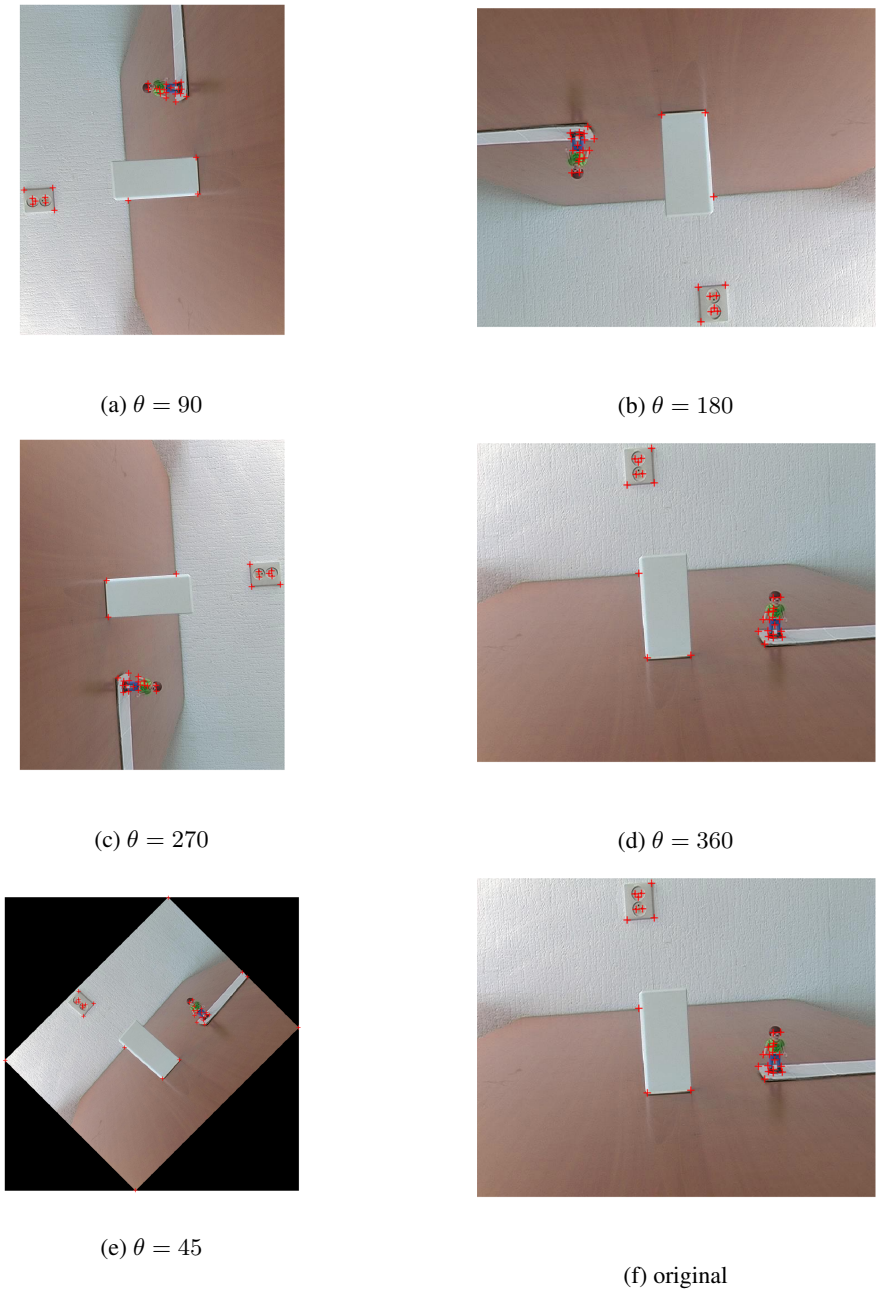
Behaviour on flat regions: *Lucas-Kanade* doesn't perform as well on flat regions. The reason is that for those regions, the derivatives will be zero, thus the inverse of $\mathrm{A}^T\mathrm{A}$ may not exist. Another case in for which $\mathrm{A}^T\mathrm{A}$ un-inversable is when x- and y-derivatives are linearly correlated(i.e. lines). Is is these derivatives that are equal to zero that causes the aperture problem mentioned earlier. In other words, for lines the motions is definite along the normal, ambiguous along the tangent and for flat regions, the motion is totally ambiguous.

Horn-Schunck algorithm performs relatively better compared to Lucas-Kanade method on flat regions because the flow information missing from homogeneous objects in filled in with information from the motion boundaries. One downside, however, is the fact that the method is more sensitive to noise than other local methods.

4 Feature Tracking

Question 1

For this part of the assignment we combine the first two algorithms implemented for the previous parts. Using the Harris Corner Detector algorithm, we detect the local feature points for the first



(a) $\theta = 90$

(b) $\theta = 180$

(c) $\theta = 270$

(d) $\theta = 360$

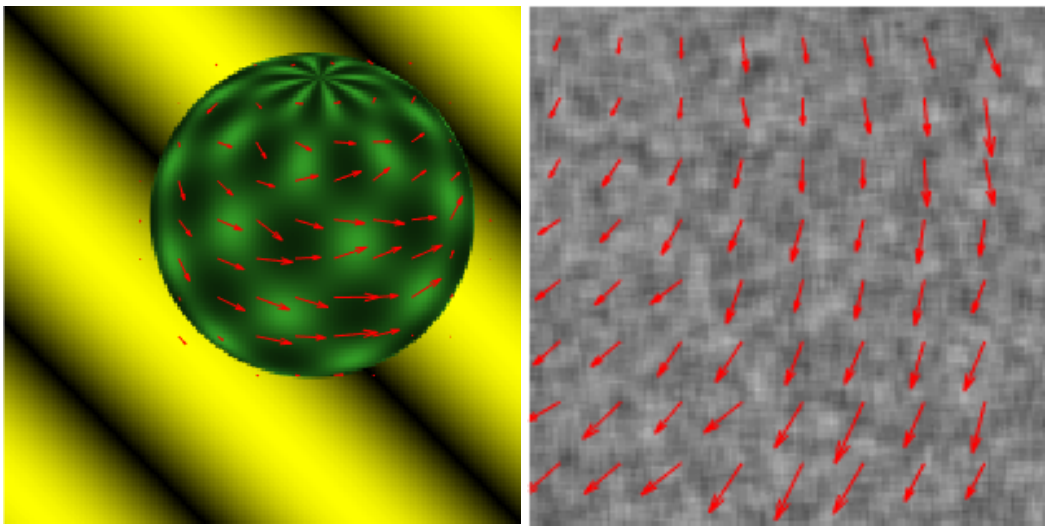
(e) $\theta = 45$

(f) original

Figure 5: Harris Corner Detector rotation invariance

frame of each video. Then, using the Lucas-Kanade algorithm we compute optical flow. For each frame, we compute the optical flow of the given interest point, updating the points by adding the displacement vectors to the x and y components. When we iterate through all frames, we see see that the interest points on moving objects remain roughly in the same position, relatively to the object.

We ran into some impediments caused by some consecutive frames in which the movement of the pixels were larger than what it is assumed when by the Lucas-Kanade algorithm. In the *ping-pong* data set, the transition of the white ball's position from one frame to another is not smooth. The ball moves too fast for the camera to capture a more continuous moves from one frame to another on a larger distan To account for this, we first scale down the frames to a much smaller size. In effect, what this downsizing does, is creating pairs of frames which better satisfy the assumption that the



(a) Optical flow for *sphere1.ppm* and *sphere2.ppm* (b) Optical flow for *synth1.pgm* and *synth2.pgm*

Figure 6: Optical flow using Lucas-Kanade method

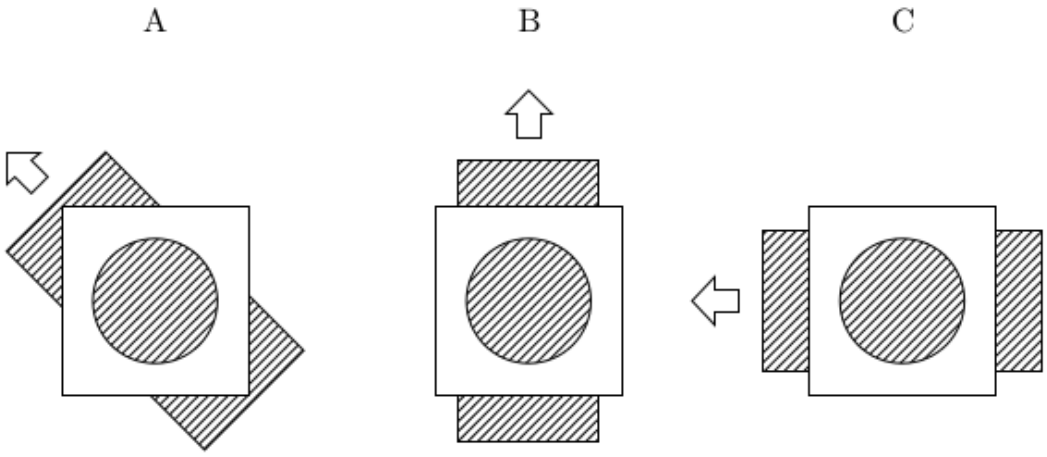


Figure 7: Aperture problem: if we look only at the local data(i.e. the round hole in the white square) we can not tell which of the three motions takes place.

movement of the pixels between them is small. We set a height of 200 pixels, and adjusted the width accordingly to keep the aspect ratio. After this downscale, fewer corners were detected at first using a threshold of 1000, but decreasing this threshold to 100 allowed for proper detection even in a down scaled image.

Question 2

For a sequence of frames, computing the optical flow for some already given interest points is more efficient than recomputing the interest points at each frame. Thus, computing the corners only once in the beginning and then tracing them across the frames is a approach worth taking into consideration. In addition to that, knowing for the interest points their position in one image and their predicted position in a second image, makes it easier to match similar interest points in two images(frames) of a video.

5 Conclusion

By working at this assignment and experimenting freely, using the techniques learned during the lectures(i.e. re-scaling, applying different types of smoothing, etc.), but also doing research on different algorithms(Horn-Schunck and The Shi and Tomasi), helped us to gain more insight into how to approach tasks similar to the ones given in the assignment. We also learned many of the reasoning behind the current practices in the field.

Of course, some of our results were not as good as the already implemented state of the art methods, but we believe that further tuning the parameters may play a role in improving our results.

References

[1] R. Szeliski, *Notes on the Harris Detector*. Springer, 2018. [Online]. Available: https://courses.cs.washington.edu/courses/cse576/06sp/notes/HarrisDetector.pdf

[2] U. Sinha, "The shi-tomasi corner detector." [Online]. Available: http://aishack.in/tutorials/shitomasi-corner-detector/