# Computer Vision 1

assignment 3


| Michiel Bonnee | Axel Bremer | Ruben van Erkelens | Tim de Haan |
|---|---|---|---|
| 11883936 | 11023325 | 12065064 | 11029668 |


## Introduction

This computer vision lab ends with a feature tracking program. This is implemented by determining the optical flow of specific points, that is the relative motion of objects, this is implemented using the Lucas-Kanade method. The specific points needed for the optical flow are found with a Harris Corner detector, which as the name already says, finds corners in an image. Therefore, this lab is divided into the three necessary parts to build a feature tracking program: firstly the Harris Corner Detector, then the Lucas-Kanade method and lastly the feature tracking program itself.

## 1 Harris Corner Detector

### Question - 1

1. ✓
2. Images 1, 2, 3 and 4 show the results of the Harris Corner Detector, fixed values for $\sigma$ (=4) and neighbourhood size (=25) were used. When experimenting with the threshold value we immediately see that the corner sensitivity increases when a low threshold is used. A lot of points that are not corners are now detected as corners. When increasing the threshold the sensitivity decreases. We observe that the sensitivity has to be balanced according to the desired end result. If we would want to detect all corners and do not really care if we also obtain noise, then we would want to use a low threshold. In contrast, if we find it more important to obtain only corners and really do not want noise in our final result, then we would use a higher threshold.

   Furthermore, we observed that the neighborhood size is especially important due to the algorithm not being invariant to scale. When the neighbourhood size is too small it would find way to many corners in a place that actually only corresponds to one corner. But if we then make the neighbourhood too big it would not find all corners for smaller objects in the image because they are too close together. Sigma is used to smooth the image, such that we capture less noise. However, out of all the parameters, we discovered that the threshold value has the most impact on the corner detection.
3. The Harris Corner Detector is rotation invariant. When you rotate a set of points in 2D space the eigenvalues remain the same. Although the orientation of the edges change, the shape (thus the eigenvalues) remain the same. If the eigenvalues are the same, the cornerness, which is based on the eigenvalues should also remain the same. The invariance can be seen in image 5. However in our implementation we've found that in the image rotated by 45 degrees some extra corners were located. This is due to the fact that the rotation function used (MATLABs imrotate) the pixel values differ slightly. In image 6 the armpit of the toy person can be seen, the circled area is the same area in the picture, in the 45 degrees angled image a tiny corner is present where in the original image it is not.

**Question - 2**

The difference between the Harris Corner Detector and the Shi-Tomasi detector is that the cornerness of a point is not defined by $H = \lambda_1\lambda_2 - 0.04(\lambda_1 + \lambda_2)^2$ but by $R = min(\lambda_1, \lambda_2)$. In this equation we have to find the lambdas directly because we have to find the minimum of the two. Due to this we cannot use the determinant and trace and we have to do an eigenvalue decomposition.

The relative cornerness values of assigned by Shi and Tomasi for the following scenarios will be:

- **Both eigenvalues are near zero.** These values correspond to a 'flat' surface. The minimum value is near zero, therefore the cornerness is also near zero.
- **One eigenvalue is big and the other is near zero.** The minimum value is still near zero, so the algorithm would return a cornerness which is also near zero. Which is good because this represents an edge and not a corner.
- **Both eigenvalues are big.** The corner case, since both the eigenvalues are big the minimal value is also big and a big cornerness score is returned.

## 2 Optical Flow with Lucas-Kanade Algorithm

**Question - 1**

The results of the Lucas-Kanade script can be observed in image 7.

**Question - 2**

1. The Lucas-Kanade Algorithm is a sparse technique, meaning it only needs a subset of pixels from the original image to estimate optical flow. The Horn-Schunck Algorithm is much more dense, and thus needs the entire image to estimate optical flow. This means that Lucas-Kanade is a technique that works locally, and Horn-Schunck works globally.
2. The Lucas-Kanade algorithm will fail on flat surfaces. This is due to the fact that the gradient cannot be computed on flat regions, making them ambiguous (aperture problem). The Horn-Schunck algorithm introduces a smoothness term, which is a minimization constraint. This way the algorithm prefers solutions that contain more smoothness while minimizing distortions. Therefore, unlike Lucas-Kanade, the Horn-Schunck algorithm will be able to detect motion on flat surfaces.

## 3 Feature Tracking

**Question - 1**

The result of our feature tracking algorithm can be seen in the enclosed gifs. They show feature tracking for the ping pong and the person toy files.

**Question - 2**

It is computationally cheaper to track the points, instead of calculating them for each pair of images. Furthermore, there might be instances where we want to estimate the motion of an object during a larger sequence of frames, instead of only looking at the relative motion of two frames. In that case we can be more certain of the 'identity' of a certain point. We can be more sure we're looking at the same point as in the last frame.

## Conclusion

This lab implements a feature tracking program. First we needed specific points in an image, for this we used the Harris Corner Detector. We observed that the quality of the detection is dependent on the given parameters of the algorithm, which in turn depends on the image. For instance, images with lots of detail require a higher threshold than images with less detail. The next step was to implement the Lucas-Kanade method, which determines the motion of an object using the points from the Harris Corner detector. Lastly, we bundled the two previous parts to implement a feature tracking program. Although the program showed to be susceptible to noise, we observed that it was able to successfully detect the motion of the object of interest.

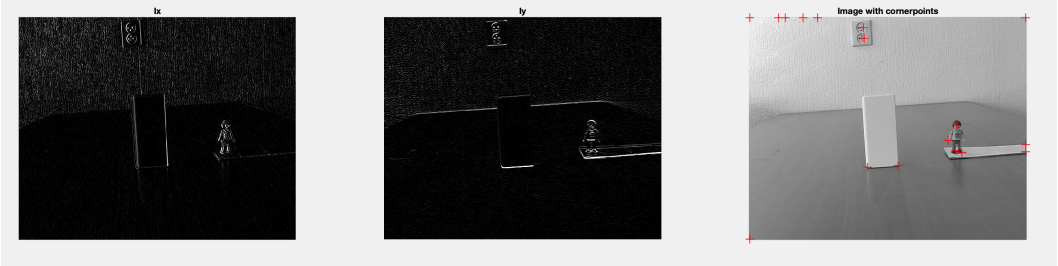


Figure 1: Image derivatives in x (left) and y (right) direction and original image with corner points, using a lower threshold value (0.2). $\sigma = 4$ and neighbourhood size equals 25.

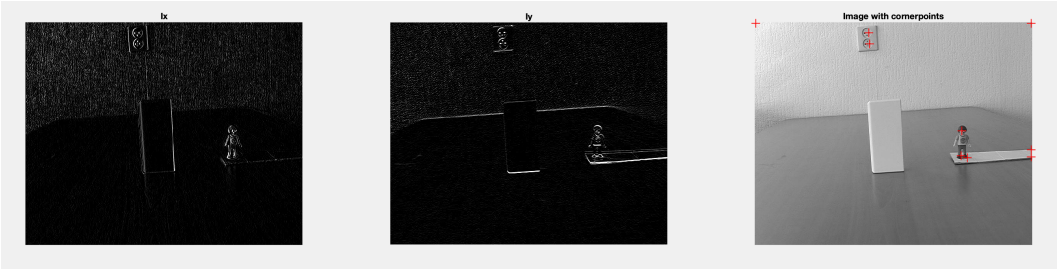


Figure 2: Image derivatives in x (left) and y (right) direction and original image with corner points, using a higher threshold value (0.5). $\sigma = 4$ and neighbourhood size equals 25.

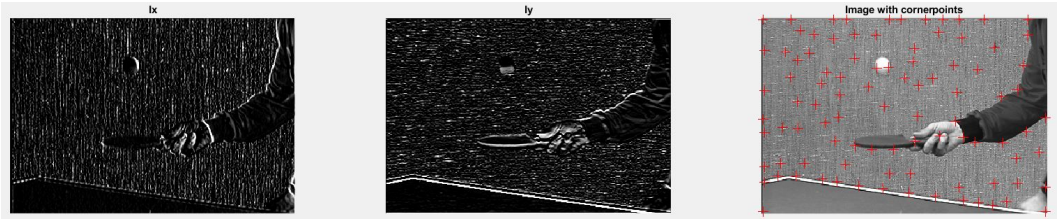


Figure 3: Image derivatives in x (left) and y (right) direction and original image with corner points, using a lower threshold value (0.015). $\sigma = 4$ and neighbourhood size equals 25.

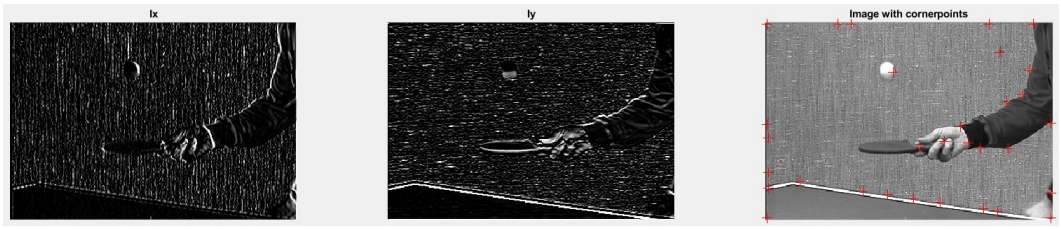


Figure 4: Image derivatives in x (left) and y (right) direction and original image with corner points, using a higher threshold value (0.085). $\sigma = 4$ and neighbourhood size equals 25.

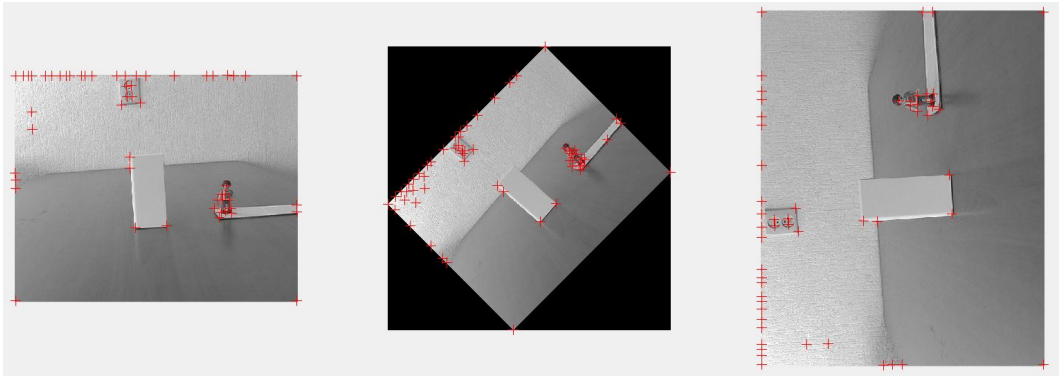

Figure 5: Cornerpoints found in image rotated by 45 and 90 degrees.

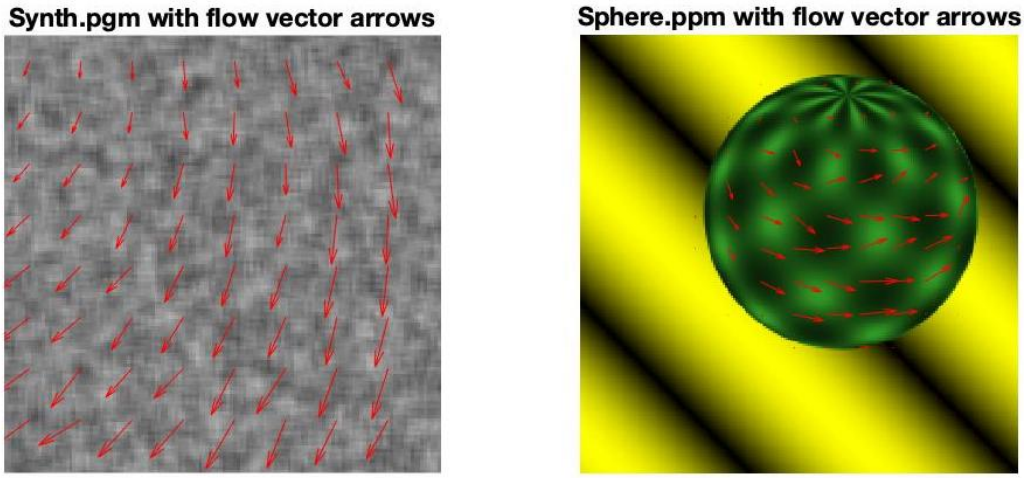


Figure 7: Optical flow vectors on top of the synth and sphere images.

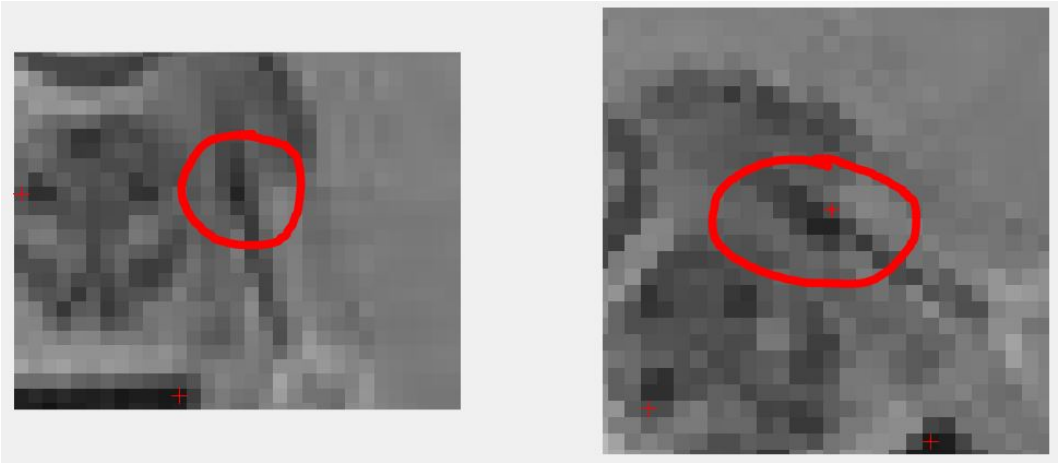

Figure 6: Left: original image, right: image rotated by 45 degrees. Circled parts are the same point in the image. Due to the differend pixel values created by rotation a new corner was found.