Lab 3 - Harris Corner Detector and Optical Flow

Jeroen Baars - 10686320

Bob van den Hoogen - 10420169

Introduction

This assignment includes the implementation of a Harris Corner Detector for images and optical flow. Both these methods have been combined to visualize feature tracking for two sets of images.

1 Harris Corner Detection

Question 1

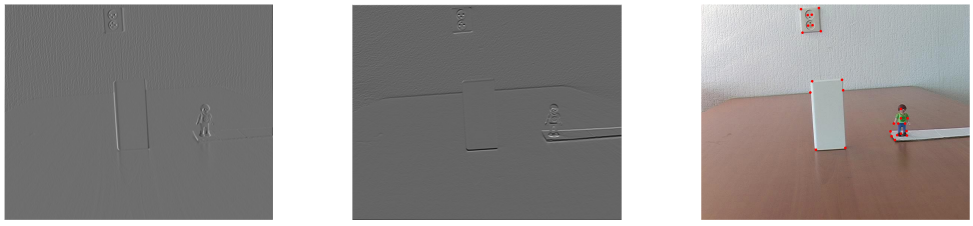
In this section a function for Harris Corner Detection is implemented. The function calculates the smoothed derivatives of the image by convolving the first order Gaussian derivatives with the image along the corresponding axis. These smoothed derivatives I_x and I_y are used for the calculation of matrix Q. Convolving ${I_x}^2$, ${I_y}^2$ and $I_X I_y$ with the Gaussian G gives the elements A, B and C for this matrix Q, with $Q = [A, B; B, C]$. Finally the *cornerness* for each points in the image is calculated in the matrix $H = (AC - B^2) - 0.04(A + C)^2$, where each point $H(x, y)$ gives the cornerness of the pixel (x, y) in the input image. A point is considered a corner point when $H(x, y)$ is the local maximum in a window of size $window\_size$ around said point and exceeds a given threshold value.

The function **harris_corner_detection.m** takes in the directory path to an image. In the parameters section there is a boolean flag to rotate the image with a random angle. The other parameters are the kernel size and sigma for the Gaussian filter, the window size for the local maximum evaluation and the threshold for the value $H(x, y)$.

The function is tested on two images. For the first image, *person_toy/00000001.jpg*, the following parameters are empirically found to be optimal: $kernel\_size = 5$, $sigma = 1.5$, $window\_size = 8$ and $threshold = 4\text{e}{-8}$ The results can be found in figure 1.

Figure 1: Harris Corner Detection person toy

(a) I_x (b) I_y (c) Corner points



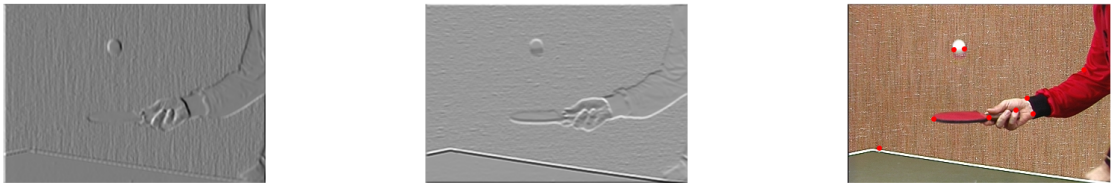
For the second image, *pingpong/0000.jpeg*, the following parameters are empirically found to be optimal: $kernel\_size = 5$, $sigma = 1.7$, $window\_size = 10$ and $threshold = 2\text{e}{-7}$

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

The results can be found in figure 2.

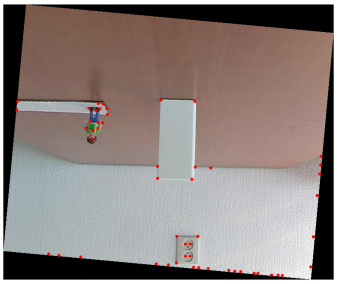
Figure 2: Harris Corner Detection pingpong

(a) I_x (b) I_y (c) Corner points



The angle does change the found corner points, albeit not drastically. This happens, since the window for detecting whether a point is a local maximum, is square window that doesn't rotate along with the image rotation. Hence, the window changes when the image rotates. This is verified with the *person_toy/00000001.jpg* image in figure 3.

Figure 3: Rotated Person Toy Image



Question 2.1

Shi and Tomasi (1994) developed a method complete based on Harris' corner detection method. However the *cornerness* was calculated directly as the minimum value of the eigenvalues of $Q(x, y)$: $H = \min(\lambda_1, \lambda_2)$

Question 2.2

These eigenvalues come from calculating the eigen-decomposition of the patches, not from the entire image. We want to evaluate for every point if it is a corner point based on the gradients surrounding the point within the given window. Therefore we look at the patches (with size *window_size*) for calculating the eigen-decomposition.

Question 2.3

Since for Shi and Tomasi (1994) only the smallest eigenvalue is of relevance, only a corner can be detected when both eigenvalues are big. Hence:

(a) Both eigenvalues are near 0 $\rightarrow$ relatively very low cornerness value assigned, no corner detected.
(b) One eigenvalue is near 0, one is big $\rightarrow$ relatively very low cornerness value assigned, no corner detected.
(c) Both eigenvalues are big $\rightarrow$ relatively high cornerness value assigned, possible corner detected.

2 Optical Flow with Lucas-Kanade Algorithm

Question 1

In this question the Lucas Kanade algorithm is implemented. This method assumes that the optical flow is essentially constant in a local neighborhood of the pixel under consideration. We will divide an image in section of 15x15 pixels and calculate the optical flow for each of these sections. In figure 4 and figure 5 is shown how this implementation worked out for these 2 figures. The Function lucas_kanadeṁ is used for this implementation, in this function the function LKṁ is used, this function takes as input or 2 images and a section size or as extra 2 list with x and y coordinates. With only the first 3 parameters as input the function returns a matrix containing every value for every section with can then be visualized with the visualizeṁ function. If there x and y coordinates lists are also given as input it calculates the corresponding values with these coordinates and plots them on the current figure. In figure 6 is shown what the influence of a 5x5 section is on the sphere image and how the influence of noise can possibly be reduced with smaller sections.

The following parameters are empirically found to be optimal: kernel size Ī5 and sigma Ī.5.

Figure 4: Lucas Kanade - sphere optical flow

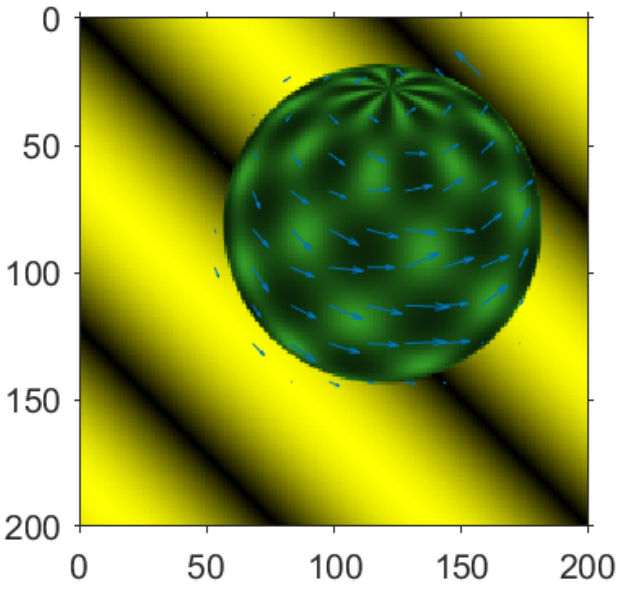


Figure 5: Lucas Kanade - synth optical flow

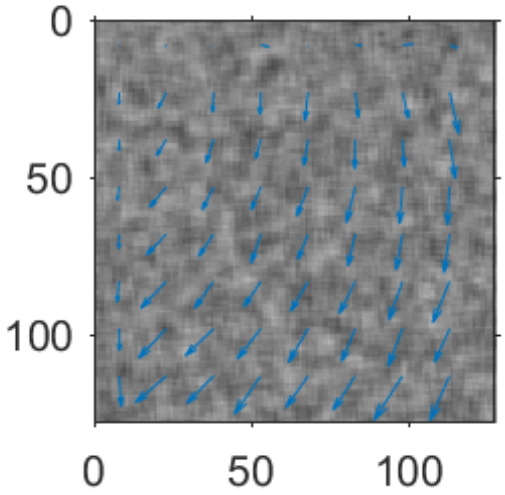
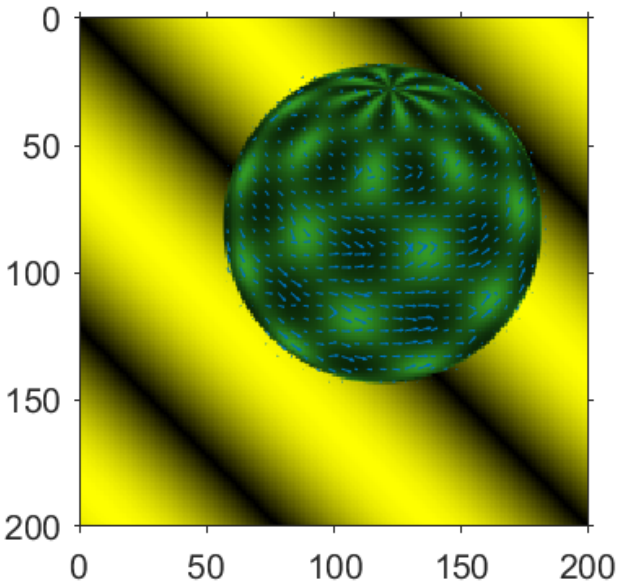


Figure 6: Lucas Kanade - sphere 5x5 optical flow



Question 2.1 Lucas Kanade - local, because this algorithm assumes that the flow is essentially constant in a local neighbourhood of the pixel under consideration. It solves the basic optical flow equations for all the pixels in that neighbourhood, by the least squares criterion.

Horn-Schunck method - The Horn-Schunck algorithm assumes smoothness in the flow over the whole image. The flow is formulated as a global energy functional which is then sought to be minimized.

Question 2.2 With lucas kanade a big flat region with the same brightness will have no movement in the middle of this region because the algorithm searches for a pixel with the same brightness and will find this on the same spot, it is another spot but the color values did not change. It will recognize movement in the corners of the region.

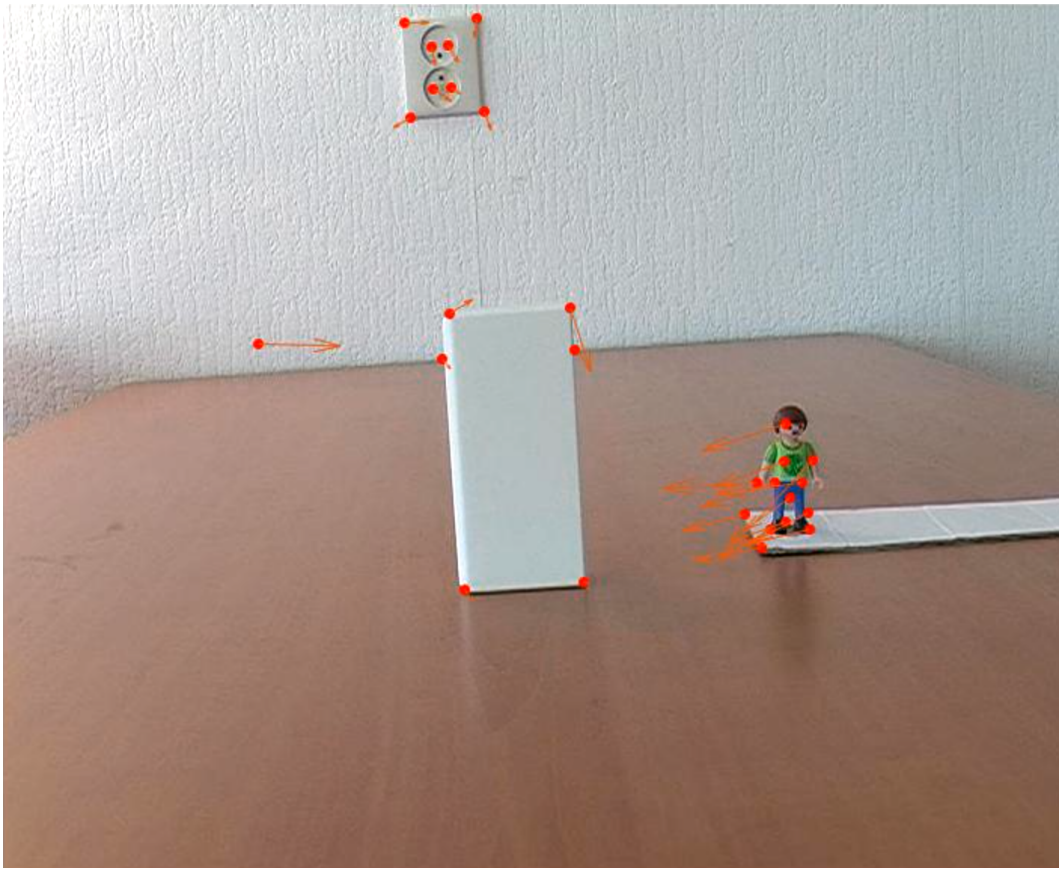
With Horn-Schunck method this problem is solved better because it looks at the global scale and will recognize the movement of the big flat region as a whole instead of focusing on local regions.

3 Feature Tracking

Question 1

In the zip file there are two .avi files included. In these video files you can see the feature tracking for both the *person_toy* and the *pingpong* image set. Figure 7 is a screenshot of the *person_toy/avi* file.

Figure 7: Screenshot from feature tracking



To make this video, the following steps have to be executed:
1. uncomment the part for for the image set you want the video made of.
2. in **harris_corner_detector.m** set the correct parameters.
3. run tracking.m

A .avi will be saved to the working directory.