Computer Vision Assignment 3

Frederik Harder
10986847
frederikharder@gmail.com

Maartje ter Hoeve
10190015
maartje.terhoeve@student.uva.nl

March 2016

1 Introduction

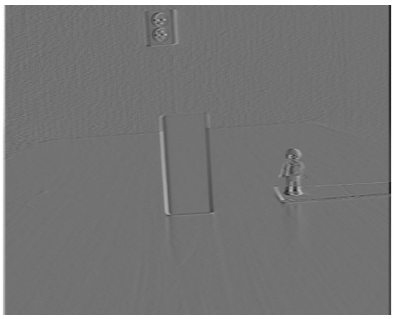
For this assignment the ultimate goal was to implement a tracking algorithm that uses Harris Corner Detection for the initial identification of interest points and Optical Flow to track these points over time.

2 Harris Corner Detector

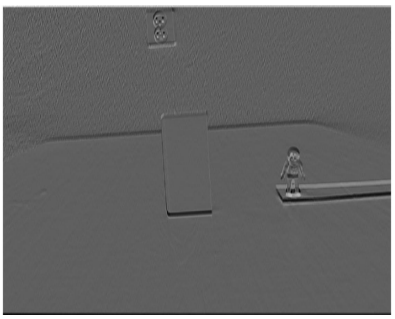
The first part of the assignment comprises the implementation of a Harris Corner Detector. For every pixel in the image one can compute a '*cornerness*'-score. Only the points with the highest '*cornerness*' values in their neighbourhood are considered to be corners. The full derivation that we used is given in the assignment itself. Here we present the some more details as to how we have implemented the Harris Corner detector in MATLAB, as well as some results.

The first step of the algorithm is to compute the Ix and Iy matrix for the image, i.e. the partial derivatives of the image in the x and y direction. For this we use a Gaussian derivative, taking a horizontal Gaussian filter for the x-direction and a vertical Gaussian filter for the y-direction. This implementation can be seen in the file `gaussianDer2.m`. The computed derivatives are shown in figure 1 and figure 2.

Next we compute the elements of the Q matrix, namely A, B and C, as described in the assignment. With these elements we can compute the H-matrix of '*cornerness*'-scores. From this matrix, corners are identified as local maxima within their neighbourhood (in our case, all pixels within a city block distance of 15), which exceed a threshold value, that we set to 0.01 unless otherwise specified.

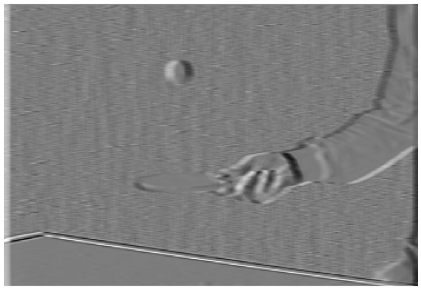


(a) Image derivatives Ix

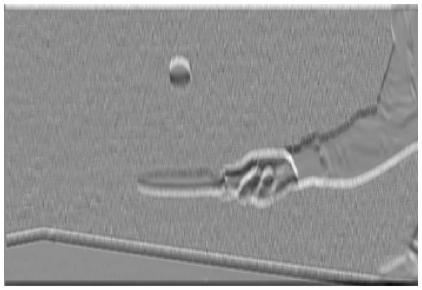


(b) Image derivatives Iy

Figure 1: Image derivatives Person Toy



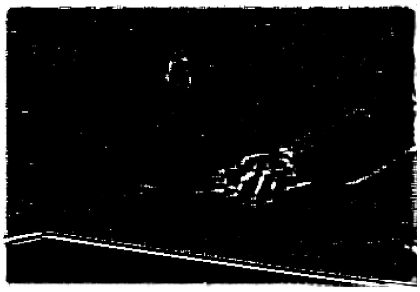
(a) Image derivatives Ix



(b) Image derivatives Iy

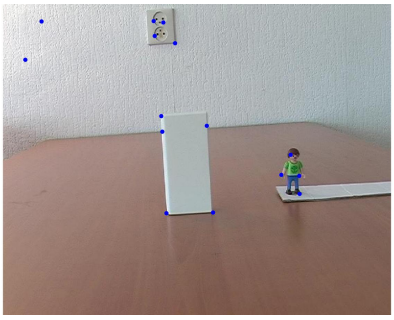
Figure 2: Image derivatives Ping-pong

(a) H-matrix Person Toy, $k = 0.04$

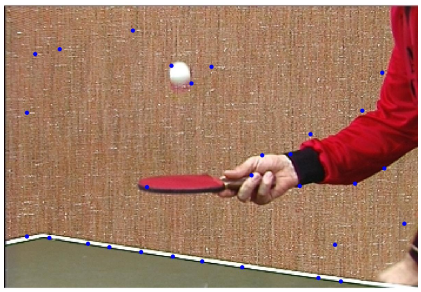


(b) H-matrix Pingpong, $k = 0.04$

Figure 3: H-matrices



(a) Corner plots person toy, threshold = 0.005



(b) Corner plots Pingpong, threshold = 0.005

Figure 4: Corner plots

To avoid finding corner points on the image border due to zero-padding, interest points at the border are not taken into account. Padding with values of the border pixels would have been an alternative. All corner points are returned and are plotted on top of the original image. The results for the person_toy as well as the ping-pong picture can be found in figure 4.

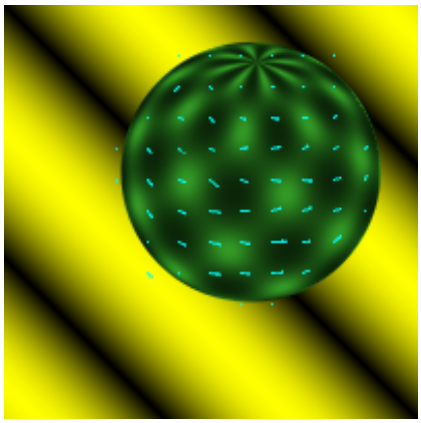
3 Optical Flow

In this part of the assignment we implement the Lucas-Kanade method for optical flow. Optical flow determines the motion of pixels from one frame or picture to the next. Again we use the algorithm as described in the assignment. First we compute the partial derivatives of the image in the x, y and t direction. For the gradients in the x- and y direction we use two relatively simple gradient filters: $[1; 0; -1]$ and $[1; 0; -1]^T$ respectively. This way of computing the gradients yields in better results than using a Gaussian derivative, as was used for the Harris Corner detection, especially once the objects in the images had to

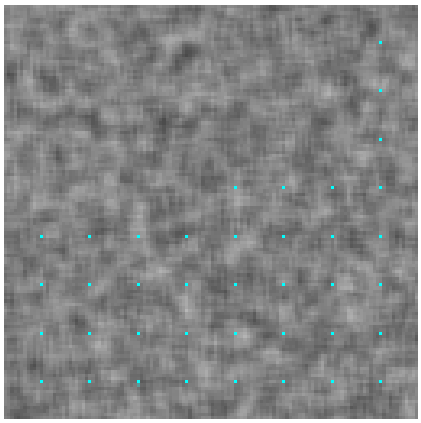
be tracked (see next part of the assignment). We compute the derivative in the t-direction by subtracting the pixel values of image 2 and image 1. Next we divide the derivatives of the image in the x and y direction into non-overlapping regions. The centers of these regions are the interest points for the optical flow. Now we construct the A matrix and b vector and compute the optical flow vectors. We plot the optical flow on the image by using MATLAB's built-in function `quiver`. The results can be seen in figure 5.

4 Tracking

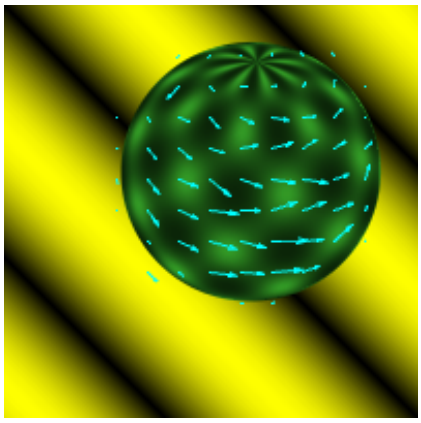
In this last part of the assignment we implement a tracking function. For a particular sequence of frames the movement of of interest points in this sequence is tracked. First we define the interest points by using our implementation of the Harris Corner Detector, i.e. the points we are going to track are corners in the image. Then we use our implementation optical flow to define the new positions of these corners in the sequential images. The only difference from the previous implementation of optical flow is that now we do not fully split the image in non-overlapping regions, but we define regions around the interest points. Three frames that show the optical flow and tracking are shown in figure 6 and 7, for both the person_toy frames as the ping-pong frames. Note that the arrows are are scaled up for better visibility and do not represent the real magnitudes of the vectors, which are much smaller. The first frames are taken from the person_toy. As can be seen the corners are tracked quite well, yet one interest point loses track half way through the tracking. Most likely this is because the colour of the face of the toy and the background are very similar at that point. In the ping-pong examples the player's right arm is tracked fairly well until it is occluded by the left arm reaching for the ball. The ball itself moves too far between frames and would likely require better temporal resolution to be tracked.



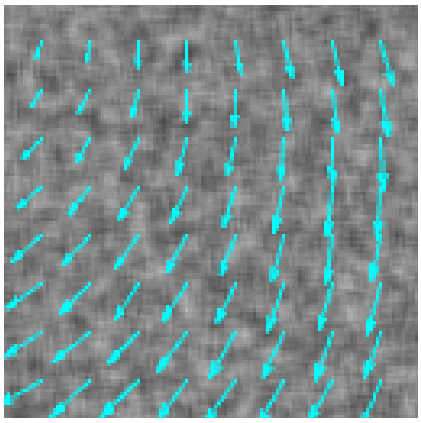
(a) Optical flow in *sphere* pictures with actual vector size



(b) Optical flow in *synth* pictures with actual vector size

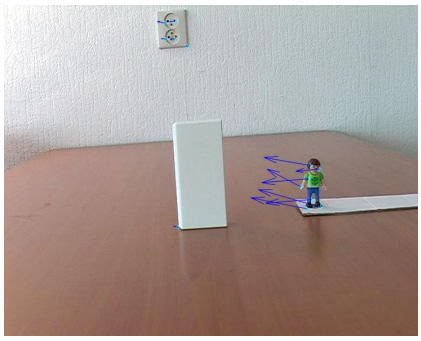


(c) Optical flow in *sphere* pictures with scaled up vectors

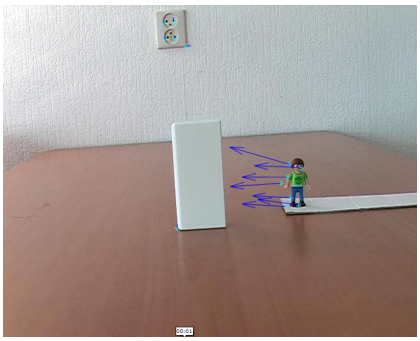


(d) Optical flow in *synth* pictures with scaled up vectors

Figure 5: Optical flow



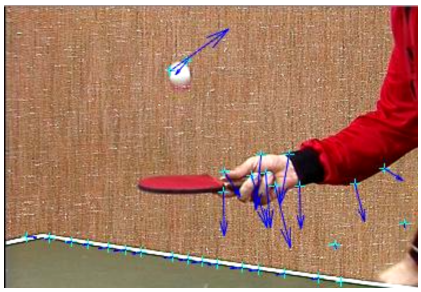
(a) First frame of the toy tracking movie



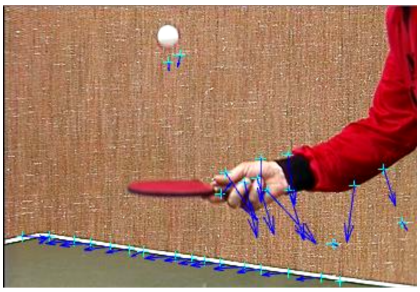
(b) Frame half way of the toy tracking movie

(c) Final frame of the toy tracking movie

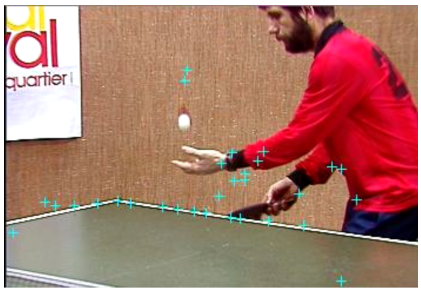
Figure 6: Three frames of the toy tracking movie



(a) First frame of the ping-pong tracking movie



(b) Frame half way of the ping-pong tracking movie



(c) Final frame of the ping-pong tracking movie

Figure 7: Three frames of the ping-pong tracking movie