Computer Vision Assignment 3

Kasper Bouwens Wolf Vos

March 2016

1 Harris Corner Detection

The Harris corner detection function computes the Ix and Iy matrixes by convoluting the image with the derivative of the gaussian kernel. The resulting image are shown below. Secondly a sliding window computes H by $(AC - B^2) - 0.04(A + C)^2$ where A and C are the sums of Ix^2 and Iy^2 within the window, B is the sum of Ix * Iy within the window. Next a second sliding window was used to go over the H matrix to find the local maxima. This was done by comparing the centre point of each window with all the other points in the window. These are the eventual corner points that were plotted on the original image. The sigma was set at 1.5. This value was found by experimenting with different values.

2 Lucas-Kanade Algorithm

The function divides the sphere image in regions of 20x20 pixels because the image size is 200x200. For the synth picture regions of 16x16 were chosen because the image size is 128x128. The images are convoluted in the x and y direction first with the derivative of the gaussian with respect to x and y respectively. Next, the derivative with respect to time is taken by subtracting the image at t=1 form the image at t=2. This gives us the difference between the pixel values. Within each window the A b matrices are calculated. By using the following formula: $v = (A^T A)^{-1} A^T b$ the optical flow can be calculated. Below are two images that demonstrate how this works.

3 Tracking

For the purpose of this assignment a separate Lucas-Kanade function was implemented that takes as arguments the images needed to compute the difference between two images at different time points and an array containing the interest points. This was done so that we could keep one function that divided the image into segment and another one that would use interest points to calculate windows. The tracking function used the first image to find interest points on

the image. The function then proceeded to send images to the Lucas-Kanade function in order to calculate the changes the interest points went trough on each frame. The Lucas-Kanade function proceeded to calculate windows of a specified size around the interest points. The change in the interest points was then added to the interest points and reinserted into the function at the next call of the function with the next two images. We found that it was hard to track the pingpong ball and therefore the resulting optical flow vector doesn't look like what we espected. The hand was tracked rather good compared to the pingpong ball.

Figure 1: kernel in X direction

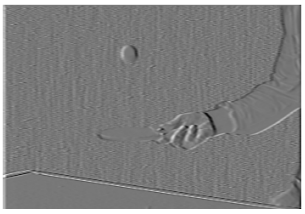
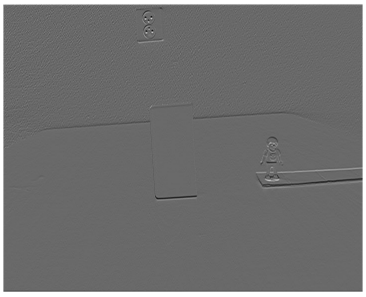


Figure 2: kernel in X direction

Figure 3: kernel in Y direction



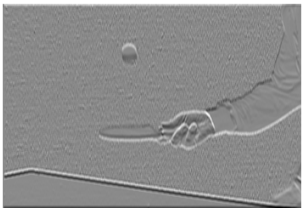
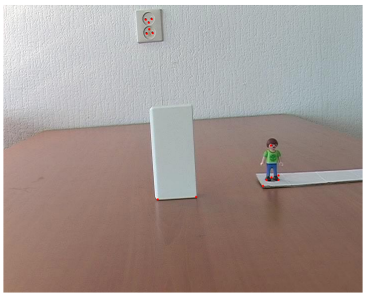


Figure 4: kernel in Y direction

Figure 5: cornerpoints



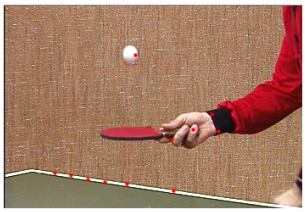


Figure 6: cornerpoints

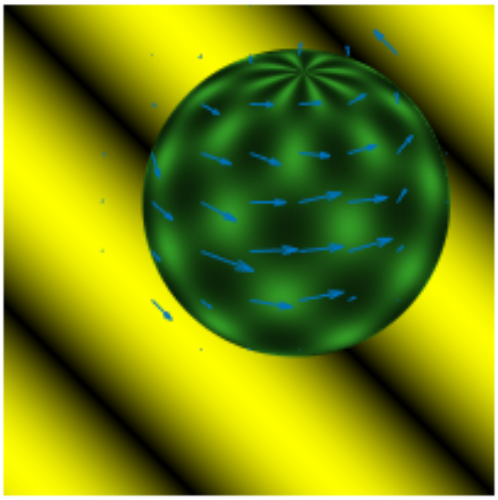


Figure 7: Optical Flow of the Sphere images

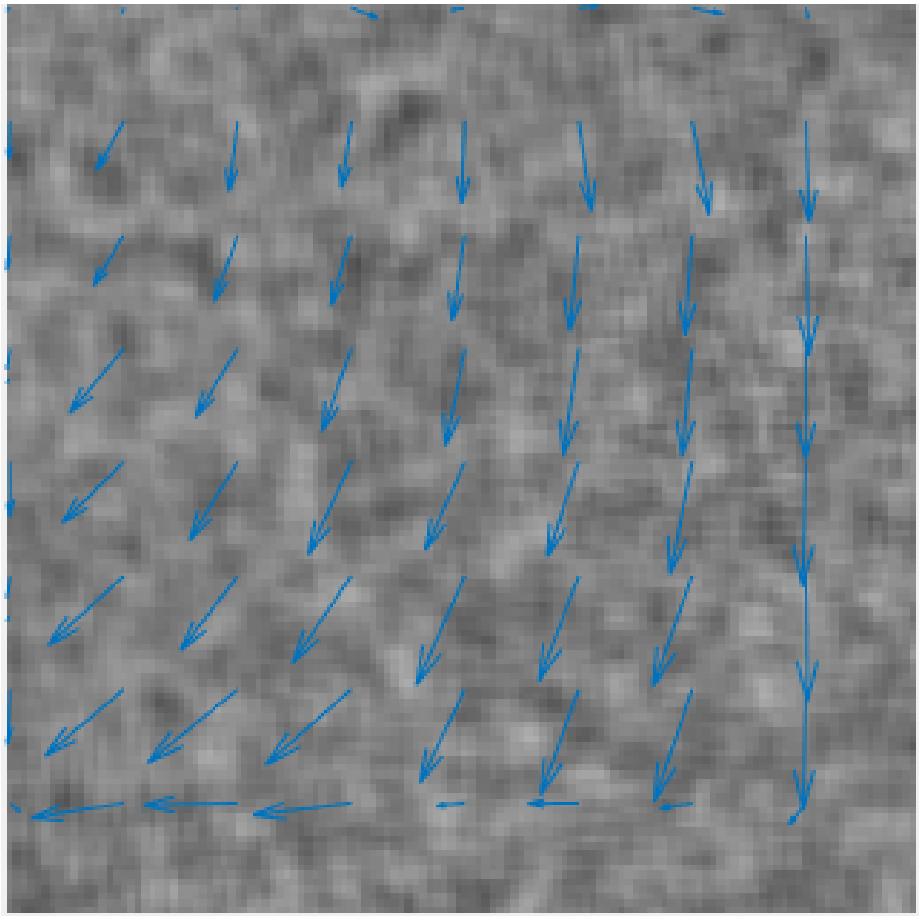


Figure 8: Optical Flow of the Synth images