Assignment 3 - Computer Vision

Tushar Nimbhorkar 11394110
Diede Rusticus 10909486

February 2017

Abstract

This weeks goal was to get an understanding of Harris Corner Detector and Tracking with Optical Flow. These three techniques were first implemented in Matlab and tested on different example images / video frames.

1 Harris Corner Detector

This function returns the *cornerness* matrix $H(x, y)$ of an image together with the resulting corners coordinates. This matrix H has the same dimensionality as the image. The algorithm knows five parameters which we can tune: region-size, sigma, kernel-size, corner-threshold, window-size. The steps we took:

1. First, we are going to compute Ix and Iy, the partial derivatives in the x and y direction of the image I. These give us two smoothed images of the different directions. They are obtained by convolving with the first order Gaussian derivative in the corresponding direction.

2. After that, the elements A, B and C of the auto-correlation function $c(x, y)$ are computed. A is obtained by squaring Ix and convolving it with a Gaussian filter. We use *convn* in matlab with the argument 'same' to get the result in the original size and gets rid of the padding.

3. Because the corner points are the local maxima of H we build a sliding window that goes over the image. For every region it evaluates the pixel with the highest cornesness value, and if this value is higher than the defined threshold, we define it as a corner. This threshold must be estimated through some experiments.

4. Two images were used to test the Harris Corner Detector we implemented. Figure 1 and 2 show the results.

Figure 1: Results of the Harris Corner Detector with region-size = 15, $\sigma = 3$, kernel-size = 17, threshold = 3000, window-size = 19

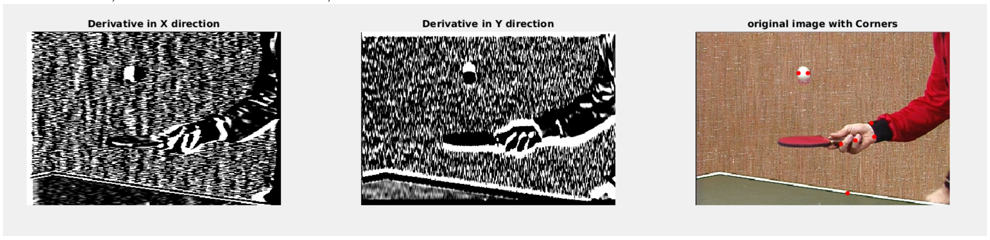
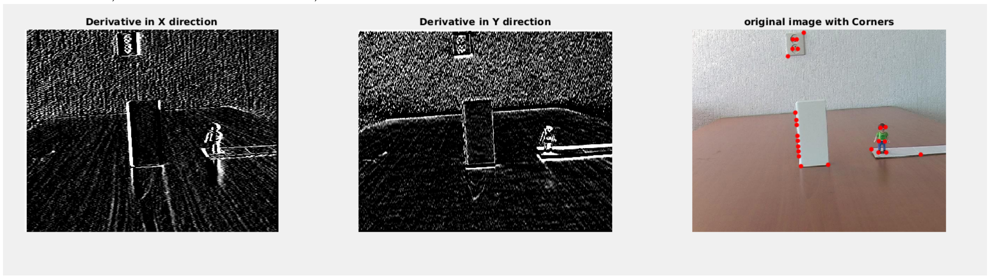

Derivative in X direction
Derivative in Y direction
original image with Corners

Figure 2: Results of the Harris Corner Detector with region-size = 15, σ = 3, kernel-size = 13, threshold = 200, window-size = 19


Derivative in X direction
Derivative in Y direction
original image with Corners

3 Optical Flow with Lucas-Kanade

We want to do tracking of moving objects through object motion or camera motion using optical flow. This optical flow is calculated with the Lucas-Kanade algorithm. The vertical (u) and horizontal (v) velocity is calculated for every pixel in a certain region. Therefore, we need partial derivatives ∂x, ∂y, ∂t, to calculate the next intensity level.

- One of the arguments the function requires is a matrix P containing the coordinates where the optical flow must be calculated. In the next section, we will use coordinates of the corners. For this section, we will use coordinates of the centroids of fixed regions. These regions do not overlap. This way, we will get one optical flow calculation per region.
- The other two arguments are the region-size and the two images. These two images should be the same but a slight change in movement/position. The region-size is set to 15 $\times$ 15.
- We calculate the partial derivatives ∂x, ∂y, ∂t by convolving the images with a 2D filter in the x direction (filter [-1 1; -1 1]), y direction (filter: [-1 -1; 1 1]) and difference of the pixel values of the two images, which will give us the gradient in t direction.
- Next, for every coordinate in P we calculate the optical flow velocities based on these partial derivatives
- The function returns $[\mathbf{u}, \mathbf{v}]$, which are the optical flow velocities in the x and y direction respectively.
- **u** and **v** are used for the function *quiver* to draw arrows on top of the original image, that represent the optical flows from the given points in P.
- See Figure 3 and 4 for the results of the optical flow estimations. The red arrows show the direction of the movements. As you can see in Figure 3, there are only arrows visible on the ball. However, optical flows do exist for the other points in the picture, but these had velocities of 0, so no arrow can be shown. In Figure 4 all points are moving, so arrows are shown all over the image.

Figure 3: Results of the Lucas-Kanade algorithm for optical flows. Region-size = 15, P = centroids of fixed regions

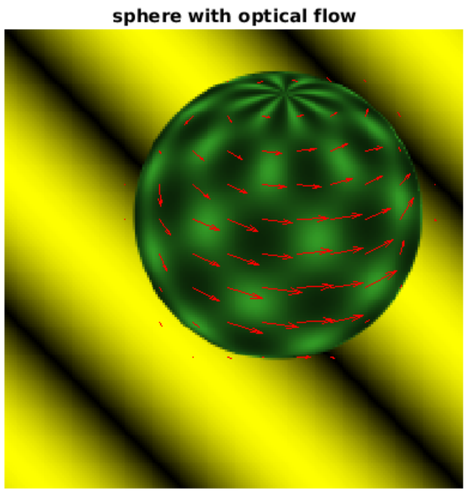
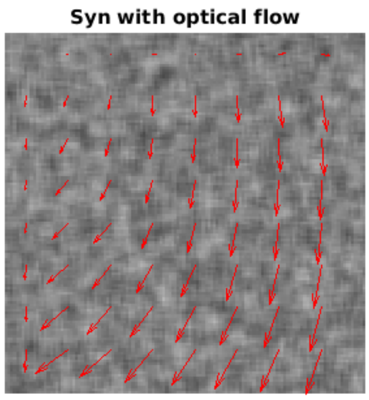


Figure 4: Results of the Lucas-Kanade algorithm for optical flows. Region-size = 15, P = centroids of fixed regions



4 Tracking

For this part we will use our implementation of Harris Corner Detection and Lucas-Kanade Optical Flows for tracking objects. Two examples are provided of objects moving: a toy person which is moved to the left and a person playing pingpong. They both consist out of several frames so when displayed sequentially, they form a movie. We are only using the corner detector for the first frame. After that, our goal is to update the location of the corners through the optical flow estimations. The results are shown in videos, contained in the zip file in the *visualisation* directory: *pingpong_vid.avi*, *toy_vid.avi*.

1. First we calculate the corners for the first frame using Harris-Corner detection method. So that we can track only these points.
2. Now having the corner points. Every corner point will become the centre of a window. Namely, these coordinates will be saved in matrix P and fed to the Lucas-Kanade function. We calculate the optical flow with Lucas-Kanade method on the window around the corner points.
3. Having calculated the optical flow vectors. We recalculate our corner points based on the velocity vectors. Just by multiplying the velocity with an optimised constant and adding the result to the previous corner coordinates.
4. Repeat step 2 and 3 until the last frame.

5 Deliverables

1. (a) The scores for corner is calculated as: $R = min(\lambda_1, \lambda_2)$. λ_1, λ_2 are the two eigen values which we also calculate in the Harris Corner Detector. However, here they only use the minimum . If R is greater than a certain predefined value, it can be marked as a corner.

   (b) For the Shi and Tomasi method, you only need to do eigen decomposition for the patches. Not for the entire image, but only the patch around the corner. Because you need to determine the minimum λ, you need the eigen values for them. For the Harris method, you do not need eigen decomposition because the product of $\lambda 1$ and $\lambda 2$ is equal to the determinant of matrix Q.

   (c)
   - When the threshold is lower than any of the lambda values, so also really small, it will consider the pixel as a corner. If higher than any of the lambda values, then considered as a non-corner.
   - If λ_1 is big and λ_2 is near zero, and at the same time the threshold is in between both values, then it will not consider the pixel as the corner. But if the threshold is lower than λ_2 (minimum of both lambdas) then it will consider it as a corner.
   - If both λ_1 and λ_2 are greater than the threshold, then the pixel will be considered as a corner.

2. (a) Horn-Schunck works at global scale while Lucas-Kanade at local scale.

   (b) Gradient will be zero on flat regions so Lucas-Kanade will fail. Horn-Schunck provides a reasonable estimate on flat regions. Namely, Horn-Schunck uses a diffusion equation, which creates good estimates of the dense field even if the local gradient is nearly zero. It smoothly fills in reasonable flow vectors based on nearby locations where flow is unambiguous. In other words, it avoids the problem that was encountered in Lucas-Kanade (flat regions have gradient zero).