UNIVERSITY OF AMSTERDAM

Assignment 3 Report

Luca Simonetto - 11413522
Heng Lin - 11392533
Computer Vision 1

March 8, 2017

Harris corner detector

This task requires the implementation of the Harris corner detector algorithm, using the gradient of the image and the calculation of the matrix H. Given a window of size $n \times n$ and a threshold t, we can identify the image corners, defined as the elements of H that have the maximum value compared to the $n^2 - 1$ elements in the window and that have a value above the threshold t. Results of the application of this algorithm can be seen from Figure 1 to 6, taking into account that the values for the parameters were:

- *kernel_size* 7
- *sigma* 1
- t 0.002
- *window_size* 7

We concentrated more on the tuning of t compared to the other parameters, as we discovered that a small change in its value has greater impact on the final result.

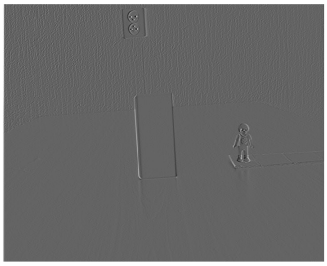

Figure 1: Gradient Ix

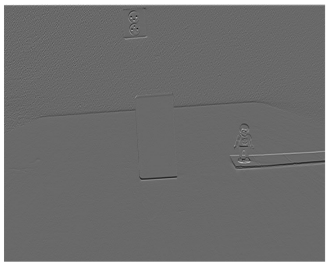

Figure 2: Gradient Iy

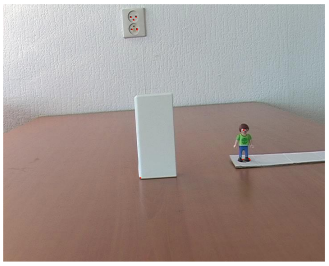

Figure 3: Detected corners

Given the Harris algorithm, the Shi-Tomasi corner detector brings the following changes:

- instead of calculating H as $\lambda_1\lambda_2 - k(\lambda 1 + \lambda 2)^2$, the new algorithm simply defines it as $min(\lambda_1, \lambda_2)$. This has the effect of defining a corner as a point where both lambdas have a value greater than t

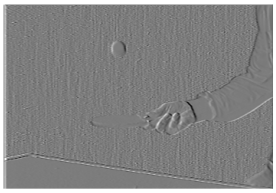

Figure 4: Gradient Ix

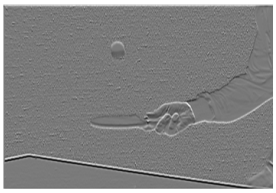

Figure 5: Gradient Iy

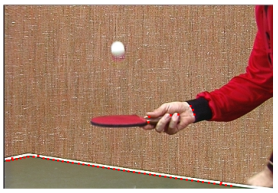

Figure 6: Detected corners

resulting in a simpler subdivision of the corner-edge space and better performance in real world cases. Figure 7 [1] and 8 [2] show the difference between the regions defined in the two algorithms.

- In this algorithm, contrary to the Harris corner detector, we are forced to calculate the eigenvalues in order to detect a corner, as the exact values of λ_1 and $\lambda 2$ are needed in order to check the minimum one against the threshold. Figure 8 visualizes this concept, where only the green area defines a corner.
- Given the previous definition of "corneress", we can distinguish the outcomes of the algorithm given the lambdas value:
  - Both eigenvalues are near 0: in this case no corner is detected.
  - One eigenvalue is big and the other is near zero: as the algorithm checks against the smallest value, we can conclude that there is no detection.
  - Both eigenvalues are big: a corner is detected.

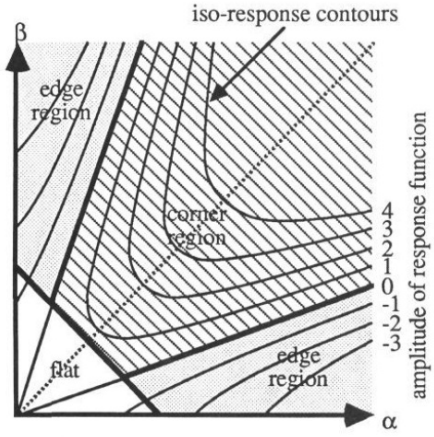

Figure 7: Harris region plot

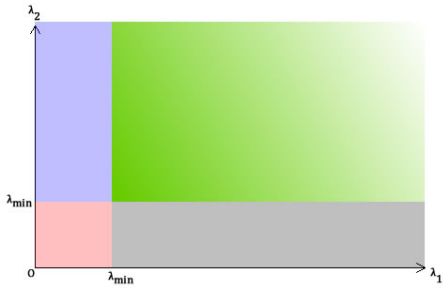

Figure 8: Shi region plot

Lucas-Kanade algorithm for optical flow

In Task3, we implemented Lucas-Kanade algorithm to estimate the optical flow of two images. In order to apply Lucas-Kanade algorithm, we made the following assumptions, 1) the flow is constant within the region around a pixel of interest, such that $I(x, y, t) \approx I(x + \delta_x, y + \delta_y, t + \delta_t)$, 2) limited to small motion between frames, such that the constraint can be approximated by Taylor series expansion, $I(x + \delta_x, y + \delta_y, t + \delta_t) = I(x, y, t) + \frac{\partial I}{\partial x}\delta_x + \frac{\partial I}{\partial y}\delta_y + \frac{\partial I}{\partial t}\delta_t + H.O.T.$.

[1] http://aishack.in/tutorials/windows-harris-corner-detector/
[2] http://aishack.in/tutorials/shitomasi-corner-detector/

The result of the optical flow obtained from the synthetic images can be seen in Figure 9, and the sphere images in Figure 10.

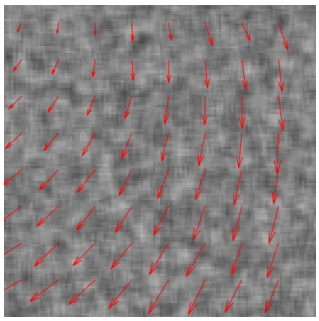

Figure 9: kernel size = 3

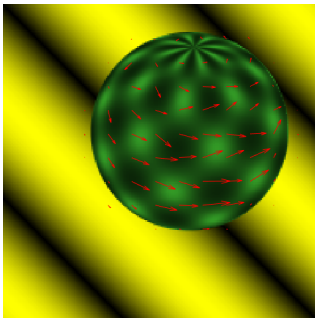

Figure 10: kernel size = 9

(a) At what scale those algorithms operate; i.e local or global?

Considering the brightness assumption in Lucas-Kanade method, only the regions around the pixel of interest is considered, hence Lucas-Kanade method is said to be operating in the local scale.

On the other hand, the Horn and Schunck method defines an energy function E that the algorithm tries to minimize, thus maximizing the global "smoothness" of the flow. The fact that the optimization problem operates on a global function, makes it to operate on global scale.

(b) How do they behave on flat regions?

Considering the equation $v = (A^T A)^{-1} A^T b$, in the case of Lucas-Kanade method, where A = $\begin{bmatrix} I_x(q_1) & I_y(q_1) \\ I_x(q_2) & I_y(q_2) \\ \vdots & \vdots \\ I_x(q_n) & I_y(q_n) \end{bmatrix}$, will be zeros if the image region is flat (when there is no difference among the pixels). This is called the "Aperture Problem", where the optical flow in a flat region cannot be determined. The same problem occurs on the lines where the ambiguity arises as the tangential velocity cannot be determined.

This problem is solved by incorporating the global smoothness constraint in the Horn and Schunck method. This is based on the assumption that the change of optical flow is smooth within an image, hence the equation penalizes when the adjacent pixels do not move accordingly.

Tracking algorithm

Using the above algorithms, we have rather simple tracking algorithm, that enables the recognition and track of the corners of a given image through a time series. First, given the first frame of a sequence, a number of corners are found using the Harris algorithm, and for every frame that follows, the algorithm tries to detect changes in the image that allows to compute the new position of the corners. This position update is made by determining the optical flow of the images using the Lucas-Kanade algorithm, in order to extract the change in velocity of the detected corners. Although decent, this algorithm tends to build up approximation errors as when a velocity vector doesn't shift a corner to a new pixel, its value is simply lost by the fact that the coordinates are rounded to the nearest integer.

This problem can be seen in the attached videos, as after around 60-70 frames, the various corners start to slowly shift out of position, without any chance to recover. In order to solve this issue we tried to modify the approach, by keeping a global value of the velocity for each corner, calculated as the sum of all the velocity

updates: when plotting the corners, instead of moving them from the last frame, we update their position by applying the global velocity to the initial corner. Although removing the problem with the rounding of the coordinates, this method resulted in much worse performance, where the corners start to shift after few frames, ending extremely far from the right positions.

The results of the final implementation of the algorithm, with coordinate update from the last velocity value, can be seen in the below Figures.

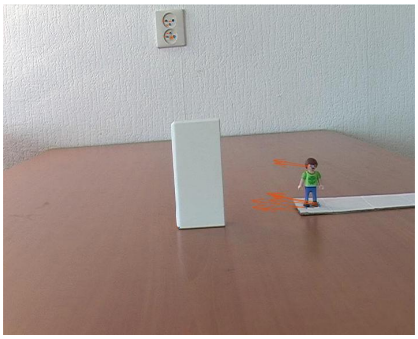

Figure 11: First frame

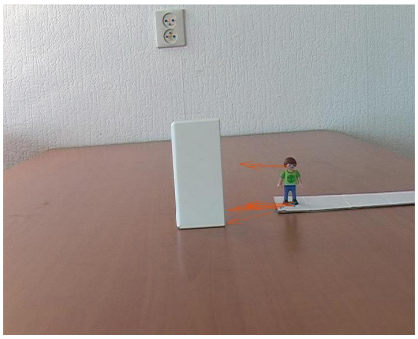

Figure 12: Middle frame

Figure 13: Lat frame

Figure 14: First frame

Figure 15: Middle frame

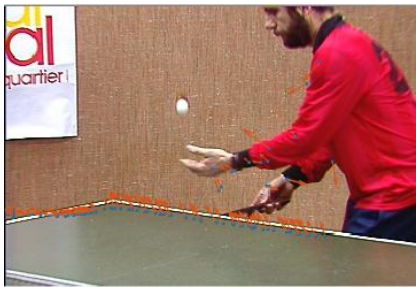

Figure 16: Lat frame