Computer Vision - Homework 3

Ioannis Gatopoulos `12141666`, Philipp Ollendorff `11734078`,
Konstantin Todorov `12402559`, Vincent Roest `10904816`

August 31, 2019

Introduction

This assignment focuses on extracting visual features from images, such as corners and edges. Firstly, an implementation of a Harris Corner Detector is presented, as well as its subsequent improvements through Shi and Tomasi's adaptations. Moreover, the assignment includes two methods of tracking these features over multiple image frames. Firstly, the definition of optical flow is introduced along with the difficulties of estimating it. One method of estimating optical flow is the Lucas-Kanade algorithm, which operates under the assumption of constant optical flow in a pixel's neighborhood. It implements a least squares solutions to the flow's differential equations of a pixel in a window (or patch). The algorithm shows some resemblances to that of the Harris corner detector, which suggest that corners and edges are good pixels to track. Finally, the Harris Corner Detector and Lucas-Kanade algorithm join forces as one feature tracking algorithm. The result of this assignment's code is a video showcasing the initial feature points (through the Harris detector) and the optical flow (through the Lucas-Kanade algorithm). Another potential application of optical flow, besides object tracking, is camera stabilization.

1 Harris Corner Detector

Question 1.1 & 1.2 On figures 1 and 2, are illustrated the raw pictures with the Harris corners [1] plotted on them, after the appropriate threshold was chosen (experimental tuning). We can justify this choice by looking at the images; although there is some noise, some spots that are detected as corners even though they are not (top left corner on the left figure of 1), the algorithm succeeds to capture the main/crucial corners (the corners around the plug, the toy boy and the corners around the box). These two figures are followed by their derivatives in x-direction (captures the vertical lines) and y-direction (captures the horizontal lines).

For our implementation, on both images we used the Sobel filters as an approximation of the first order Gaussian derivative and a window size 5×5 and $\sigma = 1$ when we convolve with the Gaussian. In terms of the threshold, we used a 14×10^7 and 6×10^9. Let us note that this numbers are high because we do not the build in function `gradient(image)`, but instead we convolve the image with the Sobel.

Figure 1: Original image with the corner points plotted on it (left), followed by the image derivatives I_x (center) and I_y (right).

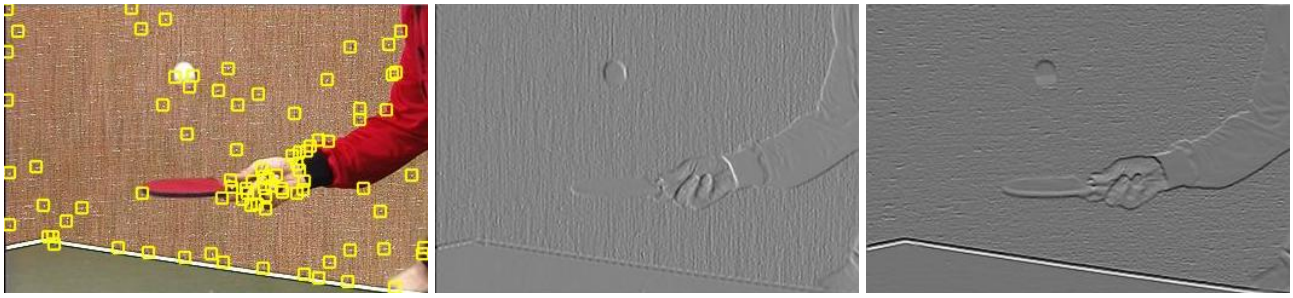


Figure 2: Original image with the corner points plotted on it (left), followed by the image derivatives I_x (center) and I_y (right).

Controversially, bellow on figure 3, there are the images with different thresholds, ranging from lower (left) to bigger (right). At the first case, most of the noise is considered as corner and at the latter many of the important corners failed to be detected. We can see that as the threshold get bigger, the algorithm becomes less and less flexible in detecting corners.

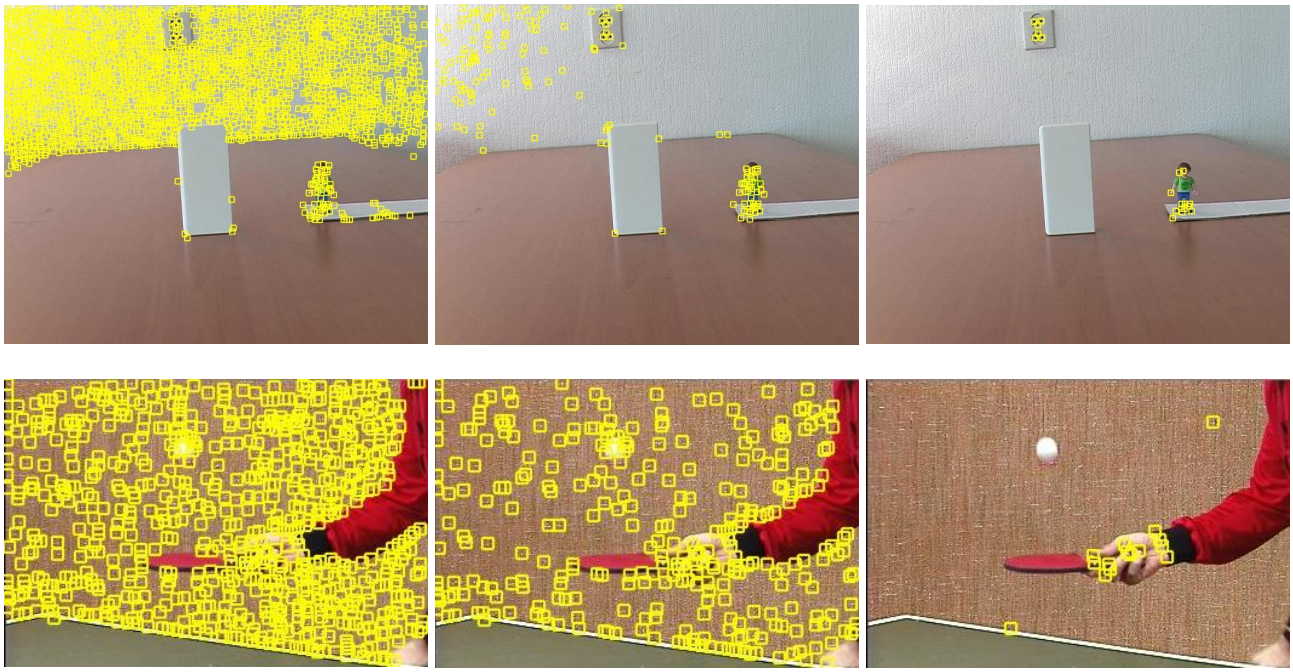


Figure 3: Harris corner detector with various thresholds; from smaller (left) to bigger (right).

Question 1.3 Figure 4 illustrates the results of Harris corner detector, after the image was rotated 45 degrees and 90 degrees. We can conclude that the Harris corner detector is rotation invariant, since the results are crucially corners of the image are detected. However, for example in left figure of the image, in 45 degrees, we can see that the algorithm detected a corner right above the box, which does not exist in the other, and some other right at the border of the picture with the black layout. The former is result of the fact that, when we rotate the image by 45 degrees, the pixel values changed and moved to different places, while the latter can be justified of the big change of the intensity (the gradient) between the image (grey) and the layout (black). However, regarding the 90 degree rotation, the corners that are detected are the same as the original image. In this case we do not have the previous problems.

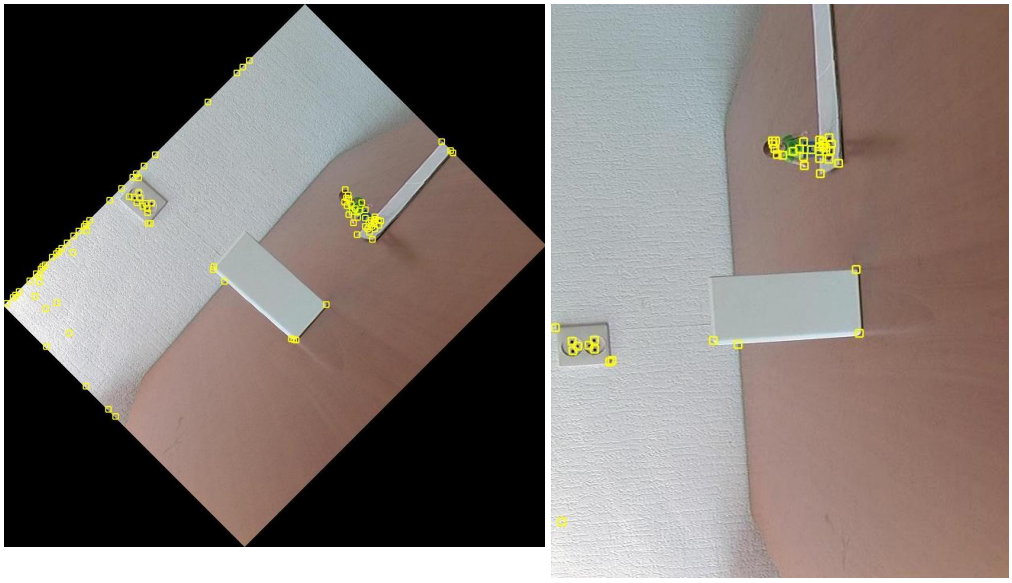


Figure 4: Image with the corner points plotted on it, after it was rotated 45 degrees (left) and 90 degrees (right).

Question 2.1 Shi and Tomassi's modification to the calculation of the corners is relatively simple. In the terms of equation 10, they define cornerness as $H(x, y) = min(\lambda_1, \lambda_2)$. That is, they replace the function manipulation by a simple selection criterion of the eigenvalues of the matrix Q. Other than this, the definition of cornerness is similar. This also means that the function H has now become a decision boundary with two straight lines ($x = \lambda_{min}$ & $y = \lambda_{min}$) in a two-dimensional plane (a division like a Mondriaan painting), as opposed to the decision boundaries created by the original Harris definition of $H(x, y)$ that have a slope. Figure 5 shows visually what these words are trying to explain:

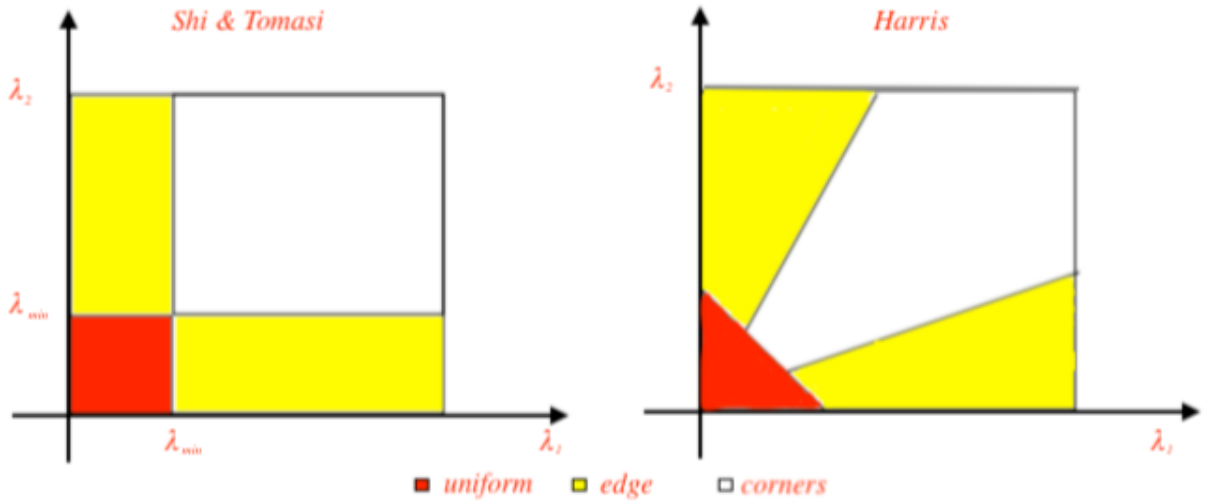


Figure 5: Differences between the cornerness H of both corner detection methods

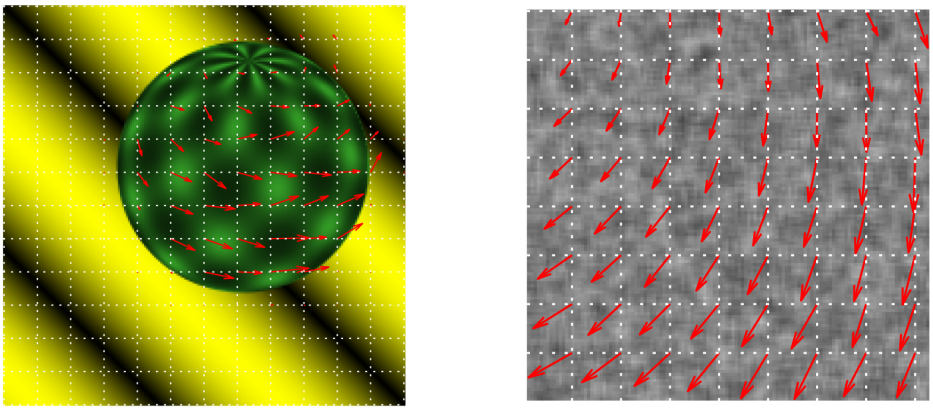
Question 2.2 From equation 10 can be concluded that for the original Harris definition, the of the matrix Q can be evaluated without eigenvalue decomposition. Instead, it suffices to compute $det(Q)$ and $trace(Q)$, by equation 11 & 12. This is computationally way cheaper than decomposing the image into its eigenvalues. However, the Shi-Tomassi variant relies directly on the eigenvalues in the image. More specifically, it relies on the eigenvalues of the matrix Q for each pixel in the image, the eigenvalues in the patch. Therefore, an eigen decomposition of the entire image would be uninformative as it does not yield any information on individual pixels. An example in which an eigen decomposition of an image could be useful is in image compression (PCA $\rightarrow$ dimensionality reduction).

Question 2.3

- If both eigenvalues are near 0, and given that they are below the thresholded value for H, which indicates the quality an edge or corner should have, these pixels fall in the category of non-corners and non-edges, or more simply put: they do not mark a corner or edge. They are part of a uniform pixel distribution in the image, which are the same in brightness as pixels in the local neighborhood. In parlance of the current exercise, these are areas with low gradient in both x and y directions. The value of $H(x, y)$ would therefore be "near zero". This region is indicated in Figure 5 by the dark region.
- In this case, the relative cornerness would actually still be "near zero", because the minimum is taken of the two eigenvalues. In this case though, the pixel would be part of an edge, with a sudden change in gradient in either the x or y direction. The hatched region in Figure 5 shows this case.
- The relative cornerness in this case is "big", because a minimum of two big values can still be considered "big". In this case though, the pixel can be qualified as a corner with both x and y gradients being "big". The white region in Figure 5 shows the relative cornerness for this scenario.

2 Optical Flow

Question 1 We implemented the Lucas-Kanade algorithm as outlined in the assignment: We divided the input images in regions of size 15x15, disregarding any remaining pixels at the corners. So the 200x200 image of a sphere was cut into 195x195. Now we compute A, b and the vectors v for each region. To display the vectors we print a grid of 15x15 regions on the original image and use quiver to plot each region's vector at the top right corner of each box. The results for both pairs of test images are shown below in Figure 6.



(a) Sphere (b) Synth

Figure 6: Optical flow vectors for pairs of images

Note that the arrows on the sphere image are a little noisier than the arrows in the grayscale image. This is due to the nature of rotation: The sphere rotates in place, thereby occluding parts on the top right side behind itself after rotation. As expected, there are no motion vectors on the black-yellow background of the image. The grayscale is rotated in its entirety/zoomed in and therefore easier to track for the Lucas-Kanade algorithm.

Question 2.1 & 2.2 In contrast to the patch-based Lucas-Kanade method, the Horn-Schunck method operates at a global scale. Where the Lucas-Kanade method computes a flow for each patch in the image, the Horn-Schunck defines a global energy function which ought to be minimized, which could be listed here, but Wikipedia does a better job in this sense than us ([Wikipedia contributors, 2019]). A major drawback of the Lucas-Kanade method appears in patches that have a "flat" region. Because of its purely local nature, flow is difficult to estimate because the "flat" area does not have any feature points which move. On the other hand, because of its global smoothness constraint, the Horn-Schunck method does not have this problem: it actually fills out the "missing" information in the "flat" regions that can be calculated from the actual boundaries of that object. If there is some movement in some part of the image, this method extrapolates the calculation of the flow on that region to all other parts of the image. In the Lucas-Kanade method, these boundaries might

lie outside a patch, which means they are not included in the calculation of the flow in that patch. Even if they were, they would not be carried over to other patches.

3 Feature Tracking

3.1 Design of the algorithm

We implemented the feature tracking by iterating over all images that we have for the current scene. We detect the feature points from the first frame using Harris Corner Detector. Then we start "tracking" the points using Lucas-Kanade algorithm. For each pair of two images one after another, we estimate the optical flow and calculate the new feature points by adding it to the results from the previous iteration. This way we keep tracking them throughout the whole video.

In the end, we have all images with the feature points applied on top of them so we put them together in a video.

You can see our results on Figure 7 and Figure 8, noticing how the yellow squares representing our feature points move along for the different frames.

Figure 7: Ping-pong feature tracking

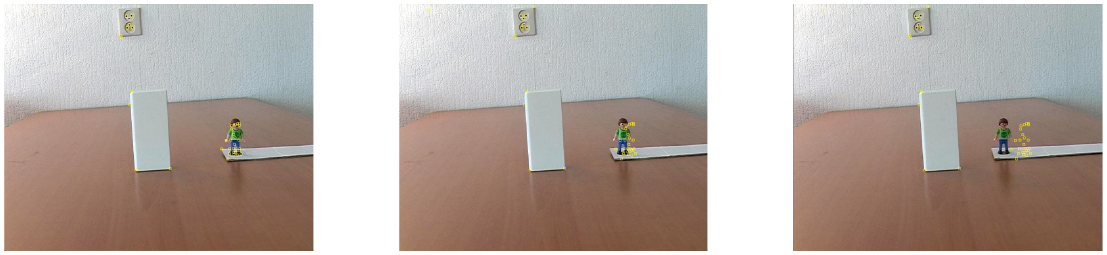


Figure 8: Person toy feature tracking

3.2 Why do we need feature tracking even though we can detect features for each and every frame?

We noticed that there is significant difference between calculating feature points using Harris Corner Detector and the Lucas-Kanade algorithm in terms of speed and complexity. If we were to detect the points for every single frame, this would become too expensive. Therefore, using the feature tracking we can easily follow the points, approximate their new positions which in most cases is good enough for the problem at hand.

Moreover, if we detect the images all over again, we will lose all the information that we gained in the previous frames. This might not be a problem in some cases, but in most we want to keep track of things we observed before and how they progressed throughout the history.

Another approach would be to "correct" the calculations after some amount of processed images by re-calculating the feature points using Harris Corner Detector again.

Conclusion

Ultimately, the merger of the Harris Corner Detector and the Lucas-Kanade algorithm yielded an effective way of tracking movement in image sequences. The Harris Corner Detector required some manual tuning (σ, filter size and threshold), but with adequate settings it was very effective in finding corner points in the example images. A major advantage of this method is its insensitivity to rotation, which is also useful in tracking these edges and corners in motion. Although an improvement was suggested in the form of the Shi Tomasi definition of cornerness, the combination of the Harris Detector and the Lucas-Kanade algorithm was more than sufficient for this application. The Lucas-Kanade algorithm was also contrasted with the "global" Horn-Schunck method. The final result uses the Harris Corner Detector to detect feature points, which were tracked in subsequent frames to yield a animated video showing the optical flow of these points.

References

[Wikipedia contributors, 2019] Wikipedia contributors (2019). Horn-schunck method — Wikipedia, the free encyclopedia. [Online; accessed 05-March-2019].

[1] A COMBINED CORNER AND EDGE DETECTOR, Chris Harris Mike Stephens (link)