UNIVERSITY OF AMSTERDAM

Assignment 3

Harris Corner Detector & Optical Flow

October 5, 2021

Students:
Sameer Ambekar
UvAnetID 13616064

Apostolos Panagiotopoulos
UvAnetID 14021900

Tutor:
Dr. Shaodi You

Group:
12

Course:
Computer Vision 1

1 Introduction

In this assignment we build a simple object tracking system, which consists of a feature detector and a feature tracker. First, we implement these building blocks, namely the Harris Corner Detector [1] for feature detection and the Lucas and Kanade Optical Flow algorithm [2] for feature tracking. We study and experimentally test their properties, and compare them with competing approaches. Finally, we combine them to perform object tracking.

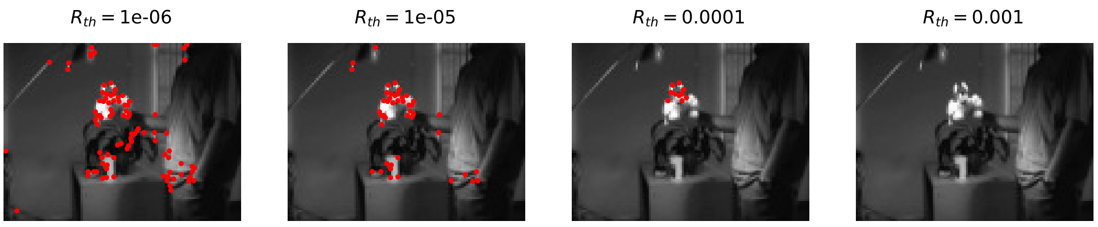
2 Harris Corner Detector

The Harris Corner Detector is an algorithm for feature or interest point detection. Feature points are image patches that contain highly discriminative information, which can be used for various applications such as object tracking and image stitching. In this section, we implement the Harris Corner Detector and experiment with its parameters and properties. Finally, we compare it against the Shi-Tomasi Corner Detector.

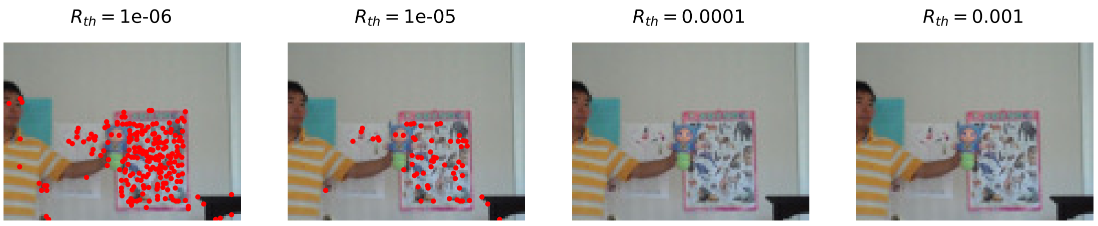
Question 1. Our implementation of the Harris Corner Detector can be found in the script `harris_corner_detector.py`. We mostly follow the original paper by Harris and Stephens [1], but we use a Gaussian filter to smooth the derivatives I_x and I_y (often mentioned in the literature as impHarris). Following [3] we use $\sigma = 1$ for the Gaussian filters that smooth the derivatives and $\sigma = 2$ for the Gaussian window that averages the Sum of Squared Differences (SSD). We also replace the original parameter $k = 0.06$ with $k = 0.04$, to compute the cornerness matrix, and use a 5x5 window to detect local maxima. Finally, for the cornerness threshold R_{th}, we follow the suggestion from [3] and use 1% of the maximum cornerness in the image. We present the corner points detected by our implementation, as well as the smoothed image derivatives I_x and I_y, in Figure 2.

We now proceed to tune the threshold R_{th} based on visual inspection of the resulting corner points in images `toy/0001.jpg` and `doll/0200.jpg`. We try values ranging from 10^{-6} to 10^{-3} on a logarithmic scale and present results in Figure 1. It is evident that anything below 10^{-6} would produce too many corner points, some of which would be on edge or flat surfaces, and anything above 10^{-3} would result in no corner points. We deem that $R_{th} = 10^{-5}$ is the best choice, which coincides with our previous choice of 1% of the maximum cornerness.

One nice property of the Harris Corner Detector is that it is, in theory, translation and rotation invariant. This is due to the combination of two facts. First, the SSD remains unchanged if we

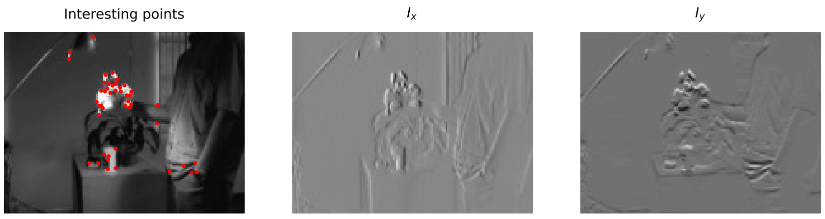


(a) `toy/0001.jpg`

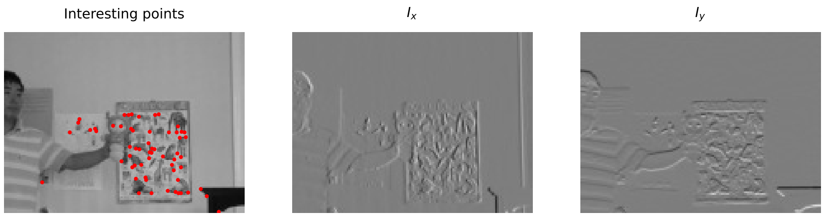


(b) `doll/0200.jpg`

Figure 1: Computing interesting points using the Harris Corner Algorithm for various thresholds.



(a) `toy/0001.jpg`

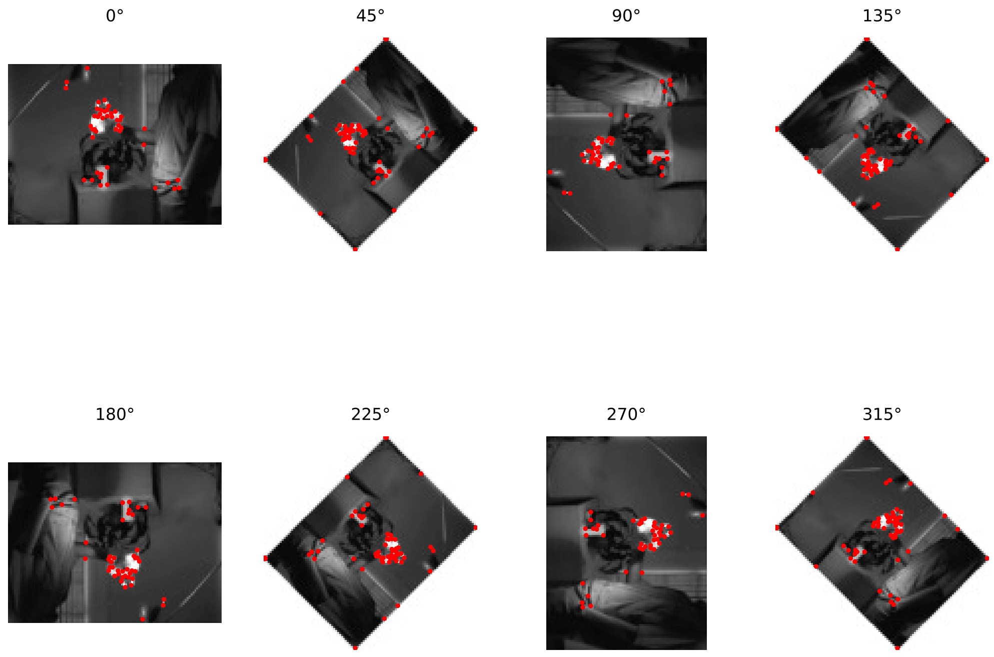


(b) `doll/0200.jpg`

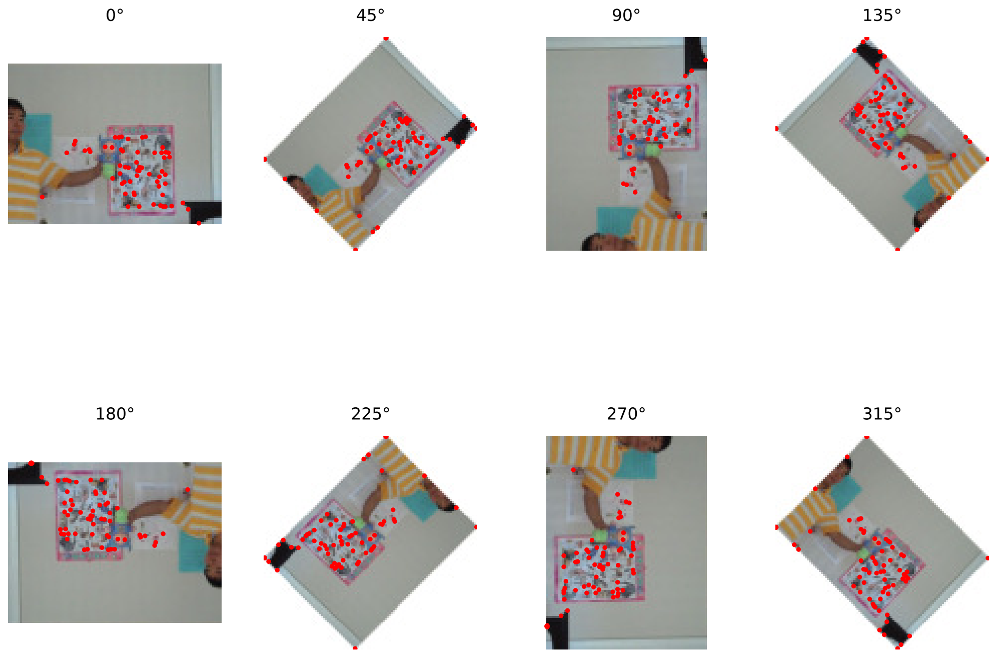
Figure 2: Interesting points relative to the image derivative I_x and I_y.

translate or rotate the image and perform the same transformation to the displacement vector $\Delta\mathbf{u}$. Second, when decomposing the SSD to $\Delta\mathbf{u}Q(\mathbf{u})\Delta\mathbf{u}^T$, the eigenvectors of $Q(\mathbf{u})$ correspond to the direction of intensity change and the eigenvalues λ_1 and λ_2 correspond to the scale of change in these directions. Hence, since we have the same SSD, in the case of translation the eigenvectors remain the same and in the case of rotation they follow the same rotation transformation. In both cases, however, the eigenvalues remain the same, resulting in the same cornerness value, and consequently in detecting the same corner points in the image. Note that this is attributed the use of a Gaussian weighting window, which preserves the SSD on rotations [5]. Experimentally, we verify this property using our implementation and showcase results in Figure 3. Due the the discretization when rotating the images and computing their derivatives I_x and I_y, the corner points could be placed a few pixels away from their theoretical position.

Question 2. There are many approaches to detect corners besides Harris. One of them is the Shi-



(a) `toy/0001.jpg`



(b) `doll/0200.jpg`

Figure 3: Computing interesting points using the Harris Corner Algorithm for various rotations of the input image.

Tomasi Corner Detector [4], which follows the same problem formulation as Harris', by analyzing the SSD as

$$\begin{bmatrix}\Delta x & \Delta y\end{bmatrix} Q(x,y) \begin{bmatrix}\Delta x \\ \Delta y\end{bmatrix}$$

where

$$Q(x,y) = \sum_W w(x,y) \begin{bmatrix} I_x(x,y)^2 & I_x(x,y)I_y(x,y) \\ I_x(x,y)I_y(x,y) & I_y(x,y)^2 \end{bmatrix}$$

However, they define cornerness as $H = \min(\lambda_1, \lambda_2)$, where λ_1, λ_2 are the eigenvalues of $Q(x,y)$.

Following a similar reasoning as in Question 1, the Shi-Tomasi Corner Detector is translation and rotation invariant but not scale invariant. That would require the standard deviation σ of the Gaussian window to be scaled accordingly, as well as the threshold R_{th} to be adapted.

From the above definition of cornerness it is obvious that if both eigenvalues are close to 0 or one eigenvalue is close to zero, then the relative cornerness will be close to zero, implying the examined pixel is not a corner point. On the other hand if both eigenvalues are big, the relative cornerness will also be big, indicating a corner point.

3 Optical Flow

Optical flow is a widely used technique for estimating the relative motion between a pair of consecutive images. In this section, we make use of the Lucas-Kanade algorithm, which makes two assumptions to calculate the optical flow. First, that the brightness is constant between the images and, second, that the displacement of the object between the two images is very small.

Question 1. Our implementation of the Lucas-Kanade algorithm can be found in the script `lucas_kanade.py`. We use our code to compute the optical flow between the `Car1.jpg` and `Car2.jpg` images, as well as the `Coke1.jpg` and `Coke2.jpg` images. We use Matplotlib's quiver plot to visualize the direction of motion and present our results in Figure 4.

By inspecting images `Car1.jpg` and `Car2.jpg` we observe that although the car moves, the camera undergoes a displacement, which results in the car being present dilated in the same spot and the background being moved. In Figure 4a we observe that this creates the false illusion that the car is stable while the environment moves. Moreover, due to the complex displacement of the camera, that is neither translational or rotational, they arrows pointing the direction of the background are inconsistent.

On the contrary, the `Coke1.jpg` and `Coke2.jpg` images exhibit a stable background and camera, which help us estimate the movement of the hand and the coke can. Indeed, in Figure 4b we observe that the movement of the hand is calculated correctly. Note that motion is also calculated for artifacts, like the shadow, which can be an undesired property in some applications.

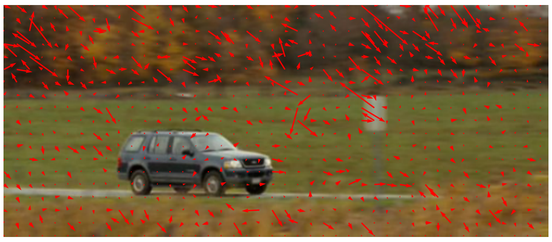
Question 2. The Lucas-Kanade algorithm uses window kernels to compute local motion change, and hence operates on a local scale. On the contrary, the Horn-Schmuck operates on a global scale, since it estimates optical flow by minimizing a global energy function. This constitutes the Lucas-Kanade algorithm less sensitive to noise and the Horn-Schmuck algorithm less prone to the aperture problem. On a final note, Lucas-Kanade algorithm is highly sensitive to the window size that we opt for, while the success of the Horn-Schmuck algorithm depends on the use of noise removal filters as a pre-processing step.

In flat regions, the Lucas-Kanade algorithm doesn't work because the spatial derivatives I_x and I_y are close to zero, and hence the pseudo-inverse of matrix A has numerical issues. On the other hand, the Horn-Schmuck algorithm imposes a smoothing constraint, which makes the flow continuous. Hence the estimated flow of flat regions is that of their surroundings.

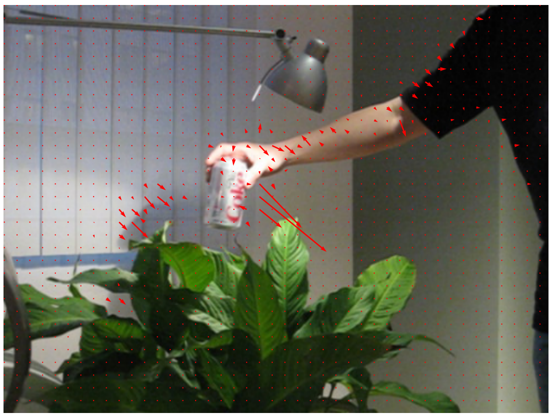
4 Feature Tracking

In this part of the assignment we combine the Harris Corner detector and the Lucas-Kanade optical flow to track the movement of `Toy` and the `Doll` image sequences.

Question 1. To begin with, dataset consists of sequential images of the doll and the toy in a object-centric scene, where the object is directly in front of camera. However there are other



(a) Optical flow observed in the `Car1.jpg` and `Car2.jpg` images.



(b) Optical Flow observed in the `Coke1.jpg` and `Coke2.jpg` images.

Figure 4: Optical Flow for the given pairs of images.

objects like lamps, humans, etc. that are present in the images. Both objects are held in the hands of a human and are moved across various trajectories. However we observe that although ideally only the object should be moving we also observe that other objects such as the human who is holding the object are also moving. Hence these movements apart of the doll/toy moving could also be picked up by the Corner detector. Also, the motion is not small in between some of the frames in the dataset. Therefore, several assumptions made by Lucas-Kanade and Harris Corner algorithms such as brightness constancy between the frames and the small change of movement of the object between the frames are being violated in the given dataset; especially in the Doll dataset as a result of which the algorithms do not give consistent results throughout every iteration.

Question 2. Our simple object tracking systems computes the corner points only for the first frame and then tracks them. This is important for two reason. First, it helps us keep a correspondence of each corner point between frames. Secondly, it is much cheaper computationally, since in optical flow estimation we resort both in less matrix operations and on less images patches that would need to perform these operations.

5 Conclusion

In this assignment we implemented and tested the Harris Corner Detector and the Lucas-Kanade algorithm. Both of them have their limitations, such as the constant brightness assumption or the need of careful tuning of parameters, which can be hindering in real-world applications. However, we saw that we can use them in simple use cases, such as some of the photos provided. We were also able to experiment with a combination of these two methods and build a simple object tracker, with modest results.

References

[1] Chris Harris, Mike Stephens, et al. A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 10–5244. Citeseer, 1988.

[2] Bruce D Lucas, Takeo Kanade, et al. An iterative image registration technique with an application to stereo vision. Vancouver, British Columbia, 1981.

[3] Cordelia Schmid, Roger Mohr, and Christian Bauckhage. Evaluation of interest point detectors. *International Journal of computer vision*, 37(2):151–172, 2000.

[4] Jianbo Shi et al. Good features to track. In *1994 Proceedings of IEEE conference on computer vision and pattern recognition*, pages 593–600. IEEE, 1994.

[5] Richard Szeliski. *Computer Vision - Algorithms and Applications.* Texts in Computer Science. Springer, 2011.