Computer Vision 1 Assignment 3

Dmitrii Krasheninnikov
University of Amsterdam
11719230
dmkr0001@gmail.com

Akash Raj Komarlu Narendra Gupta
University of Amsterdam
11617586
akashrajkn@gmail.com

1 Introduction

This report describes completion of the tasks of the third assignment of the Computer Vision 1 course at the University of Amsterdam. First, we experiment with the Harris corner detection algorithm by implementing it and looking into threshold selection, and understanding the rotation invariance property of the algorithm. Next, the Shi-Tomasi corner detection algorithm is briefly examined. Afterwards we experiment and reason about properties of two optical flow detection algorithms: the Lucas-Kanade algorithm and the Horn-Schunck algorithm. Finally, we combine the Harris corner detector and the Lucas-Canade algorithm to create a simple feature tracking system.

2 Harris Corner Detector

Question 1: Experiments with Harris Corner Detector

1.1 For Question 1 we have implemented and tested the Harris corner detection algorithm. The `harris_corner_detector` function detects the corners and returns the matrix H as well as row and column coordinates of the pixels whose cornerness is above the threshold. The function `demo_harris_corner_detector` uses the `harris_corner_detector` function to run experiments from Question 1.

1.2 The results of applying Harris corner detection to images *person_toy/00000001.jpg* and *pingpong/0000.jpeg* are shown in Figures 1, 2 and 3.

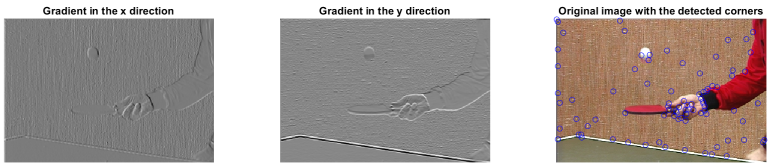

Figure 1: Gradient of *pingpong/0000.jpeg* in x and y directions, and corners detected by Harris corner detection algorithm with a threshold of .08, $n = 2$ and $\sigma = 1$.

For the *person_toy* image the threshold parameters resulting in qualitatively best performance are between .007 and .035. The former value favors recall: most of the "true corner" points are captured, but a large number of erroneous corners (due to the wall texture) are captured as well (Figure 2); the latter value favors precision: the majority of the points found by the algorithm are true corner points, but many other true corner points are not captured (Figure 3).

For the *pingpong* image higher threshold values are needed to filter out the large number of erroneous corners (Figure 1), mainly due to the texture of the wall being more fine-grained and having more variation in intensity values in this image compared to the wall in the *person_toy* image.

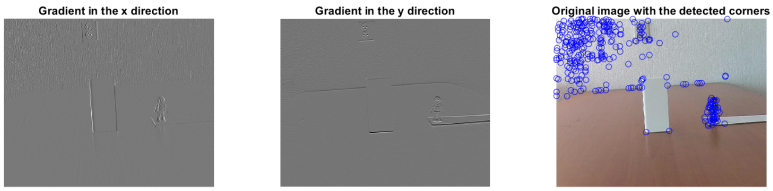

Figure 2: Gradient of *person_toy/00000001.jpg* in x and y directions, and corners detected by Harris corner detection algorithm with a threshold of .007, $n = 2$ and $\sigma = 1$.

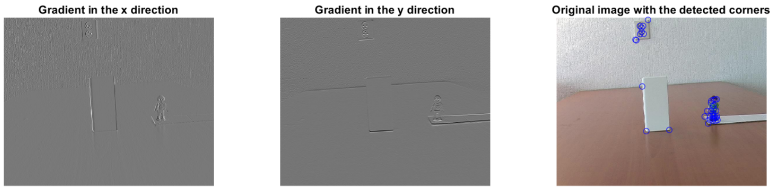

Figure 3: Gradient of *person_toy/00000001.jpg* in x and y directions, and corners detected by Harris corner detection algorithm with a threshold of .035, $n = 2$ and $\sigma = 1$.

1.3 The Harris corner detection algorithm is rotation invariant, since the eigenvalues λ_1 and λ_2 do not change due to rotation (in the continuous space). In practice however we observe a very good *but not perfect* correspondence of corners in the rotated images with corners in the original image (Figures 3 and 4). The imperfect correspondence is likely due to two reasons:

1. The image cannot rotate perfectly unless rotating by multiples of 90 degrees since the pixels are discrete; the image ends up slightly changed as a result of rotation. Additionally, the rotated image is smaller than the original one, and as the hyperparameters were kept the same and not adjusted with the image size. Due to these image changes caused by rotation the detected corner pixels change.
2. The image is rotated, but the windows around each pixel are still parallel to the x and y axes. As such, a different local neighborhood is considered for each pixel, which also insignificantly changes the detected corner pixels.

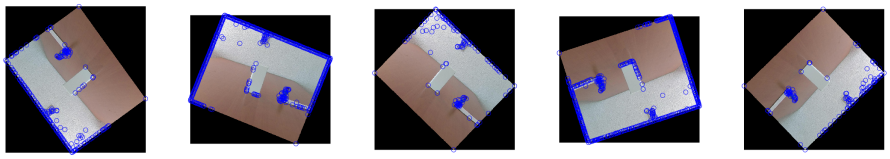

Figure 4: Corners detected by Harris corner detection algorithm with a threshold of .035, $n = 2$ and $\sigma = 1$ on several rotated versions of *person_toy/00000001.jpg* image.

Despite the imperfect correspondence of the corners detected in the original image and the rotated images, the important true corners *are* detected in the rotated images. This highlights that despite the image perturbations caused by rotation it is still acceptable to treat the Harris corner detection algorithm as rotation invariant for the vast majority of practical purposes.

Question 2

The Shi-Tomasi Corner Detector operates similarly to the Harris corner detector, with the cornerness function being the only difference between the algorithms. The cornerness function of the Shi-Tomasi Corner Detector is $H(x, y) = \min(\lambda_1, \lambda_2)$, where λ_1, λ_2 are the eigenvalues of $Q(x, y)$.

In fact, it is not necessary to compute eigen decomposition of the image or patches since $\min(\lambda_1, \lambda_2)$ can be expressed in terms of A, B and C (in the notation of the assignment instruction):

$$\begin{aligned}
\min(\lambda_1, \lambda_2) &= \frac{\lambda_1 + \lambda_2 - \sqrt{(\lambda_1 - \lambda_2)^2}}{2} \\
&= \frac{\lambda_1 + \lambda_2 - \sqrt{\lambda_1^2 - 2\lambda_1\lambda_2 + \lambda_2^2}}{2} \\
&= \frac{\lambda_1 + \lambda_2 - \sqrt{(\lambda_1^2 + 2\lambda_1\lambda_2 + \lambda_2^2) - 4\lambda_1\lambda_2}}{2} \\
&= \frac{A + C - \sqrt{(A + C)^2 - 4AC + 4B^2}}{2} \\
&= \frac{A + C - \sqrt{A^2 + 2AC + C^2 - 4AC + 4B^2}}{2} \\
&= \frac{A + C - \sqrt{(A - C)^2 + 4B^2}}{2}
\end{aligned}$$

When both eigenvalues are near zero or one eigenvalue is near zero and the other one is large, the cornerness value $\min(\lambda_1, \lambda_2)$ is near zero and likely below $\lambda_{threshold}$, so no corner is detected at the pixel. Only when both eigenvalues are big cornerness $\min(\lambda_1, \lambda_2)$ is big (and likely above $\lambda_{threshold}$), the pixel is considered to be a corner (Figure 5).

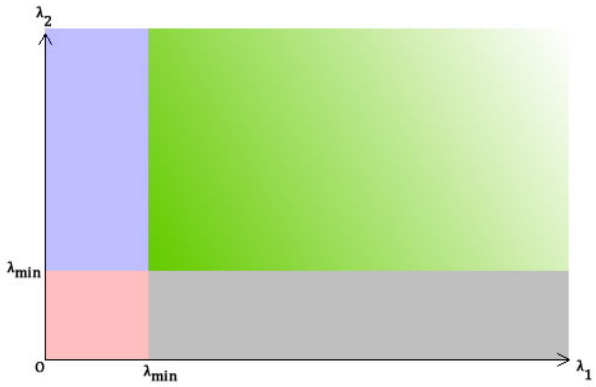

Figure 5: Shi-Tomasi cornerness depending on eigenvalues of Q. The values of λ_1, λ_2 in the green region correspond to the corner being detected. Image source: [1].

3 Optical flow

Question 1

Let the intensity of an image at position (x, y) at time t be $I(x, y, t)$. Under the assumption of *brightness constancy*, two frames at time t and $t + \partial t$ have the same intensity. This results in the following constraint,

$$I_x V_x + I_y V_y + I_t = 0$$

where V_x and V_y are the components of the optical flow in x and y directions respectively. Lucas and Kanade assume that the optical flow is constant in the local neighborhood of a pixel, which results in,

$$\begin{bmatrix} I_x(q_1) & I_y(q_1) \\ \vdots & \vdots \\ I_x(q_n) & I_y(q_n) \end{bmatrix} \begin{bmatrix} V_x \\ V_y \end{bmatrix} = \begin{bmatrix} -I_t(q_1) \\ \vdots \\ -T_t(q_n) \end{bmatrix}$$

where $q_1, q_2, ...q_n$ are pixels in the neighborhood of the pixel under consideration. This system of equations is solved by the weighted least squares method (using matlab's in-built function, `pinv`), $[V_x, V_y] = (A^T A)^{-1} A^T b$. Figure 6 shows the optical flow vectors (depicted as white arrows) on the `synth` and `sphere` images.

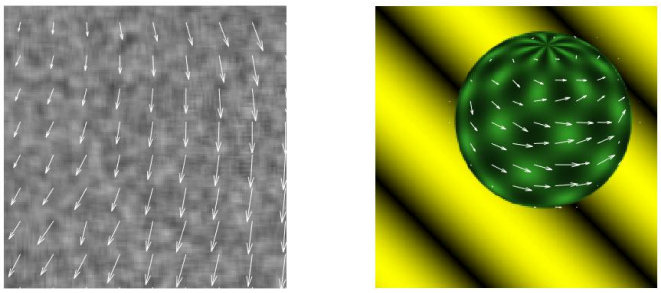

Figure 6: Optical flow of the *synth* images (left) and *sphere* images (right)

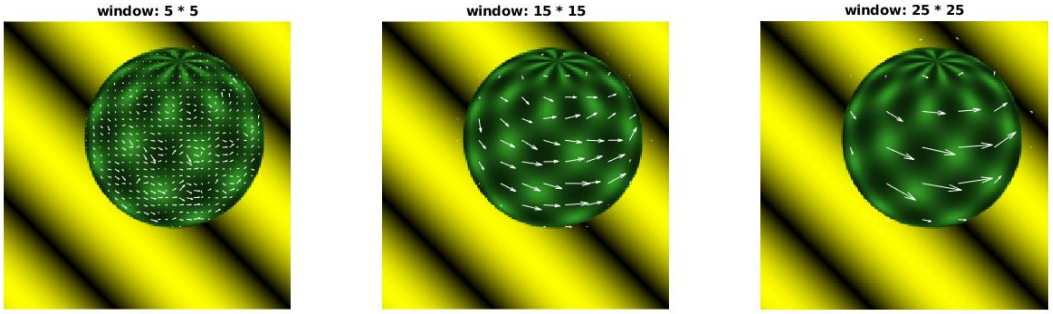

Figure 7: Optical flow of the *sphere* images with different window sizes. As the window size decreases, it is more sensitive to noise.

Question 2

The Horn-Schuck algorithm assumes that motion of an object in the direction perpendicular to that of the image brightness gradient does not result in a change in brightness. These components of motion cannot be determined locally. 'Opaque' objects can be viewed as undergoing rigid motion; therefore, points close to each other might have similar velocities. The smoothness constraint can be expressed by minimizing the square of the gradient magnitude of the flow,

$$\nabla^2 u = \left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 \quad \text{and} \quad \nabla^2 v = \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2$$

where u and v denote the components of the velocity, and x and y are the components of the optical flow.

Horn and Schunck [2] constrain the estimated velocity by combining image gradient constraint with a global smoothness term, by minimizing the equation

$$\int\int \left(\nabla I_x u + I_y v + I_t\right)^2 + \lambda^2 \left(||\nabla u||_2^2 + ||\nabla v||_2^2\right) dxdy$$

λ is a hyper parameter of the model which denotes the influence of the smoothing term. The Lucas Kanade algorithm works at a local level by assuming spatial constancy, while the Horn-Schunck algorithm is a global method since it also uses global information in the smoothing term. Oftentimes global methods result in a dense optical flow field whereas local level algorithms are more robust to noise [3].

In flat image regions the gradient is zero, and thus the inverse for computing $[V_x, V_y]$ would not exist, making Lucas-Kanade fail. On the other hand, due to the smoothness term the Horn-Schunck method still produces smooth flow vectors in the flat image regions.

4 Feature Tracking

Question 1

For this question, we implemented 'Feature tracking' using the results of Harris corner detection and the Optical flow (using Lucas Kanade algorithm). Using the first images of the `pingpong` and `person_toy`, we computed the *interest points* on the respective images. For the subsequent images, the position of the selected features are approximated using the optical flow vectors obtained from Lucas-Kanade algorithm. The optical flow is estimated for the neighborhood of the interest points. We used a window of size $15 * 15$. The video files for each set of images can be found on the *videos* folder.

Figure 8: Feature tracking for `pingpong` images.

Ideally, feature detection can be performed only once in the first frame. Due to reasons such as non-smooth edges, or an object being hidden (occlusion), etc., the predicted features have a high error rate. Figure 9 shows this scenario. The predicted feature position is slightly left to the actual feature position. For a small window size, the small error propagates as more frames are processed.

To correct this, we generate features every p frames (We used $p = 4$ in the video).

Question 2

With feature tracking, the relation (between) objects in different frames can be obtained. Using many frames to track features reduces sensitivity to noise when detecting features [4]. Moreover, detecting features in each and every frame will in most cases be more computationally expensive.

5 Conclusion

In this assignment we have experimented with the Harris corner detector and the Lucas-Kanade optical flow algorithm, and combined the two to create a simple feature tracking system. Additionally, we examined the theoretical properties of the above-mentioned algorithms as well as theoretical properties of the Shi-Tomasi corner detector and the Horn-Schunck optical flow algorithm. While working on the assignment we found it interesting to think about using more advanced algorithms such as neural networks for corner and optical flow detection, and which problems this may solve.

Figure 9: Error in the prediction of feature position. The vectors are slightly off the actual feature position.

Figure 10: Feature tracking for `person_toy` images. Figure shows the flow vectors (black) after the 39^{th} frame.

References

[1] Utkarsh Sinha. Fundamentals of features and corners: The shi-tomasi corner detector, 2016.

[2] Horn B.K.P and Schunck B.G. Determinig optical flow. *Artificial Intelligence*, 17:185–203, 1981.

[3] Weickert J. Bruhn, A. and C. Schnorr. Lucas/kanade meets horn/schunck: Combining local and global optic flow methods. *International Journal of Computer Vision*, 61(3):211–231, 2005.

[4] Carol Tomasi and Takeo Kanade. Shape and motion from image streams: a factorization method. 1991.