Group 38 # Lab 3: Report Computer Vision 1
October 3, 2021

Introduction

In this assigment, we look at two very fundamental tasks in Computer vision: corner detection and optical-flow estimation. For each of these, we look at some basic but widely used algorithms, implement them, experiment with their parameters and make inferences based on our observations. We also tried to mathematically prove answers to some of the questions, e.g., proving that Harris corner detection is rotation-invariant. Further, we also perform feature-tracking using the combination of these two tasks.

1 Harris Corner Detector

Answers to question 1:

1. The Harris corner detector has been implemented in `harris_corner_detector.py`.
2. For the two sample images, results obtained with threshold of 0.001, window size of 5×5 and Gaussian kernel $\sigma = 1.0$ are shown in Figure 1. Note that we experiment with various thresholds for each of the two images and notice that 0.001 seems like a decent pick. The results for varying thresholds are shown in Figure 3. As the threshold increases, naturally, the number of corners detected reduces.
3. Harris corner detector ($\mathbb{H}$) is rotation invariant because if we consider a local patch around pixel (x, y), if we rotate the image, then although the directions of eigenvectors will change, the eigenvalues will remain the same. The proof is based on the observation that $\mathbb{H}$ depends on image gradients I_x, I_y and these are rotation equivariant, i.e, if the image rotates by θ, then the gradients also rotate by θ. A more rigorous proof is given in the Appendix A.1. The results for the sample image with 45 and 90 degree rotation is shown in Figure 2. Even if the image is rotated, the algorithm still finds the corners pretty much with similar accuracy. There is still some noise, for example, some corners on the man's shirt are not detected in 45 degree case. This may be attributed to violation of assumption that the intensity is not preserved in rotation since rotation involves interpolation of pixel intensities. Also, note that rotated image will have some padding which sort of introduces artificial edges which may cause corner detection on points that are not actual corners.

Answers to question 2:

1. The Shi-Tomasi corner detection (say, denoted by $\mathbb{S}$) is very similar to Harris corner detection except that the score is computed as follows:

$$\mathbb{S} = \min\{\lambda_1, \lambda_2\} \tag{1}$$

2. Properties of Shi-Tomasi corner detector:

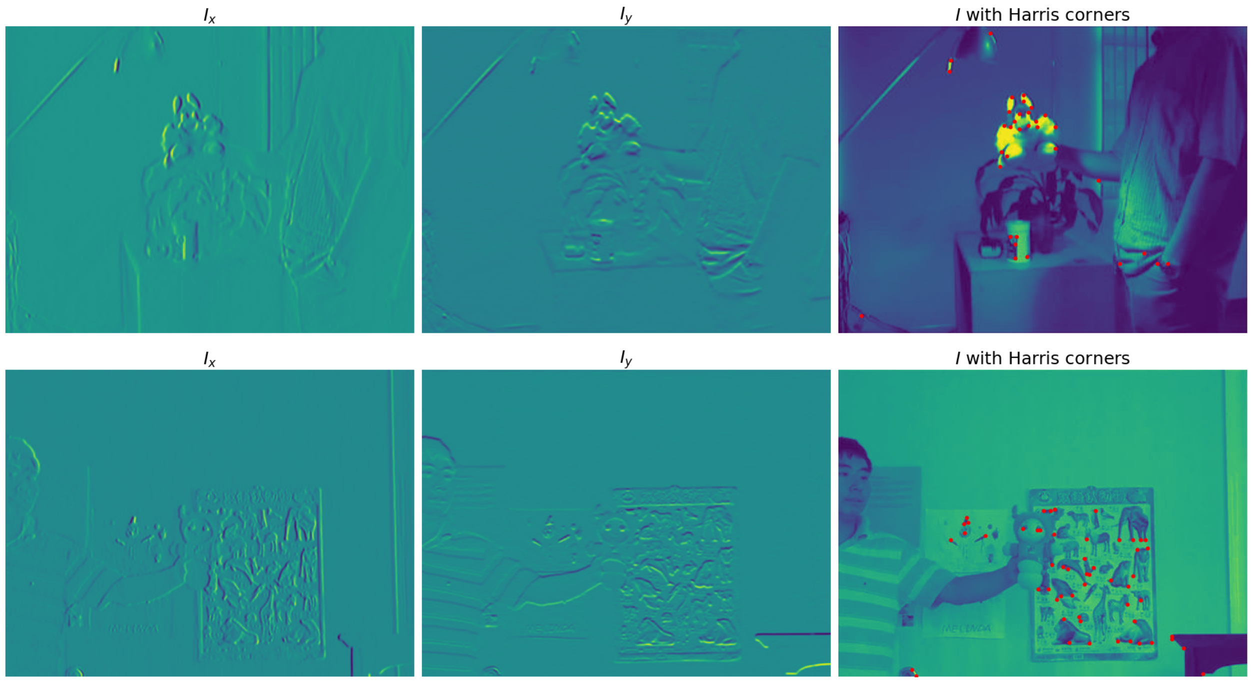


Figure 1: Harris corner detector: Results for given sample images.

- It is translation invariant since the image gradients themselves are invariant and M, and, thus the eigenvalues remain the same.
- $\mathbb{S}$ is rotation-invariant since Harris is rotation invariant as shown in the previous part (since eigenvalues remain the same, so does the min).
- $\mathbb{S}$ is not scaling-invariant. Intuitively, we can consider a case where a corner may get scaled up to become an edge in the scaled up image. Mathematically, the two eigenvalues could be scaled differently and the minimum eigenvalue may not remain the minimum after scaling.

3. Relative cornerness

- When both λ_1, λ_2 are close to zero, $\min\{\lambda_1, \lambda_2\}$ is also close to zero and it is a non-corner.
- When λ_1 is high and λ_2 is near zero, $\min\{\lambda_1, \lambda_2\} = \lambda_2 \to 0$ and thus it is a non-corner.
- When both eigenvalues are high, the minimum will also be high and thus it will be a corner.

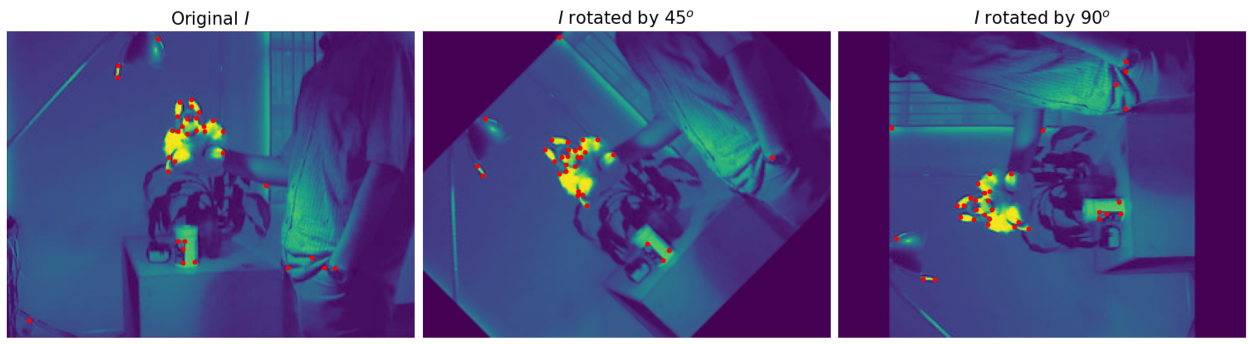


Figure 2: Illustration of rotation invariance of Harris corner detection algorithm.

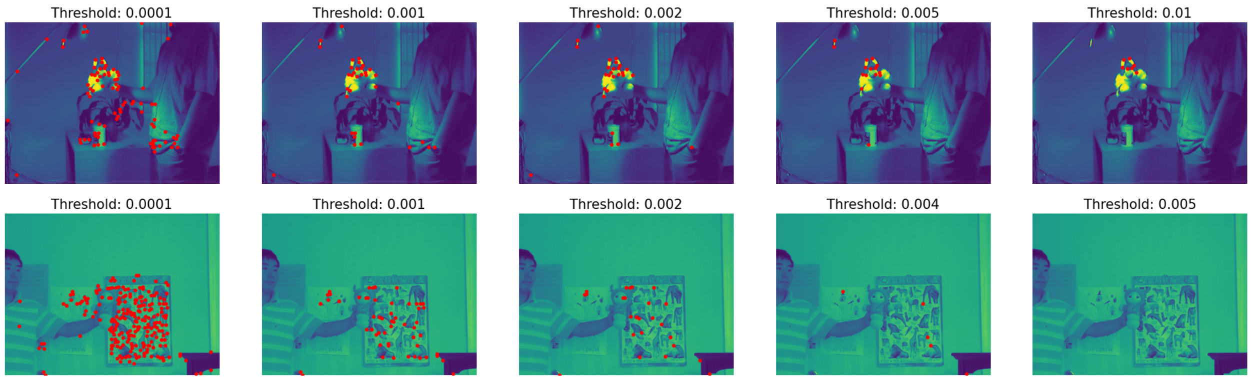


Figure 3: Harris corner detector: Impact of varying threshold.

2 Optical Flow

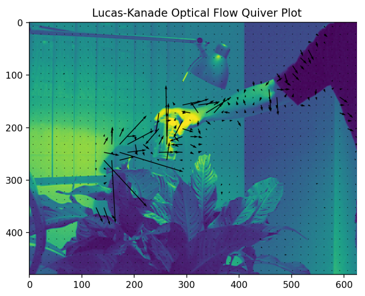
2.1 Lucas-Kanade Algorithm

Answers to question 1: The Lucas-Kanade algorithm for optical flow has been implemented in `lucas_kanade.py`. The results for sample images have been shown in Fig. 4.

Answers to question 2:

1. The Lucas-Kanade algorithm operates on a local scale. An image is divided into non-overlapping regions, and the optical flow for each region is calculated individually. The Horn-Schunck algorithm includes a global smoothness constraint, and therefore operates at a global scale.

2. The Lucas-Kanade algorithm cannot provide any flow information at flat regions. A flat region has no flow information, and therefore any motion in that region cannot be inferred. The global smoothness constraint in the Horn-Schunck algorithm minimizes flow distortions in an image. To meet this constraint, a flat region will assigned similar flow values to the regions surrounding it.



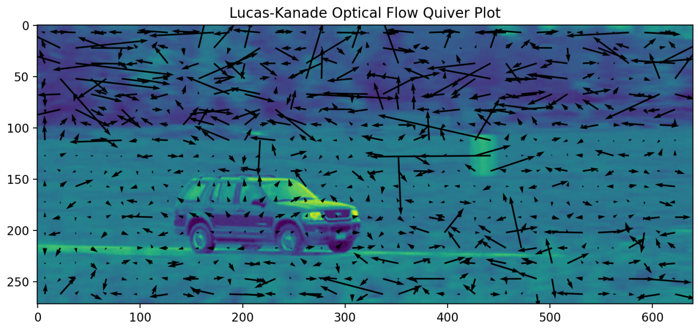


Figure 4: Optical flow using Lucas-Kanade algorithm.

3 Feature Tracking

Answers to question 1:

1. The feature-tracking algorithm is implemented in `tracking.py`. The script generates video of the flow and is stored in `results/toy_flow.py`. A sample set of frames with the estimated Harris corners and flow directions is shown in Figure 5 for the two examples provided. Through our implementation, we experimented with some of the parameters (e.g. threshold for Harris corner detection).

   - For Harris corner detector, we chose threshold to be 0.0001 since not enough points were detected with a higher threshold for the `doll` example, especially in the area that has motion.
   - We played around with various values of `scale` parameter of the `quiver` functionality and chose 0.05 since it seemed to show reasonable arrows for both examples.

2. Code for visualizing video of feature-tracking is provided in `tracking.py`.

Answers to question 2: Feature detection on different frames may not correspond to the exact same points in the 3D world projected onto the image. With feature tracking, we get perfect correspondence across images since we track precisely the same set of points. Also, detecting features from scratch may be computationally more expensive than tracking features.

Conclusion

We looked at really simple yet powerful algorithms in computer vision for two fundamental tasks: corner detection and optical flow estimation. For corner detection, we studied the

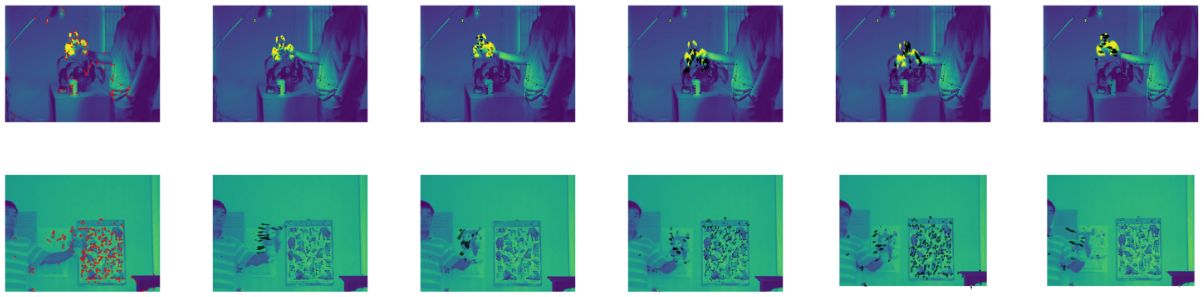


Figure 5: Feature tracking using optical flow estimated via the Lucas-Kanade algorithm.

Harris detector and Phi-Tomasi detector. For optical flow, we studied, implemented the Lucas Kanade algorithm and also looked at Horn-Schnuck method. Finally, we saw how these two could be combined to do feature-tracking that provides perfect correspondence.

A Harris Corner Detector

A.1 Rotation Invariance

Lemma 1. *Image gradients are rotation equivariant. Formally, if an image I is rotated by θ, then image gradients of the rotated image will be given by rotating I_x, I_y by θ.*

Proof. Let $I(x, y)$ define an image as a function of pixel co-ordinates (x, y). A rotation of the image by θ means the following transformation in the pixel co-ordinates:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = R_\theta \begin{bmatrix} x \\ y \end{bmatrix} \tag{2}$$

The image gradients (in original image co-ordinates) are defines as:

$$I_x = \frac{\partial I}{\partial x};\ I_y = \frac{\partial I}{\partial y} \tag{3}$$

Now, we can use the Chain rule to obtain the image gradients in the transformed rotated space.

$$\begin{aligned} I_{x'} &= \frac{\partial I}{\partial x'} = \frac{\partial I}{\partial x}\frac{\partial x}{\partial x'} + \frac{\partial I}{\partial y}\frac{\partial y}{\partial x'} = \frac{\partial I}{\partial x}\cos\theta + \frac{\partial I}{\partial y}\sin\theta \\ I_{y'} &= \frac{\partial I}{\partial y'} = \frac{\partial I}{\partial x}\frac{\partial x}{\partial y'} + \frac{\partial I}{\partial y}\frac{\partial y}{\partial y'} = -\frac{\partial I}{\partial x}\sin\theta + \frac{\partial I}{\partial y}\cos\theta \end{aligned}$$

Thus, substituting values, we get

$$\begin{bmatrix} I_{x'} \\ I_{y'} \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} I_x \\ I_y \end{bmatrix} = R_\theta \begin{bmatrix} I_x \\ I_y \end{bmatrix} \tag{4}$$

Hence, proved. □

Lemma 2. *If A and B are matrices such that $A = SBS^{-1}$ where S is an invertible matrix and $(\lambda, \mathbf{v})$ is an eigen-pair for B, then $(\lambda, S\mathbf{v})$ is an eigen-pair for A.*

Proof. Note that, from given information, $S^{-1}AS = B$. Since λ is an eigenvalue for B,

$$\begin{aligned} B\mathbf{v} &= \lambda\mathbf{v} \\ S^{-1}AS\mathbf{v} &= \lambda\mathbf{v} \\ AS\mathbf{v} &= \lambda S\mathbf{v} \\ A(S\mathbf{v}) &= \lambda(S\mathbf{v}) \end{aligned}$$

Thus, λ is an eigenvalue for A with eigenvector $S\mathbf{v}$. □

Theorem 1. *Harris Corner detector $\mathbb{H}$ is rotation-invariant.*

Proof. Note that $\mathbb{H}$ is completely determined by the eigenvalues of the matrix:

$$M := \begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix} = \begin{bmatrix} I_x \\ I_y \end{bmatrix} \begin{bmatrix} I_x & I_y \end{bmatrix}$$

Now, we can get matrix M' for the rotated image using Lemma 1:

$$M' = \begin{bmatrix} I_{x'} \\ I_{y'} \end{bmatrix} \begin{bmatrix} I_{x'} & I_{y'} \end{bmatrix} = R_\theta \begin{bmatrix} I_x \\ I_y \end{bmatrix} \begin{bmatrix} I_x & I_y \end{bmatrix} R_\theta^T = R_\theta M R_\theta^T$$

Note, R_θ is rotation matrix and is orthogonal ($R_\theta^{-1} = R_\theta$). By Lemma 2, eigenvalues of M and M' are exactly identical. Since $\mathbb{H}$ is completely determined by eigenvalues of M, hence the proof. $\square$

Note: This proof will only be approximately true in practice since we use Gaussian smoothing while computing gradients.