Harris Corner Detector and Optical Flow

Tarun Krishna
University of Amsterdam
11593040
tarun.krishan@student.uva.nl

Dhruba Pujray
Univeristy of Amsterdam
11576200
dhruba.pujray@student.uva.nl

1 Introduction

Corner detection is an approach used within computer vision systems to extract certain kinds of features and infer the contents of an image. Corner detection is frequently used in motion detection, image registration, video tracking, image mosaicing, panorama stitching, 3D modeling and object recognition. Corner detection overlaps with the topic of interest point detection. In this report, we implement some of the most basic corner detection methods and use them for feature tracking using optical flow.

2 Harris Corner detector

Harris corner detector is one of many such feature detectors to extract corners or interest points from an image. Corners are the important points in the image which are invariant to illumination, translation or rotation and can be used for motion tracking, image stitching and many more. The intuition behind Harris corner detector is, given a window over an image if it is shifted along the edge or orthogonal to the edge, the change will be less as compared to shifts at L-corners or junction like T or Y.1

Concretely, given a window which is shifted by Δx and Δy, the change in the image or the autocorrelation is defined as

$$E(x,y) = \sum_{(u,v)\in W(x,y)} w(u,v)(I(u+\Delta x, v+\Delta y) - I(u,v))^2$$

where $W(x,y)$ is a window centered at point (x,y) and $w(u,v)$ is a Gaussian function. Using first-order Taylor expansion of $I(u+\Delta x, v+\Delta y)$ and using matrix representation, we define $Q(x,y)$ which is a matrix which captures the gradient information of the patch.

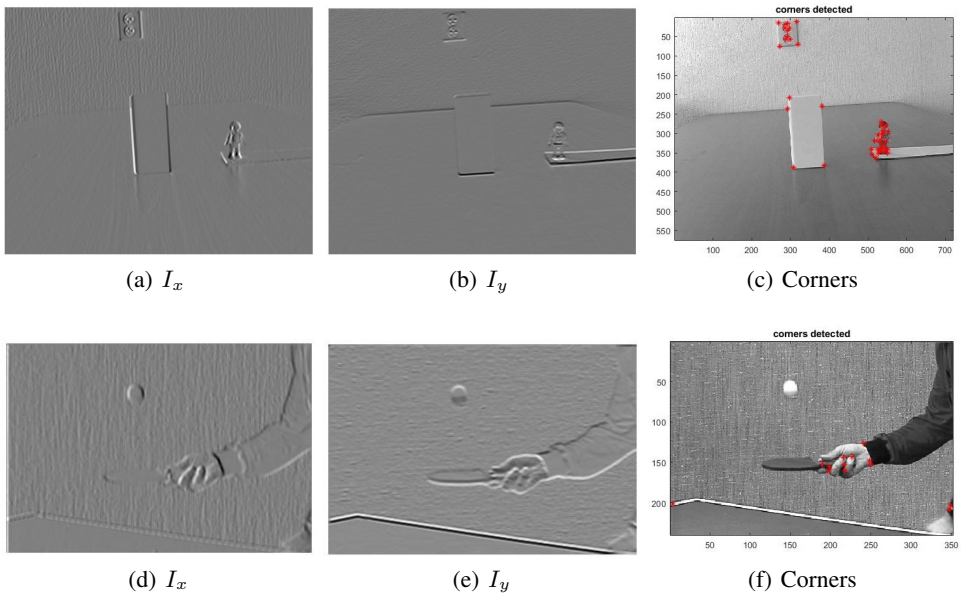
$$Q(x,y) = \sum_{(u,v)\in W(x,y)} \begin{bmatrix} I_x(x,y)^2 & I_x(x,y)I_y(x,y) \\ I_x(x,y)I_y(x,y) & I_y(x,y)^2 \end{bmatrix}$$

Eigenvalue analysis of the auto-correlation matrix produces two eigenvalues (λ_1, λ_2) and based on their magnitude the cornerness can be defined as

$$R = \lambda_1\lambda_2 - k(\lambda_1 + \lambda_2)$$

Computing the eigenvalues are computationally expensive so to find corner and interest points the determinant and trace of $Q(x,y)$ is used. To identify a fixed point among the probable clusters of points, non-maximum suppression is applied which is basically looking in the neighborhood of a point and keeping only the maximum value within the neighborhood.[1] A threshold is also used to

[1] Experiment with different threshold.



(a) I_x (b) I_y (c) Corners

(d) I_x (e) I_y (f) Corners

Figure 1: Harris corner detector for Person toy and Pingpong

consider only those points that can be classified as corners. We used 2 as our threshold value for a Gaussian window size 7 and sigma 3 for *person toy*. And for *pingpong* we used a threshold of 18 for a Gaussian window of size 5 and sigma 3. Increasing the threshold value will reduce the number of points that are classified as corners Figure2 . To determine a good threshold value, we plotted the histogram of corner response values before and after non-maximum suppression. The Histogram of corner response values before non-maximum suppression will give us the range corner response values. The highest corner response values will lie towards the right side in the plot. Similarly, the plot of histogram of corner response after non-maximum suppression will give us the range of the maximum values within the cluster of points. By comparing both the plots we can find a suitable threshold values that would suffice the need of determining a good threshold.

NOTE: *run demo_harris.m with input as 'person' or 'pingpong' followed by second argument 'true' or 'false' for rotation.*

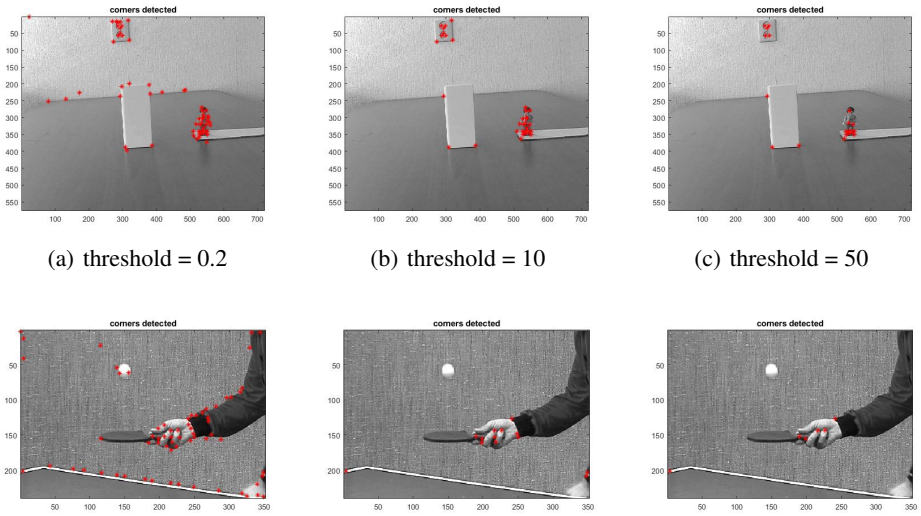
[2] The eigenvalues of auto-correlation matrix can be represented in an ellipse where the major axis is equal to $(\lambda_{min})^{-\frac{1}{2}}$ and minor axis is equal to $(\lambda_{max})^{-\frac{1}{2}}$. The ellipse is invariant to rotation, i.e. its major and minor axis remains the same. Thus, the corner response R is also invariant to rotation. The Figure4 shows the number of corner within the image remains the same when it is rotated by an angle of 63.5 degrees. The points along the edges should be ignored because they are evaluated based on changes with the boundary and the background; and also the actual corner of the rotated image gets captured. Takeaway lesson from this experiment is, if our corner detection algorithm is rotation invariant then we should be able to sustain the corners in the rotated image as well i.e "repeatability", which is one of the major criteria for evaluating various corner detection algorithms.

[3]According to Shi and Tomasi[3], the features like corners, or windows with a high spatial frequency content are problematic for tracking. They proposed a modified definition of cornerness which considers a feature to be an interest point only if the minimum of the two eigenvalues in larger than a predefined threshold. This definition makes feature tracking good and also the selection criteria is optimal by construction.

$$R = min(\lambda_1, \lambda_2)$$

[2]Is the algorithm rotation-invariant? Explain your answer and support it with your observations.

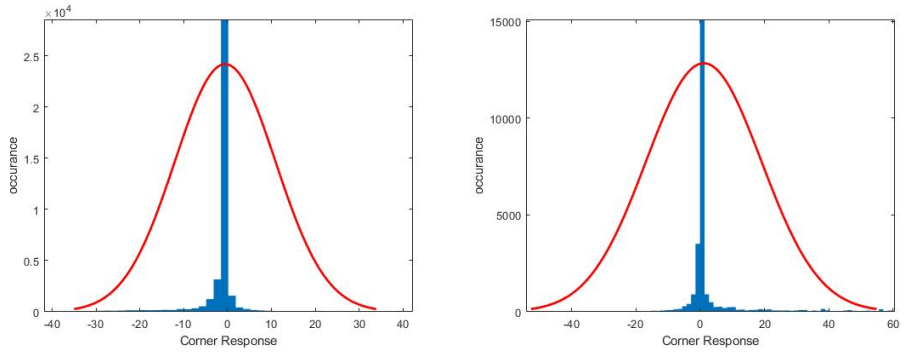
[3]How do they define cornerness? Write down their definition using the notation of Equation 10.



(a) threshold = 0.2 (b) threshold = 10 (c) threshold = 50

(d) threshold = 3 (e) threshold = 18 (f) threshold = 40

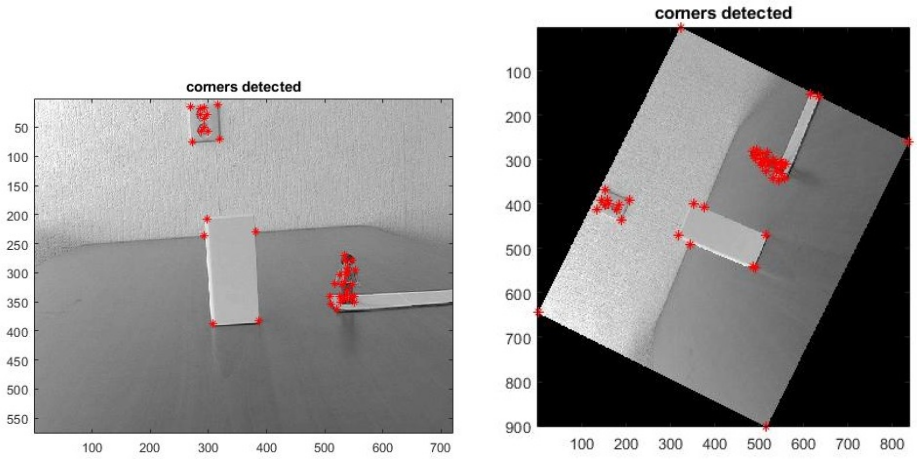
Figure 2: Harris corner detector using different threshold values



(a) before (b) after

Figure 3: Histogram of corner response before and after non-maximum suppression

[4]The eigenvalues are computed with respect to a window or patch and not the whole image at once. The small shift in a patch will be able to capture the difference along edges or corners. If both the eigenvalues of the patch are above some threshold we consider that patch to have an interest point. [5]If both the eigenvalues are small and close to zero would mean the surface has constant intensity or flat region. The value assigned would be near to zero. If one eigenvalue is large and another one is small would mean a unidirectional texture pattern, like edges. The value assigned would be minimum one, i.e. near to zero. If both the eigenvalues are large would mean corners, salt-and-pepper textures or other patterns that can be tracked reliably[5]. The value assigned would be the minimum of the two eigenvalues.



(a) before (b) after

Figure 4: Harris corner detector invariant to rotation

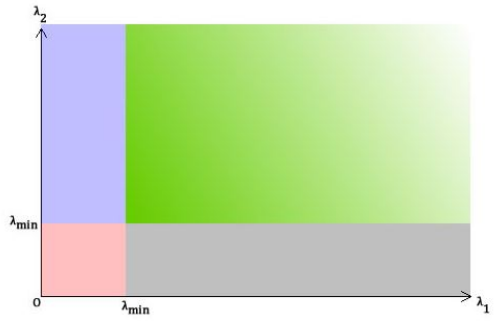
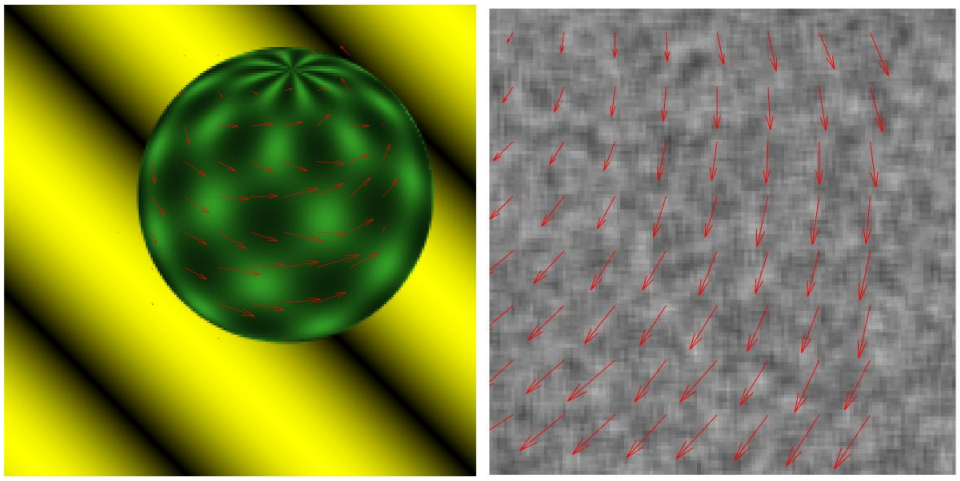


Figure 5: Effect region for a point to be a corner by Shi and Tomasi



(a) Sphere optical-flow (b) Synth optical-flow

Figure 6: Results from Lucas-Kanade implementation

3 Optical Flow with Lucas-Kanade Algorithm

Shown in Figure6 are the results from the implementation of Lucas-Kanade implemented in *lucas_kanade.m*

[6]The basic intuition behind **Lucas_Kanade** [2] is that it assumes that the flow is essentially constant in a local neighborhood of the pixel under consideration, and solves the basic optical flow equations for all the pixels in that neighborhood, by the least squares criterion. This method is a local approach for solving aperture problem. While on the other hand **Horn_Schunck method** [1] estimating optical flow is a global method which introduces a global constraint of smoothness to solve the aperture problem. The idea is to optimize a function based on residuals from the brightness constancy constraint, and a particular regularization term expressing the expected smoothness of the flow field. It is more sensitive to noise than local methods. This method is based on the variational differential, which is given by:

$$E = \int\int [(I_x u + I_y v + I_t)^2 + \alpha^2(||\nabla u||^2 + ||\nabla v||^2)]dxdy$$

[7] Based on the formulation of these two methods, they behave differently on flat regions. To be more precise **Horn_Schunck** method is able to deal with flat regions as compared to **Lucas_Kanade** which fails to address this problem. Let's look at both methods one by one. **Lucas_Kanade** is based on solving least squares which is given by:

$$\begin{bmatrix} V_x \\ V_y \end{bmatrix} = \begin{bmatrix} \sum_i I_x(q_i)^2 & \sum_i I_x(q_i)I_y(q_i) \\ \sum_i I_x(q_i)I_y(q_i) & \sum_i I_y(q_i)^2 \end{bmatrix} \begin{bmatrix} -\sum_i I_x(q_i)I_t(q_i) \\ -\sum_i I_y(q_i)I_t(q_i) \end{bmatrix}$$

The solution given above is the best possible, whenever square matrix is invertible. This might not be the case, if the pixel (x, y) is located in a region with no structure(flat) (for example, if I_x, I_y and they are all zero for all pixels in the neighborhood). Even if the matrix is invertible it can be ill conditioned, if its elements are very small and close to zero. In order to be solvable, eigenvalues must satisfy $\lambda_1 \geq \lambda_2 > 0$. To avoid noise issue, usually λ_2 is required to not be too small. Also, if λ_1/λ_2 is too large, this means that the point is on an edge, and this method suffers from the aperture problem also the same is valid even in case of flat region where $\lambda_1 \approx 0$ and $\lambda_2 \approx 0$. So for this method to work properly, the condition is that λ_1 and λ_2 are large enough and have similar magnitude. This condition is similar to the one, we had for Corner Detection. This observation shows that one can easily tell which pixel is suitable for the Lucas–Kanade method to work on by inspecting a single image. However, **Horn_Schunck** which is based on the global estimate resolves this problem of flat region. **Horn_Schunck** algorithm yields a high density of flow vectors, i.e. the flow information missing in inner parts of homogeneous objects is filled in from the motion boundaries, i.e in parts of the image where the brightness gradient is zero, the velocity estimates will simply be averages of the neighboring velocity estimates.

NOTE: *run demo_lucas.m with input as 'synth' or 'sphere'*

4 Feature Tracking

One of the observations for 'pingpong' dataset. It kind of violates some of the assumptions of optical flow. Firstly, there is a rapid change of motion in subsequent frames as well as there is a obstruction of corner points as you will see at the end of the video, that hand of the player comes in.

[8] To make the process of feature tracking computationally efficient we used to track features which are already detected in the initial frame sequence, rather than calculating corner or interest points

[4]Do we need to calculate the eigen decomposition of the image or the patches? Explain your answer.

[5]In the following scenarios, what could be the relative cornerness values assigned by Shi and Tomasi? Explain your reasoning. a) Both eigenvalues are near to 0. b) One eigenvalue is big and the other is near zero. c) Both eigenvalues are big.

[6]At what scale those algorithms operate; i.e local or global? Explain your answer.

[7]How do they behave on flat regions?

[8]Why do we need feature tracking while we can detect features for each and every frame?

in every frame and tracking it. Detecting features in subsequent frames may not be an ideal way of doing feature tracking as new interest points may get introduced every time. Keeping the assumption of optical flow intact, feature tracking must correspond to keeping track of points initially detected rather than introducing some new corner points in subsequent frames which may not be the desired interest point that we want to track.

NOTE: *A few implementation details for file tracking.m it takes folder-name as input i.e 'pingpong' or 'person_toy' and creates two folder with name pingpong_track and person_toy_track and finally saves the video file with names pingpong.avi and person_toy.avi respectively.*

5 Conclusion

In this report we have analyzed Harris corner detector, the importance of threshold and the rotational invariance of the algorithm. We also mentioned the difference with the Shi and Tomasi algorithm and their corner response conditions. We also analyzed optical flow with Lucas-Kanade algorithm and its difference form Horn-Schunck method. Finally, we tracked the interest point identified by the Harris corner detector using the Lucas-Kanade algorithm over a sequence of images.

References

[1] Berthold K.P. Horn and Brian G. Schunck. Determining optical flow. Technical report, Cambridge, MA, USA, 1980.

[2] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence, IJCAI '81, Vancouver, BC, Canada, August 24-28, 1981*, pages 674–679, 1981.

[3] Jianbo Shi and Carlo Tomasi. Good features to track. Technical report, Ithaca, NY, USA, 1993.