# HARRIS CORNER DETECTOR

Vishal Yadav
2018csb1128
2018csb1128@iitrpr.ac.in

Indian Institute of Technology
Ropar, 140001
Punjab, India

**Abstract**

This document is the report of Part-2 CS518 Computer Vision Assignment 2, where we implement the Harris Corner Detection Algorithm for Corner Detection from first principles. The Harris corner detector uses a multi-stage algorithm to detect a wide range of corners in images. It was developed by Chris Harris and Mike Stephens[1] in 1988.: Corner detection is a traditional type of feature point detection method. Among methods used, with its good accuracy and the properties of invariance for rotation, noise and illumination, the Harris corner detector is widely used in the fields of vision tasks and image processing. Although it possesses a good performance in detection quality, its application is limited due to its low detection efficiency. The efficiency is crucial in many applications because it determines whether the detector is suitable for real-time tasks.

## 1 Introduction

The Harris corner detector is a standard technique for locating interest points on an image. Also, it is a mathematical operator that finds features (i.e corners) in an image. It is based on the auto correlation function of the signal. The basic idea of this detector is we find whether point shows significant change in all direction or not. If yes then point is marked as a corner point. To do this second moment matrix and corner function is calculated. If both of the Eigen values of the second moment matrix are large and nearly equal than that point are considered as the corner point. Also, it is popular because it is rotation, scale and illumination variation independent.

**Key idea** is that Harris used the **calculus of variations** – a technique which finds the function which optimizes a given functional.

## 2 Methodology | Algorithm

The Harris Corner detection algorithm is composed of 4 Steps :

- Luminence Extraction
- Filtered Gradient calculation
- Finding Possible Corners
- Non-maximal Suppression

## 2.1 Luminence Extraction

Harris corner point detectors use only the grayscale information of an image.However, the color information of an image is wasted.So, We have worked on 2D luminence image, instead of the original 3D RGB image. Since RGB doesn't have any role to be played in corner detection.
In our example, we have converted the image to float64 and got the luminence image.

**Algorithm 1** convertToLuminenceImage(img)

1: $R\_channel \leftarrow img[:,:,0]$
2: $G\_channel \leftarrow img[:,:,1]$
3: $B\_channel \leftarrow img[:,:,2]$
4: $luminence \leftarrow (0.2989 * R\_Channel) + (0.5870 * G\_Channel) + (0.1140 * B\_Channel)$
5: **return** luminence

(a) Original Image (b) Luminence Image

Figure 1: Original Image and Luminence Image

## 2.2 Filtered Gradient Calculation

The Gradient calculation step detects the edge intensity and direction by calculating the gradient of the image using edge detection operators. Edges correspond to a change of pixels' intensity. To detect it, the easiest way is to apply filters that highlight this intensity change in both directions: horizontal (x) and vertical (y)
In our example, we have applied the Sobel Filter (approximated Gaussian derivative) in both directions and return F_x and F_y.

**Algorithm 2** getImageGradientComponents(img)

1: $sobel\_filter\_x \leftarrow array[[-1,0,1],[-2,0,2],[-1,0,1]]$
2: $sobel\_filter\_y \leftarrow array[[1,2,1],[0,0,0],[-1,-2,-1]]$
3: $Fx \leftarrow convolve(img, sobel\_filter\_x)$
4: $Fy \leftarrow convolve(img, sobel\_filter\_y)$
5: **return** F_x, F_y

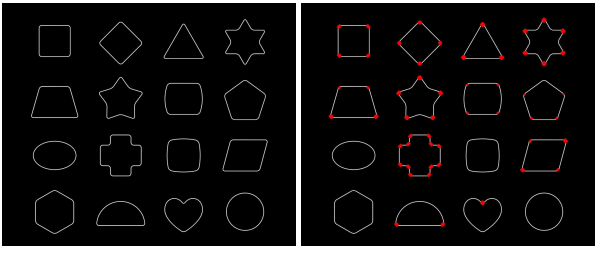
## 2.3 Finding Possible Corners

For each pixel (x, y), we look in a window of size 2m+1 x 2m+1 around the pixel (we have taken various m for visualization) and accumulated over this window to get the covariance matrix C, which contains the average of the products of x and y gradients.

Then using the R_values of corner (formula for detection in original Harris corner detector paper) and thresholding it, we get the possible corner_list. We have also used a dynamic Threshold for finding corners.

**Algorithm 3** findCorners(img, Ix, Iy, m, k, thresh = 0.01)

Initialize $Ixx = Ix^2, Iyy = Iy^2, Ixy = IyIx$, empty list cornerList and
$height, width = img.shape$ and Initialize zero matrix rMat(height, width)
**for** $y \leftarrow m : height - m$ **do**
    **for** $x \leftarrow m, width - m$ **do**
        $Sxx \leftarrow (Ixx(y-m : y+m+1, x-m : x+m+1)).sum()$
        $Sxy \leftarrow (Ixy(y-m : y+m+1, x-m : x+m+1)).sum()$
        $Syy \leftarrow (Iyy(y-m : y+m+1, x-m : x+m+1)).sum()$
        $det \leftarrow (Sxx * Syy) - (Sxy^2)$
        $trace = Sxx + Syy$
        $r = det - k(trace^2)$
        $rMat(y, x) = r$
    **end for**
**end for**
$T \leftarrow thresh * rMat.max$
**for** each pixel (i,j) in img **do**
    **if** $rMat(i, j) > T$ **then**
        $cornerList.append(rMat(j, i), j, i)$
    **end if**
**end for**
**return** cornerList



(a) Luminence Image (b) Corners

Figure 2: Corner Detected before applying NMS on Luminence Image, m = 4 k = 0.04

## 2.4 Non-Maximal Suppression

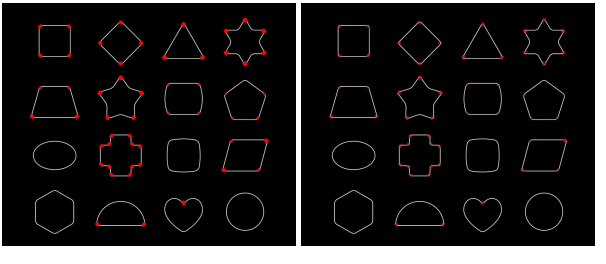
Now we need to reduce the cluster of corner points in an area to 1 point, where the the r value is the maximum. For this we sort the entire list in descending order and traverse the list. We mark all the visited corners in the list. For each unvisited pixel, we append the value into another list and mark the pixel as well as its neighbouring pixels as visited, if they are present in the list. We finally return the list that we made to store the unvisited corners as Non-maximal supressed corners.

We use the neighbourhood in this implementation to be within a distance of 3 pixels of the pixel in consideration.

After this step we have the final list of corner points, we simply add those points as in the image as some symbols. In this implementation we have used a red plus sign for corners.

**Algorithm 4** nonMaximalSupressionCorners(img)

Initailize empty list finalCornerList
$sortedInd \leftarrow argsort((cornerList)[:,0])$
**for** $i \leftarrow 0 : len(sortedInd)$ **do**
  **if** $cornerList(sortedInd(i))(1) != -1$ **then**
    $y \leftarrow cornerList(sortedInd(i))(1)$
    $x \leftarrow cornerList(sortedInd(i))(2)$
    $finalCornerList.append(cornerList(sortedInd(i))(0), j, i)$
    $cornerList(sortedInd(i))(1) \leftarrow -1$
    $cornerList(sortedInd(i))(2) \leftarrow -1$
    **for** each index j of neighbouring pixel of (x,y) present in cornerList **do**
      $cornerList(j)(1) \leftarrow -1$
      $cornerList(j)(2) \leftarrow -1$
    **end for**
  **end if**
**end for**
**return** finalCornerList



(a) Corners Before NMS (b) Harris Corner Detected

Figure 3: Harris Corner Detected Image, m = 4 k = 0.04

# 3 Observations

For the toy image, circles show no corner detected, and as threshold increases, lesser the number of corners are detected. Upon changing the neighbourhood limit in the Non-maximal suppression we notice a pretty big change in the final output. There is this trade-off between noise and more detailed edges. The plane figure is a very good example of the same. In the images involving humans or animals, the harris-corner detector gives a a lot of points near eyes and other facial features. The bicycle gradient angle image accurately shows the illumiation with respect to gradient angle on the wheels and explains the angle to intensity. Moreover, most Harris corner point detectors use only the grayscale information of an image. However, the color information of an image is wasted. To employ the color information, a new Harris corner point detector method was proposed. In this method, Harris corner detector is applied both to the grayscale image using gray level intensity information and to the color image using the RGB information. After corner points are detected in both the grayscale image and the color image, cross correlation and Random sample consensus are used to find matching corner points.

# 4 Inferences and takeaways

- Most Harris corner point detectors use only the grayscale information of an image. However, the color information of an image is wasted.
- We can see an inverse correlation between corner angles and the threshold limit.
- Higher the threshold, less corners are detected (but higher chance of being a corner)
- Lesser the threshold, more corners are detected (but higher chance that a non-corner may be identified as a corner).
- Harris detector is most repetitive and most informative.
- We can change the neighbourhood limit in the non-maximal suppression if we mean to identify say fine edges or only the larger, more prominent edges. We can hence use this parameter to fine-tune for various purposes in edge detection like for example to get only a high level outline of the object in an image, instead of the roughness of the edges in the object.
- The disadvantage of this detector is it is not invariant to large scale change.
- provides good repeatability under varying rotation and illumination.
- Detectors like Harris-corner detector detect a lot of facial features and could have an application in face detection.
- The Harris corner detector is invariant to translation, rotation and illumination change.

# References

[1] Chris Harris Mike Stephens. A combined corner and edge detector. 1988.

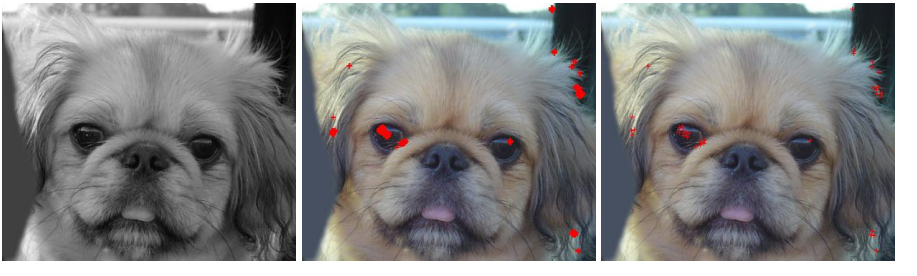
# 5 Results

(a) Luminence Image (b) Threshold Image (c) Harris Edge Detected

Figure 4: All Intermediate and Final Images through Harris Edge Detector of Bicycle, m = 3 k = 0.04

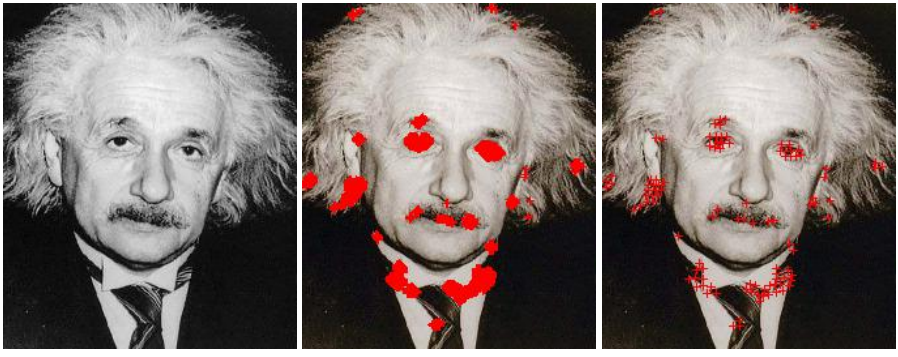
(a) Luminence Image (b) Threshold Image (c) Harris Edge Detected

Figure 5: All Intermediate and Final Images through Harris Edge Detector of Bird, m = 3 k = 0.04



(a) Luminence Image (b) Threshold Image (c) Harris Edge Detected

Figure 6: All Intermediate and Final Images through Harris Edge Detector of Dog, m = 3 k = 0.04

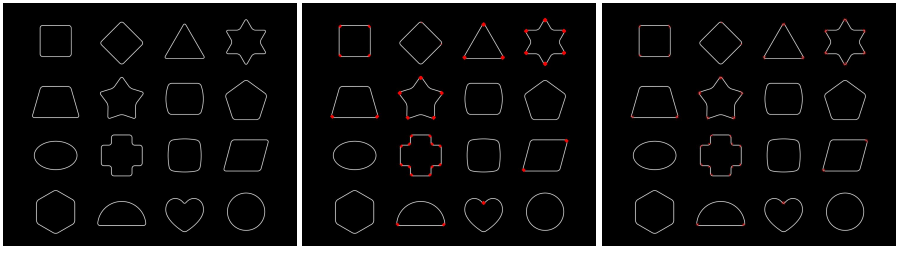


(a) Luminence Image (b) Threshold Image (c) Harris Edge Detected

Figure 7: All Intermediate and Final Images through Harris Edge Detector of Einstein, m = 3 k = 0.04

(a) Luminence Image (b) Threshold Image (c) Harris Edge Detected

Figure 8: All Intermediate and Final Images through Harris Edge Detector of Plane, m = 3 k = 0.04



(a) Luminence Image (b) Threshold Image (c) Harris Edge Detected

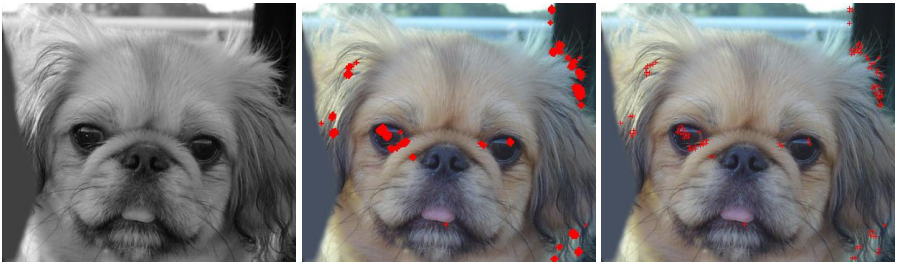
Figure 9: All Intermediate and Final Images through Harris Edge Detector of Toy, m = 3 k = 0.04

(a) Luminence Image (b) Threshold Image (c) Harris Edge Detected

Figure 10: All Intermediate and Final Images through Harris Edge Detector of Bicycle, m = 3 k = 0.05

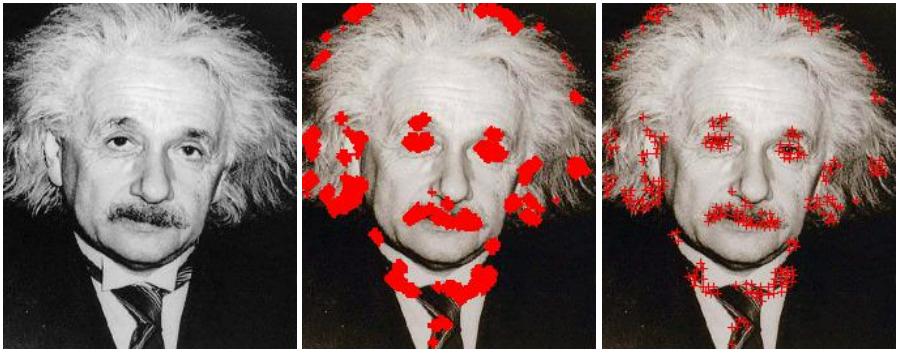
(a) Luminence Image (b) Threshold Image (c) Harris Edge Detected

Figure 11: All Intermediate and Final Images through Harris Edge Detector of Bird, m = 3 k = 0.05



(a) Luminence Image (b) Threshold Image (c) Harris Edge Detected

Figure 12: All Intermediate and Final Images through Harris Edge Detector of Dog, m = 3 k = 0.05

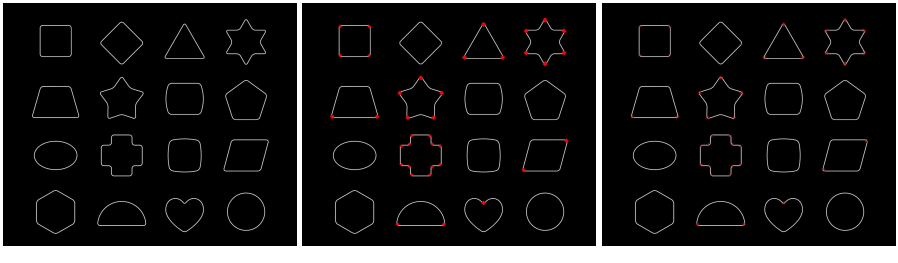


(a) Luminence Image (b) Threshold Image (c) Harris Edge Detected

Figure 13: All Intermediate and Final Images through Harris Edge Detector of Einstein, m = 3 k = 0.05

(a) Luminence Image (b) Threshold Image (c) Harris Edge Detected

Figure 14: All Intermediate and Final Images through Harris Edge Detector of Plane, m = 3 k = 0.05



(a) Luminence Image (b) Threshold Image (c) Harris Edge Detected

Figure 15: All Intermediate and Final Images through Harris Edge Detector of Toy, m = 3 k = 0.05

(a) Luminence Image (b) Threshold Image (c) Harris Edge Detected

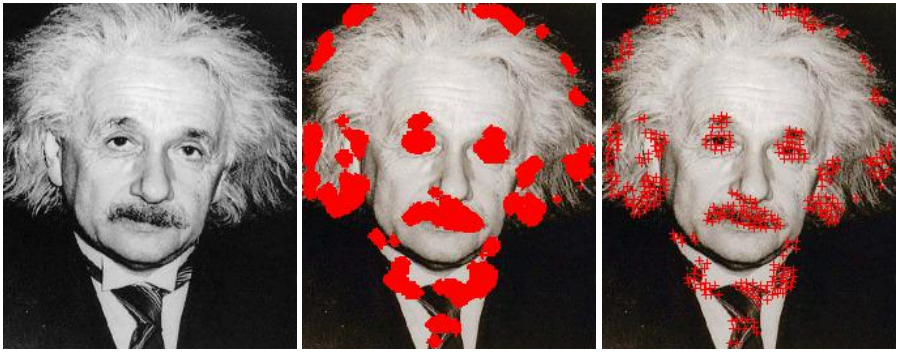
Figure 16: All Intermediate and Final Images through Harris Edge Detector of Bicycle, m = 4 k = 0.04

(a) Luminence Image (b) Threshold Image (c) Harris Edge Detected

Figure 17: All Intermediate and Final Images through Harris Edge Detector of Bird, m = 4 k = 0.04

(a) Luminence Image (b) Threshold Image (c) Harris Edge Detected

Figure 18: All Intermediate and Final Images through Harris Edge Detector of Dog, m = 4 k = 0.04

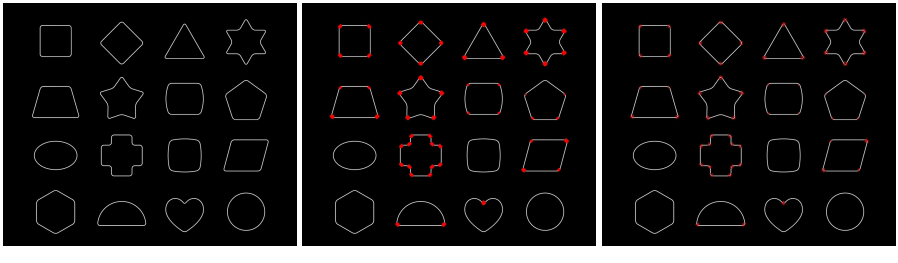


(a) Luminence Image (b) Threshold Image (c) Harris Edge Detected

Figure 19: All Intermediate and Final Images through Harris Edge Detector of Einstein, m = 4 k = 0.04

(a) Luminence Image (b) Threshold Image (c) Harris Edge Detected

Figure 20: All Intermediate and Final Images through Harris Edge Detector of Plane, m = 4 k = 0.04



(a) Luminence Image (b) Threshold Image (c) Harris Edge Detected

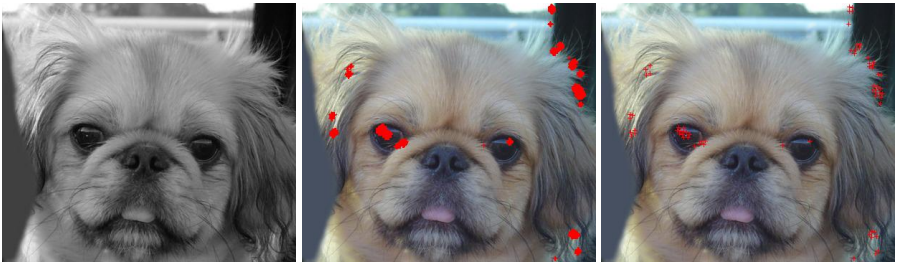
Figure 21: All Intermediate and Final Images through Harris Edge Detector of Toy, m = 4 k = 0.04

(a) Luminence Image (b) Threshold Image (c) Harris Edge Detected

Figure 22: All Intermediate and Final Images through Harris Edge Detector of Bicycle, m = 4 k = 0.05

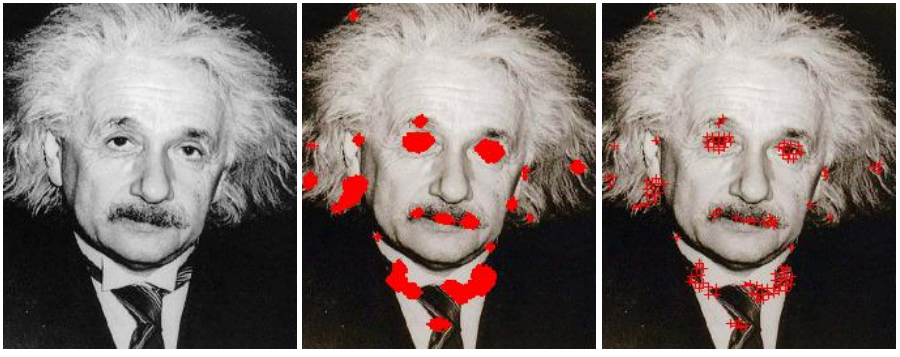
(a) Luminence Image (b) Threshold Image (c) Harris Edge Detected

Figure 23: All Intermediate and Final Images through Harris Edge Detector of Bird, m = 4 k = 0.05



(a) Luminence Image (b) Threshold Image (c) Harris Edge Detected

Figure 24: All Intermediate and Final Images through Harris Edge Detector of Dog, m = 4 k = 0.05

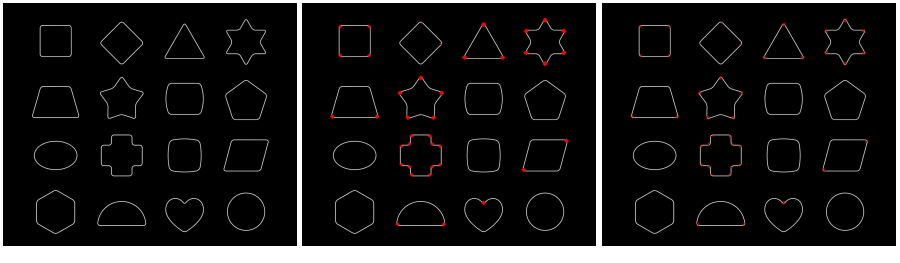


(a) Luminence Image (b) Threshold Image (c) Harris Edge Detected

Figure 25: All Intermediate and Final Images through Harris Edge Detector of Einstein, m = 4 k = 0.05

(a) Luminence Image (b) Threshold Image (c) Harris Edge Detected

Figure 26: All Intermediate and Final Images through Harris Edge Detector of Plane, m = 4 k = 0.05



(a) Luminence Image (b) Threshold Image (c) Harris Edge Detected

Figure 27: All Intermediate and Final Images through Harris Edge Detector of Toy, m = 4 k = 0.05