Harris Corner detection Detection

Kamal Sharma
Entry_no.:2017csb1084
Indian Institute of Technology, Ropar
Punjab, India

Abstract

Harris Corner Detector is the base of many other fast and efficient corner detection algorithms. It is used in a range of applications. Here in this report shows the implementation of Harris corner detection algorithm.

1 Introduction

Corners are unique and do not suffer from aperture problem, that may afflict edges. Corner is a position where a slight shift in position will lead to a large change in intensity in both horizontal(X) and vertical(Y) directions. Although corner are in very less amount with respect to image, but still used in applications like motion detecting and object recognition.
It was first introduced by Chris Harris and Mike Stephens in 1988 upon the improvement of Moravec's corner detector. [2].

2 Methodology and Algorithms

First I calculated the gradient of image by applying (sobel or differention of gauss).
Fiters used are:

$for\ x-direction$ $\quad$ $for\ y-direction$

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

We have to maximize $\sum_{x,y}[I(x+u,y+v)-I(x,y)]^2$ [1].

As, E(u,v) = $\sum_{x,y} w(x,y)[I(x+u,y+v)-I(x,y)]^2$

E(u,v) can be written in matrix form as.

$$\text{E(u,v)} \approx \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix}$$

$$\text{where M} = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix}$$

$\text{R} = det(M) - k(trace(M))^2$
Here is its implementation . See Algorithm1 harris corner detection.

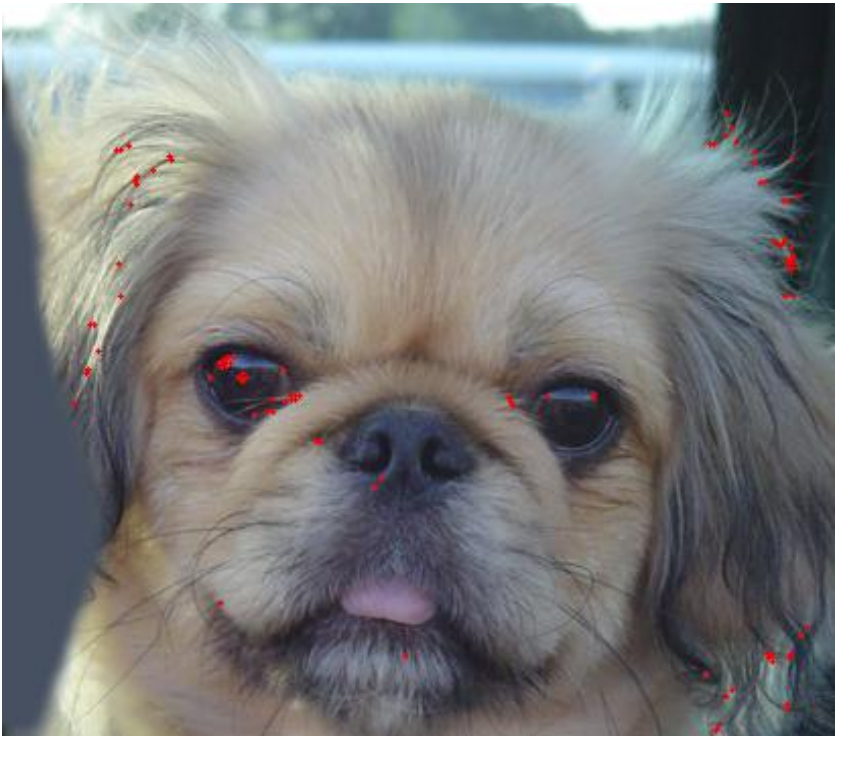
Algorithm 1 harris corner detection

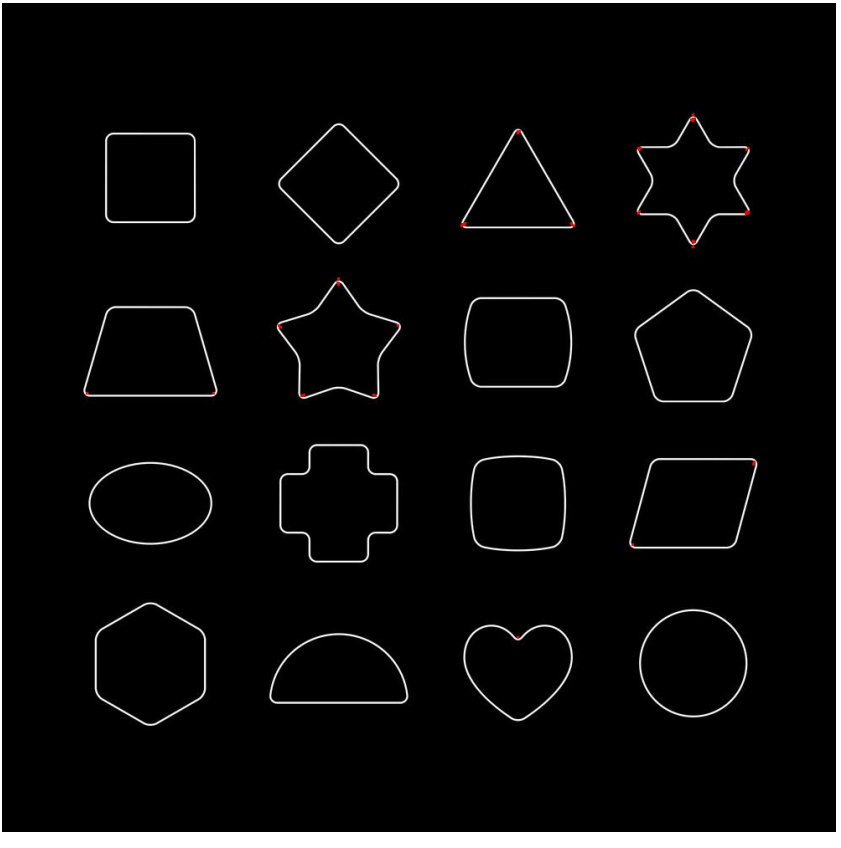
Input: *img_cpy*, *gray_img*, *Harris_const*, *window*
Output: *img_cpy*

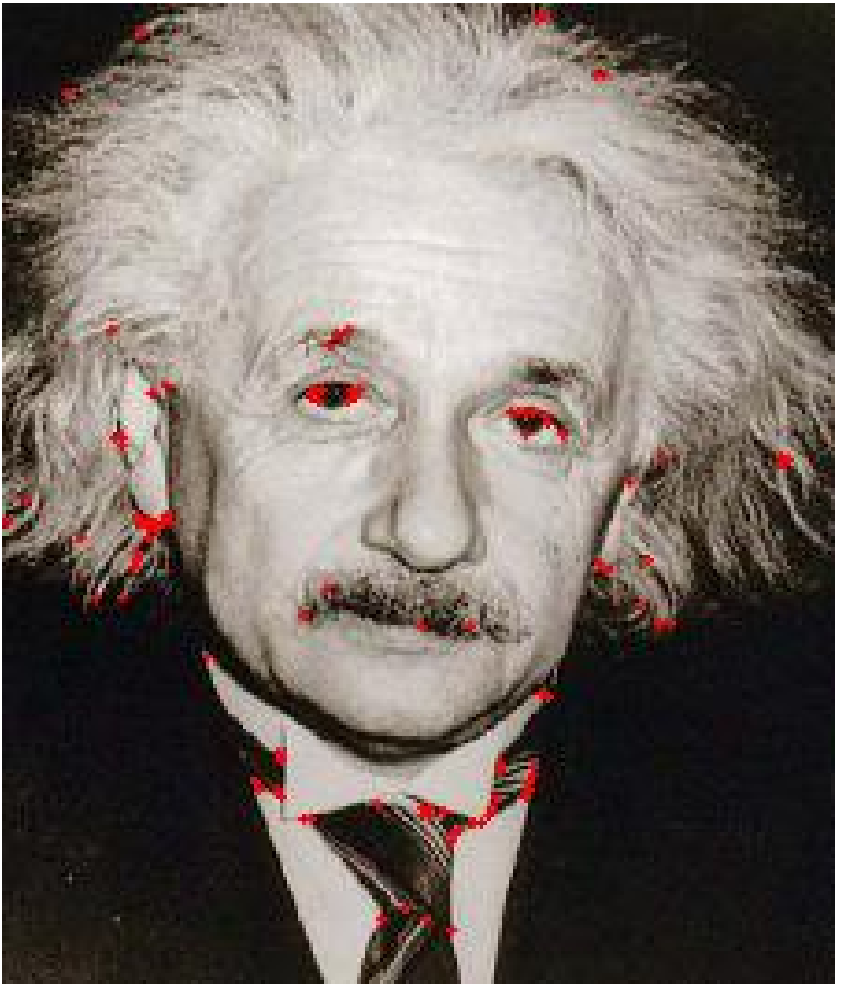
Fx, Fy $\leftarrow$ *apply gradient on gray_img*
Fxx, Fyy, Fxy $\leftarrow$ Fx^2, Fy^2, $Fx*Fy$
height, *width* $\leftarrow$ *img_cpy.shape*
$offset$ $\leftarrow$ $int(window/2)$
for $y \leftarrow offset$ to *height-offset* **do**
    for $x \leftarrow offset$ to *width-offset* **do**
        $Sxx \leftarrow \sum(Fxx[y-offset : y+offset+1, x-offset : x+offset+1])$
        $Syy \leftarrow \sum(Fyy[y-offset : y+offset+1, x-offset : x+offset+1])$
        $Sxy \leftarrow \sum(Fxy[y-offset : y+offset+1, x-offset : x+offset+1])$
        $det \leftarrow (Sxx * Syy) - (Sxy^2)$
        $trace \leftarrow Sxx + Syy$
        $r \leftarrow det - Harris\_const * (trace^2)$
        if r > Th **then**
            cornerList.append$([x, y, r])$
        end if
    end for
end for
sort cornerList with respect to r in decreasing order
suppress 8 *neighbour pixels if thier r value is less value of r.*
for *top* 100 *corner pixels* **do**
    $img\_cpy[Y1, X1] \leftarrow [255, 0, 0]$
end for

3 Results and Observations

- Generally, Harris constant belongs to [0.04,0.06] for all images. Lowering the threshold value increases the number of corners detected and also increases code's time complexity.
- In order to pick up the optimal values to indicate corners, we find the local maxima as corners within the window which is a 3 by 3 filter.
- Intersection of blur edges are not showing corners.
- A large value of R indicates a corner, a negative value indicates an edge.
- This works only for sharp corners, as in toy_harris image triangle and star image shows corners.







4 Conclusion and Takeaway

- Works best on binary image in comparison to coloured images.
- Lowering the threshold increses number of corners as well as time complexity of algorithm.

References

[1] openCV. Harris corner detection. https://docs.opencv.org/3.4/d4/d7d/tutorial_harris_detector

[2] Wikipedia. Harris corner detection. https://en.wikipedia.org/wiki/Harris_Corner_Detector.