HARRIS CORNER DETECTOR

AMIT SRIVASTAVA
2017CSB1189

Indian Institue of Technology
Ropar - Punjab,
India

Abstract

Harris Corner Detector is a corner detection operator that is commonly used in computer vision algorithms to extract corners and infer features of an image. In this work, we present an implementation of the Harris corner detector. This feature detector relies on the analysis of the eigenvalues of the autocorrelation matrix. The algorithm comprises several steps, including several measures for the classification of corners, a generic non-maximum suppression method for selecting interest points, and the possibility to obtain the corners position with subpixel accuracy.

A corner is a point whose local neighborhood stands in two dominant and different edge directions. In other words, a corner can be interpreted as the junction of two edges, where an edge is a sudden change in image brightness. Corners are the important features in the image, and they are generally termed as interest points which are invariant to translation, rotation, and illumination.

The Harris Corner Detector algorithm in simple words is as follows :

STEP 1. It determines which windows (small image patches) produce very large variations in intensity when moved in both X and Y directions (i.e. gradients).

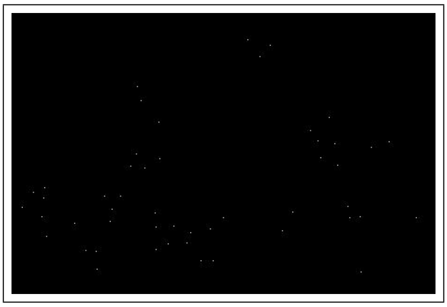
STEP 2. With each such window found, a score R is computed.

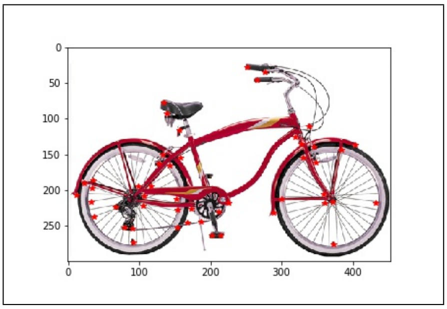
STEP 3. After applying a threshold to this score, important corners are selected and marked.

Take the gray-scale of the original image. Apply a Gaussian filter to smooth out any noise. Apply Sobel operator to find the x and y gradient values for every pixel in the grayscale image. For each pixel p in the grayscale image, consider a m*m window around it and compute the corner strength function. Call this its Harris value. Find all pixels that exceed a certain threshold and are the local maxima within a certain window (to prevent redundant dupes of features). Compute a feature descriptor of all such points.

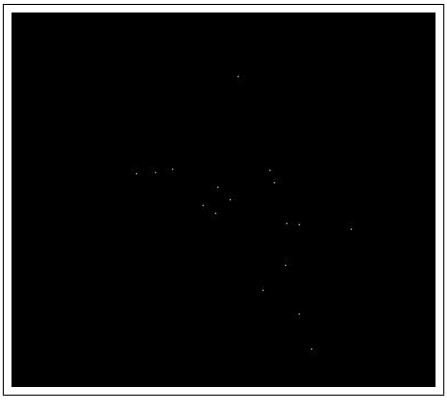
Output and Intermediate Image format :
1. Corner Interest Points
2. Corner Interest Points overlayed on Original Image

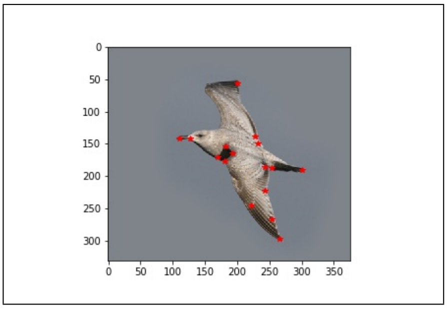
1 Intermediate and Final Images of Bicycle



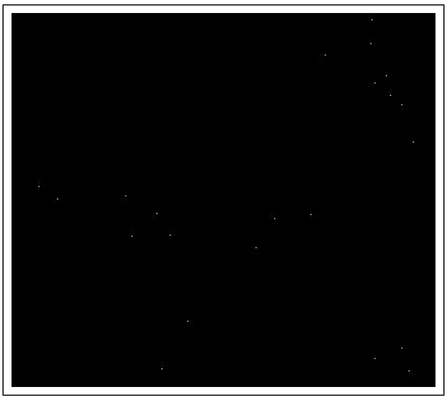


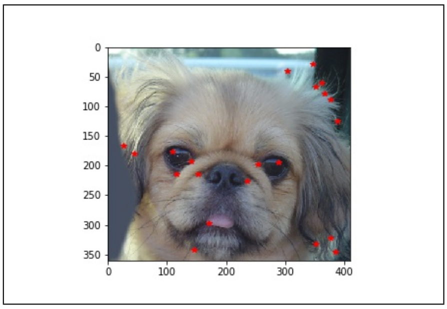
2 Intermediate and Final Images of Bird



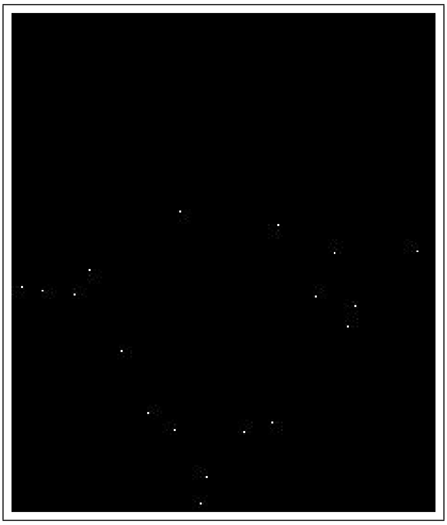


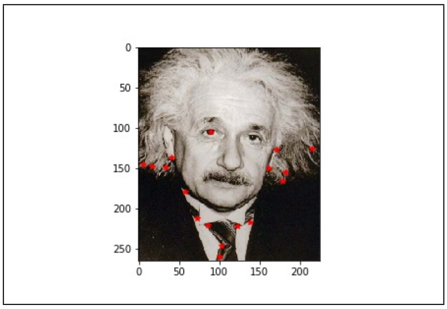
3 Intermediate and Final Images of Dog



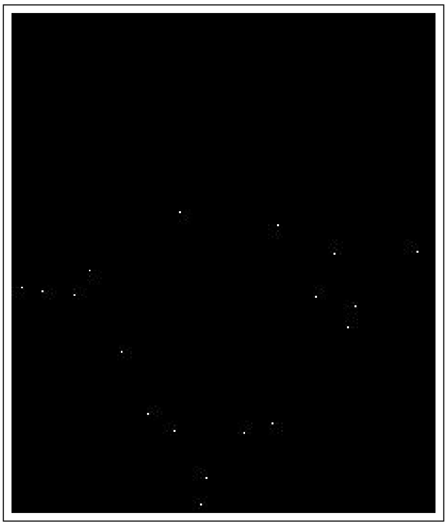


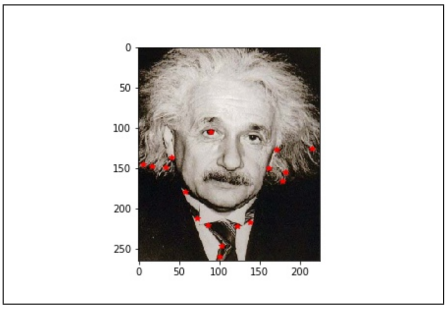
4 Intermediate and Final Images of Einstein



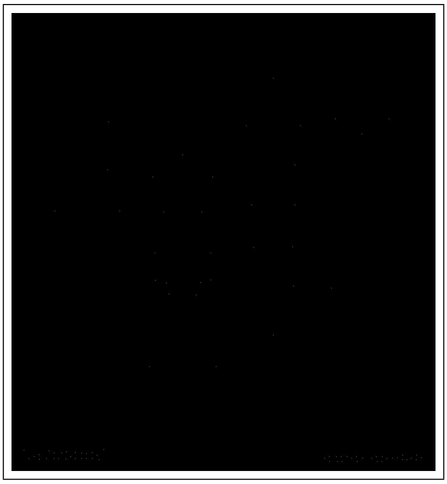


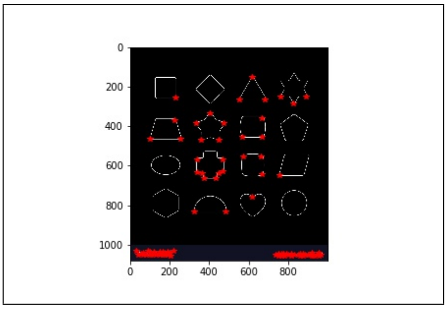
5 Intermediate and Final Images of Toy Image





6 Intermediate and Final Images of Toy Image





7 Algorithms

Algorithm 1 find-corners(input-img)

```
1: Convert input image to gray-scale and then apply Gaussian Blur
2:
3: Find gradient in x and y direction using sobel filter
4: $x - grad = gradient\, in\, x\, direction$
5: $y - grad = gradient\, in\, y\, direction$
6: $xx - grad = x - grad * x - grad$
7: $yy - grad = y - grad * y - grad$
8: $xy - grad = x - grad * y - grad$
9:
10: $k = 0.04$
11: tuple-data contains y, x Co-ordinates and its corner response
12:
13: while $i < height$ do
14:     while $j < width$ do
15:         In window of size 9 * 9 around i, j find determinant and trace
16:         R = determinant - (k * trace * trace)
17:         Add this i,j,R to tuple-data
18:     end while
19: end while
20:
21: L contains y, x co-ordinate(whose value is greater than threshold) and their corner response of those co-ordinates
22: threshold = thres-ratio * max(R)
23:
24: while $res < tuple - data.length$ do
25:     i, j, R = res
26:     if $R > threshold$ then
27:         L.append([i, j, R])
28:     end if
29: end while
30: return L
```

Algorithm 2 Non-Maximal-Suppressor(L)

```
1: sorted-L contains L sorted in decresing order of R value
2: final-L contains list after non maximal suppression
3: Insert first element of sorted-L in final-L
4:
5: while $i\ in\ sorted-L.length$ do
6:     while $j\ in\ final-L.length$ do
7:         if $(abs(i[0]-j[0] <= dis)\ and\ abs(i[1]-j[1]) <= dis)$ then
8:             break
9:         end if
10:    end while
11:    Include this co-ordinate in final-L
12: end while
13:
14: Mark these co-ordinates : Corners
15: Overlay these co-ordinates with input image for representing better distinctive feature
```

8 Observation and Take-away/Conclusion

In this work, we presented an implementation of the Harris corner detector. We implemented a generic non-maximum suppression algorithm that allows to select the prominent features on the image. To improve the speed of the method, it is necessary to implement a faster Gaussian convolution technique, or to replace it with a box filter, at the expense of an accuracy loss.

The output heavily depends on the threshold value. So inspite of choosing threshold value, I chose threshold ratio. Threshold value equals to threshold ratio times maximum of R value for the given image. Best threshold ratio for various images are :

Bicycle : 0.05
Bird : 0.05
Dog : 0.05
Einstein : 0.05
Plane : 0.01
Toy-Image : 0.01

Final Output also depends on the distance value. Also it depend on the size of matrix. For 3*3 matrix, it was not so good and contained some non-corner points too. But for 9*9 matrix output was accurate enough to in-built harris detector.

The detector algorithm implemented relies on the analysis of the eigenvalues of the autocorrelation matrix. The algorithm comprises several steps, including several measures for the classification of corners, a generic non-maximum suppression method for selecting interest points, and the possibility to obtain the corners position with subpixel accuracy.