Canny Edge and Harris Corner Detector

Akansha Singh
2017csb1065@iitrpr.ac.in

Indian Institute
of Technology Ropar
Punjab, India

1 Canny Edge Detection

1.1 Introduction

1. Apply Gaussian Blur Filter after converting the image to gray-scale.
2. Get x and y component of the image gradient after smoothing with the gaussian.
3. At each pixel compute the edge strength (gradient magnitude) and the edge orientation.
4. Apply non-maximal suppression.
5. Then perform hysteresis thresholding by varying two thresholds T-low and T-high.

1.2 Methodology

1.2.1 Gradient convolution is done using the following matrix

$$Gx = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} \quad Gy = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$$

1.2.2 Non Maximal Suppression

For non maximal suppression angle is first rounded off to 0°, 45°, 90°, 135°

```
def non_max_suppression(F, D)
   out = np.zeros(F.shape)
   for i in range(1, int(F.shape[0]) - 1):
      for j in range(1, int(F.shape[1]) - 1):
         if((D[i,j] >= -22.5 and D[i,j] <= 22.5) or (D[i,j] <=
             -157.5 and D[i,j] >= 157.5)):
            if((F[i,j] > F[i,j+1]) and (F[i,j] > F[i,j-1])):
               out[i,j] = F[i,j]
            else:
               out[i,j] = 0
         if((D[i,j] >= 22.5 and D[i,j] <= 67.5) or (D[i,j] <=
             -112.5 and D[i,j] >= -157.5)):
```

```
            if((F[i,j] > F[i+1,j+1]) and (F[i,j] > F[i-1,j-1])):
               out[i,j] = out[i,j]
            else:
               out[i,j] = 0
         if((D[i,j] >= 67.5 and D[i,j] <= 112.5) or (D[i,j] <=
             -67.5 and D[i,j] >= -112.5)):
            if((F[i,j] > F[i+1,j]) and (F[i,j] > F[i-1,j])):
               out[i,j] = F[i,j]
            else:
               out[i,j] = 0
         if((D[i,j] >= 112.5 and D[i,j] <= 157.5) or (D[i,j] <=
             -22.5 and D[i,j] >= -67.5)):
            if((F[i,j] > F[i+1,j-1]) and (F[i,j] > F[i-1,j+1])):
               out[i,j] = F[i,j]
            else:
               out[i,j] = 0

   return out
```

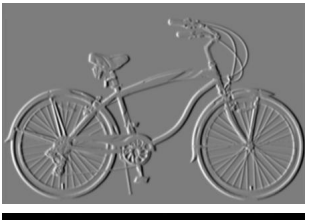
1.2.3 Hysteresis Thresholding

1. Two thresholds low and high are found, the values above high thresholds are considered as strong edge points.
2. The values in between low and high thresholds are considered as weak edges points.
3. The weak edges if surrounded by strong edge is considered as strong edge. This is done until all such weak edges are exhausted.

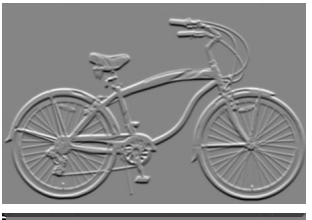
```
def HystThresh(img):
   highT = 0.40
   lowT = 0.30
   h = int(img.shape[0])
   w = int(img.shape[1])
   res = np.zeros(img.shape)
   out = np.copy(img)

   for i in range(0, h-1):
      for j in range(0, w-1):
         if(img[i][j]>=highT):
            res[i][j]=2
         elif(img[i][j]>=lowT):
            res[i][j]=1
         else:
            res[i][j]=0
   x = 0.1
   outx = 0

   while(outx != x):
      outx = x
```

```
    for i in range (1, h-1):
        for j in range(1, w-1):
            if(res[i][j]==1):
                if((res[i-1][j]==2) or (res[i-1][j+1]==2) or
                    (res[i-1][j-1]==2) or (res[i+1][j]==2) or
                    (res[i+1][j-1]==2) or (res[i+1][j+1]==2) or
                    (res[i][j-1]==2) or (res[i][j+1]==2)):
                   res[i][j]=2
    x = np.sum(res == 2)

res = (res == 2)*res

for i in range(0, h-1):
    for j in range(0, w-1):
        if(res[i][j]==2):
            out[i][j] = res[i][j]
        else:
            out[i][j] = 0

return out
```

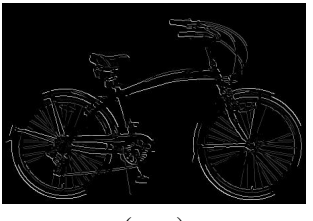
1.3 All images and other results



(gradient magnitude)

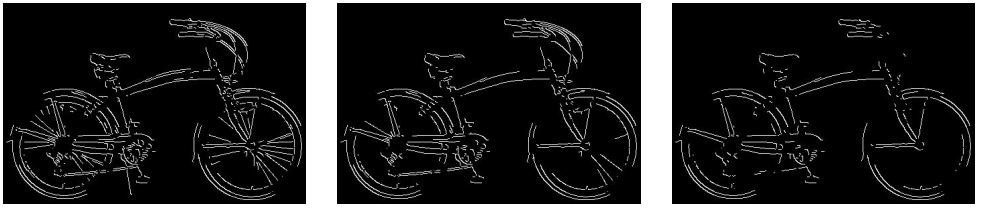


(orientation)



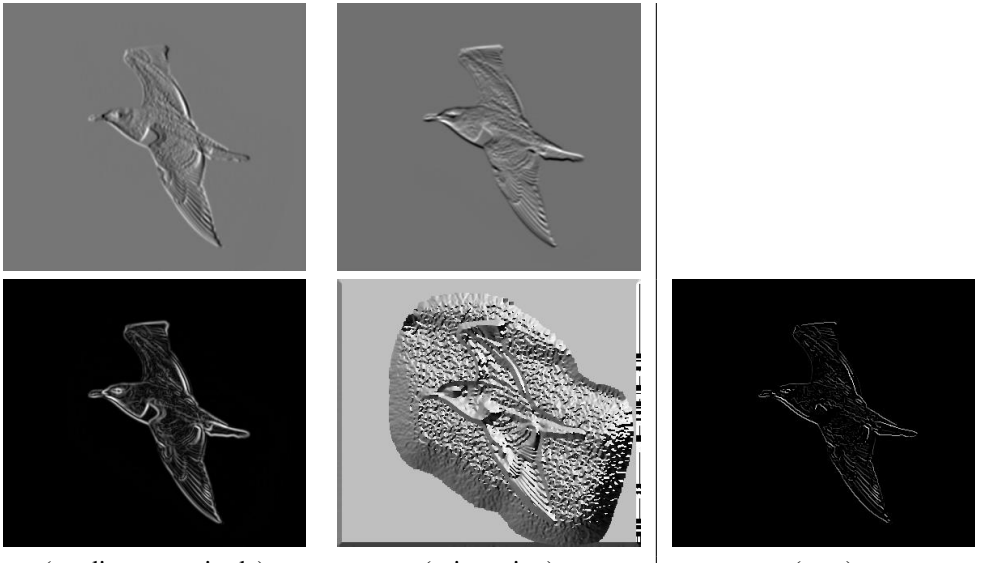
(nms)

Figure 1: Bicycle (First two images are the fx and fy components of the image)



(0.2-0.3) (0.3-0.4) (0.4-0.5)

Figure 2: Thresholds of Bicycle with 0.2-0.3 being the one human can sketch for.



(gradient magnitude) (orientation) (nms)

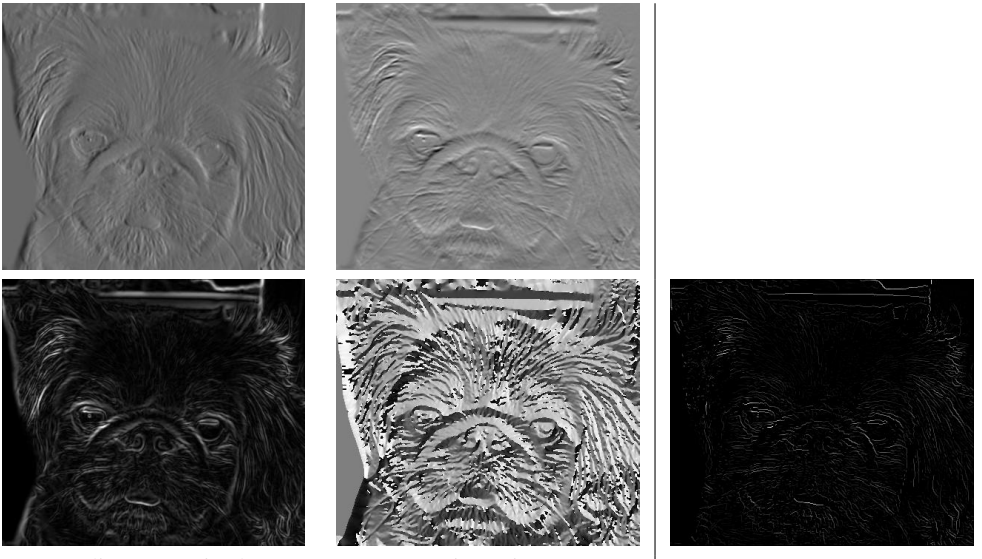
Figure 3: Bird (First two images are the fx and fy components of the image)

(0.1-0.3) (0.2-0.4) (0.1-0.5)

Figure 4: Thresholds of Bird with 0.2-0.4 being the one human can sketch for

1.4 Observations

On performing Non-Maximal Suppression following observations are made:



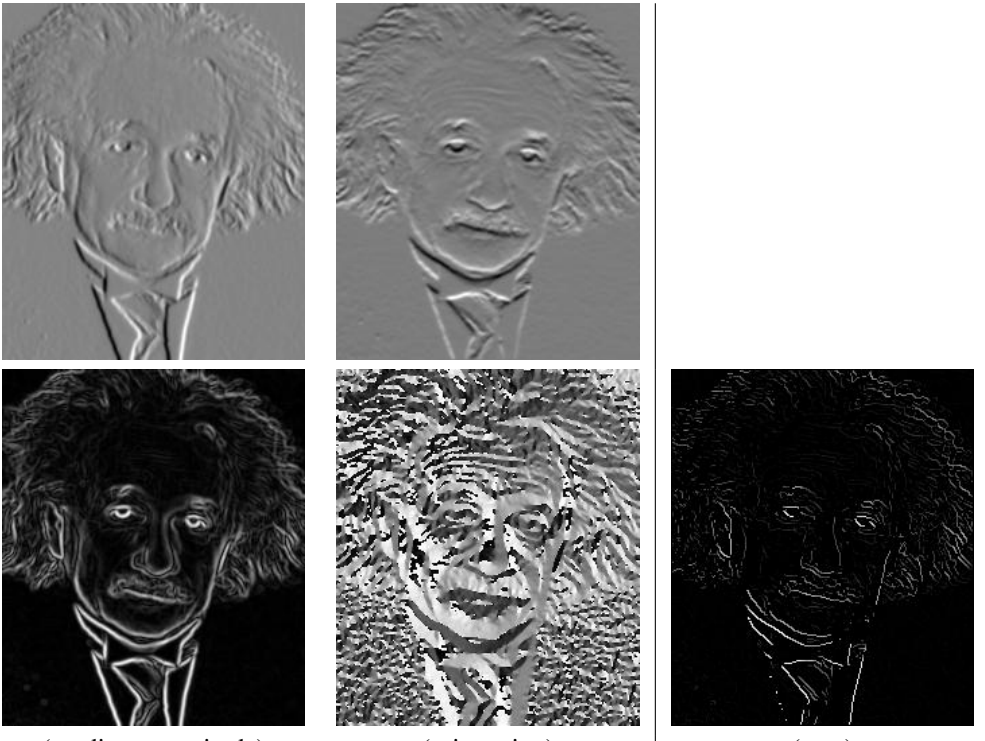
(gradient magnitude) (orientation) (nms)

Figure 5: Dog (First two images are the fx and fy components of the image)

(0.15-0.20) (0.20-0.25) (0.25-0.35)

Figure 6: Thresholds of Bird with 0.2-0.25 being the one human can sketch for

1. If the rounded gradient angle is 0° (i.e. the edge is in north-south direction) the point will be considered on the edge if its magnitude is greater than that of the pixels in east and west direction.
2. If the rounded gradient angle is 90° (i.e. the edge is in east-west direction) the point will be considered on the edge if its magnitude is greater than that of the pixels in the north and south direction.
3. If the rounded gradient angle is 45° (i.e. the edge is in southeast-northwest direction) the point will be considered on the edge if its magnitude is greater than that of the pixels in the northeast and southwest direction.
4. If the rounded gradient angle is 135° (i.e. the edge is in northeast-southwest direction) the point will be considered on the edge if its magnitude is greater than that of the pixels in the northwest and southeast direction.



(gradient magnitude) (orientation) (nms)

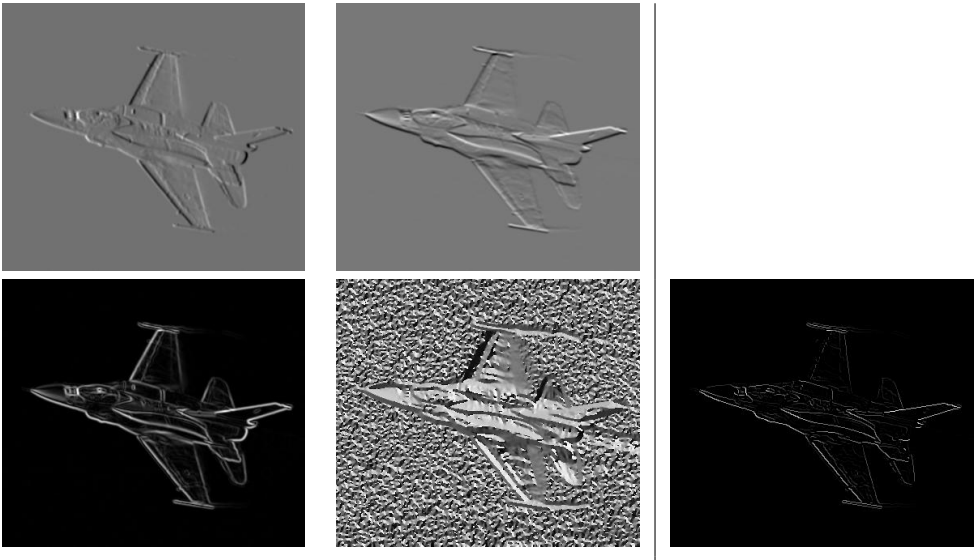
Figure 7: Einstein (First two images are the fx and fy components of the image)

(0.15-0.25) (0.25-0.35) (0.15-0.35)

Figure 8: Thresholds of Einstein with 0.25-0.35 being the one human can sketch for

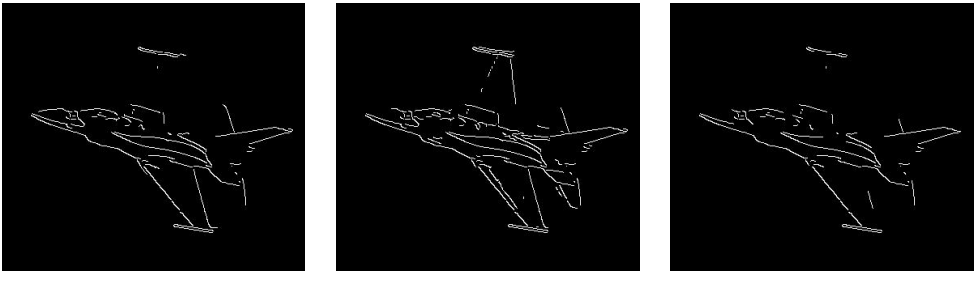
On performing Hysteresis Thresholding following observations are made:
1. On increasing the higher threshold value, some potential strong edges can be missed upon.

2. On increasing the higher threshold value, we allow lesser weak edges to be a part of final



(gradient magnitude) (orientation) (nms)

Figure 9: Plane (First two images are the fx and fy components of the image)



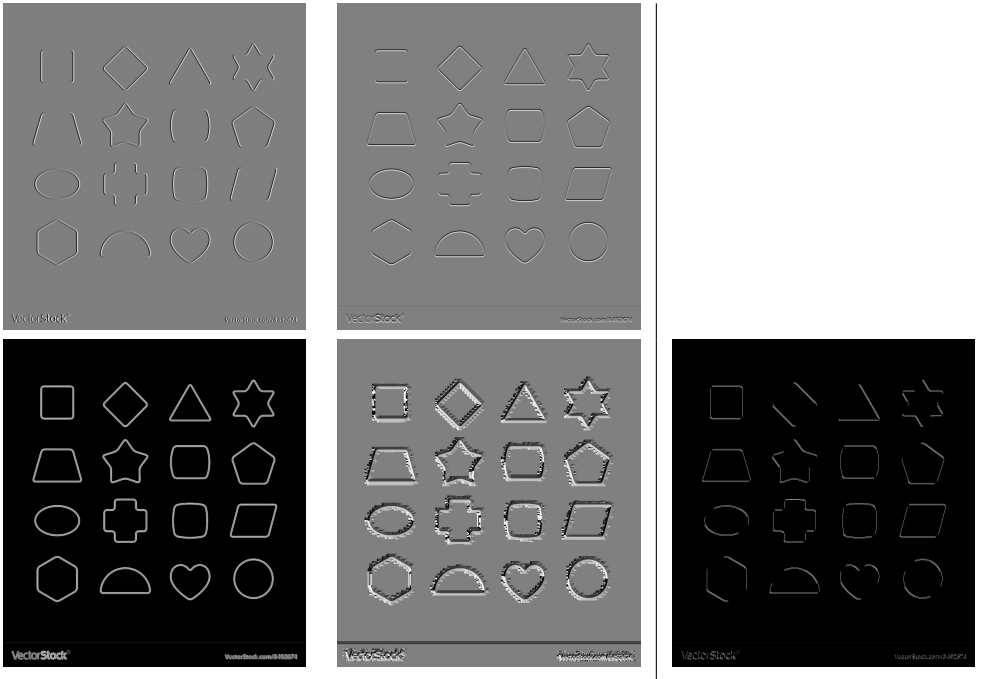
(0.10-0.35) (0.15-0.25) (0.25-0.35)

Figure 10: Thresholds of Plane with 0.15-0.25 being the one human can sketch for

edge image

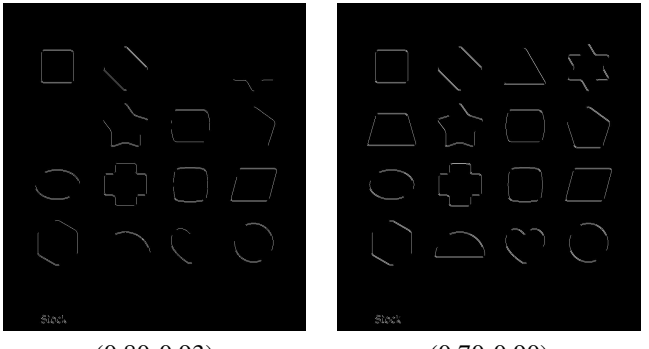
1.5 Inferences

1. The threshold values for images differ from image to image.
2. The threshold for images can be found out from clustering the edge pixels. From clustering the edge pixels we would be able to form clusters of same intensity edges (i.e. strong and weak kind of edges). It would become more efficient to predict low and high thresholds from these clusters.



(gradient magnitude) (orientation) (nms)

Figure 11: Toy-image (First two images are the fx and fy components of the image)



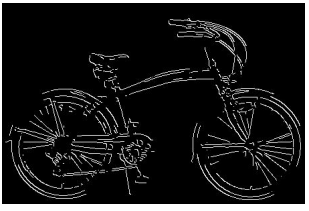
(0.80-0.93) (0.70-0.90)

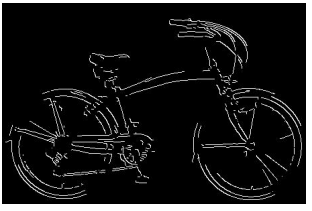
Figure 12: Thresholds of Toy-image with 0.70-0.90 being the one human can sketch for

2 Corner detector

2.1 Introduction

1. Compute image gradient in x and y.
2. Compute Ix.Ix Iy.Iy and Ix.Iy.
3. Apply Gaussian Blur.
4. Compute R = det(M) - alpha.trace(M)**2 each
5. Choose points with R above threshold.


(0.20-0.30)


(0.20-0.40)

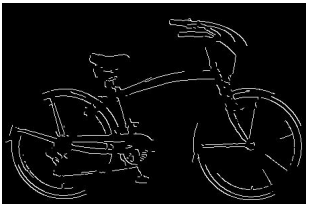
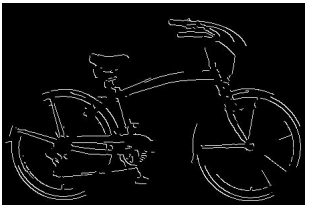
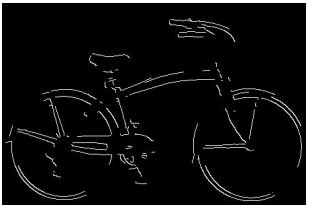

(0.20-0.50)

Figure 13: Bicycle Hysteresis thresholded image by increasing higher threshold


(0.20-0.50)


(0.30-0.50)

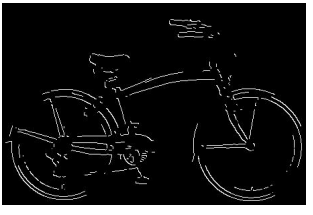

(0.40-0.50)

Figure 14: Bicycle Hysteresis thresholded image by increasing lower threshold

6. Compute Non-Maximal Suppression.

2.2 Methodology

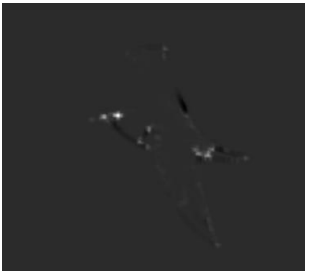
2.2.1 Find a corner

```
def corner( Fx, Fy, offset, k, th):
    fx2 = np.square(Fx)
    fy2 = np.square(Fy)
    fxfy = Fx*Fy

    fx2 = fx2*(1/np.square(2*offset+1))
    fy2 = fy2*(1/np.square(2*offset+1))
    fxfy = fxfy*(1/np.square(2*offset+1))

    harris = np.zeros_like(Fx)
    cornerList = []

    fx2 = pad(fx2, offset)
    fy2 = pad(fy2, offset)
    fxfy = pad(fxfy, offset)

    for y in range(offset, fx2.shape[0]-offset):
        for x in range(offset, fy2.shape[1]-offset):
            windowxx = fx2[y-offset:y+offset, x-offset:x+offset]
            windowyy = fy2[y-offset:y+offset, x-offset:x+offset]
```

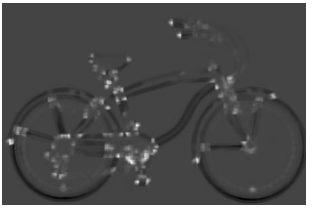
```
            windowxy = fxfy[y-offset:y+offset, x-offset:x+offset]

            sxx = windowxx.sum()
            syy = windowyy.sum()
            sxy = windowxy.sum()

            det = sxx*syy - sxy**2
            trace = sxx + syy
            c = det - k*(trace**2)

            harris[y-offset][x-offset] = c
            if c>th:
               cornerList.append([x-offset, y-offset, c])

    #print(cornerList)
    return harris, cornerList
```

2.2.2 Non Maximal Suppression using alpha = 0.04

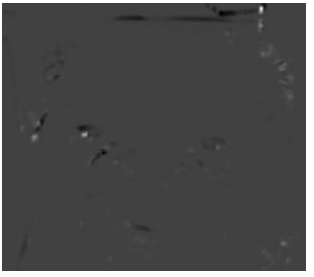
```
def nms(img, cornerList):
#corner list stores the values of corners having value greater than
    threshold
    cornerList.sort(key=itemgetter(2), reverse=True)
    corners = []
    output = np.copy(img)

    for i in range(len(cornerList)):
       x,y,e = cornerList[i]
       corners.append([x,y])

    cornersC = []

    for i in range(len(corners)):
       x,y = corners[i]
       cornersC.append([x,y])

    for i in range(len(corners)):
       x,y = corners[i]
       cornersC.append([x,y])

       if [x+1, y] in cornersC:
          cornersC.remove([x+1, y])
       if [x+1, y-1] in cornersC:
          cornersC.remove([x+1, y-1])
       if [x+1, y+1] in cornersC:
          cornersC.remove([x+1, y+1])
       if [x, y-1] in cornersC:
          cornersC.remove([x, y-1])
       if [x, y+1] in cornersC:
          cornersC.remove([x, y+1])
```



(bird)



(bicycle)



(dog)

Figure 15: Harris image.

```
    if [x-1, y] in cornersC:
        cornersC.remove([x-1, y])
    if [x-1, y-1] in cornersC:
        cornersC.remove([x-1, y-1])
    if [x-1, y+1] in cornersC:
        cornersC.remove([x-1, y+1])

for i in range(len(cornersC)):
    x, y = cornersC[i]
    output.itemset((y,x,0),255)
    output.itemset((y,x,1),0)
    output.itemset((y,x,2),0)

return output
```

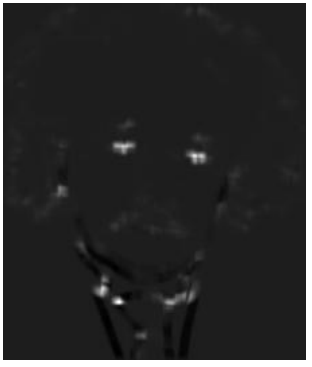
2.3 Observation

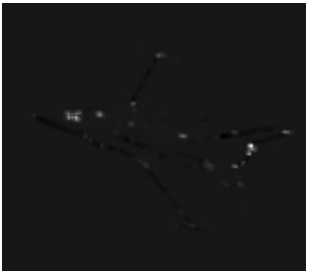
As threshold is decreased the detected interest points falsely report edges as on an edge one of Ix.Ix and Iy.Iy is huge which makes it lie above small values of threshold.

2.4 All images and results

2.5 Inferences

This algorithm requires tuning of threshold value. If the threshold value is considered to be high then very less number of corners will be detected while if it is kept low a lot of them would be detected (even it may not be a corner). One point to ponder over is that Non-maximal Suppression removes neighbourhood points and converges to a single point called local maxima.
Threshold value should be set so as to get between 10-100 corners.
Threshold value should be set such that we get points between 10 to 100. It can be used in Image Alignment Stitching, 3D Scene Modeling and reconstruction, Video Tracking, etc.


(einstein)


(plane)

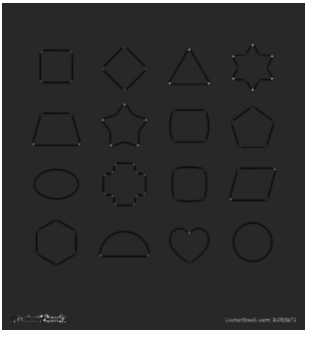

(toy image)

Figure 16: Harris image.


(0.003)


(0.004)


(0.005)

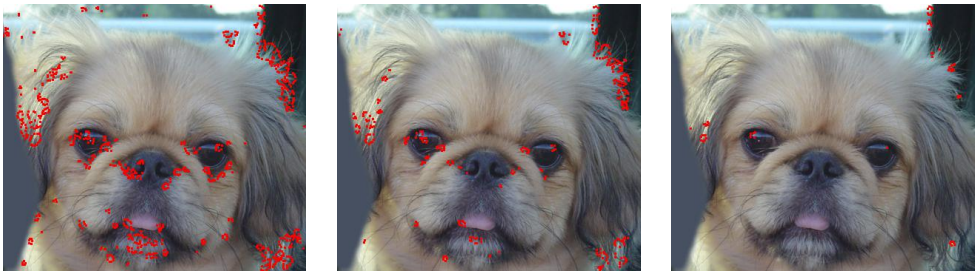
Figure 17: Bird corner detection using above thresholds with 0.004 being appropriate one.


(0.005)


(0.006)

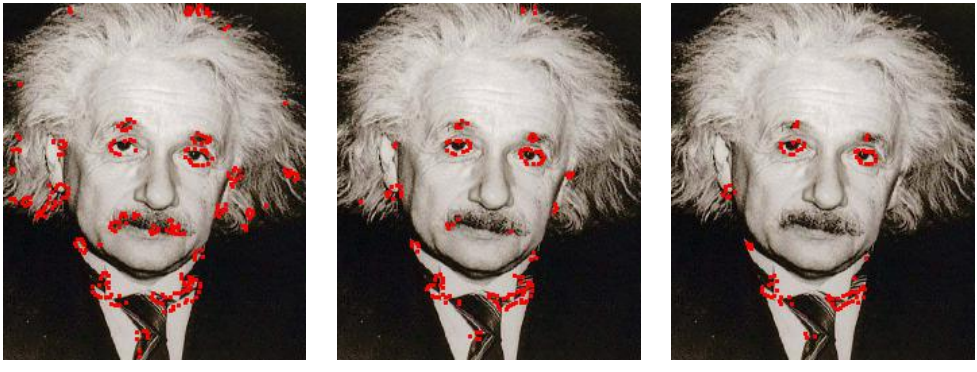

(0.007)

Figure 18: Bicycle corner detection using above thresholds with 0.006 being appropriate one.



(0.00009) (0.0002) (0.0008)

Figure 19: Dog corner detection using above thresholds with 0.0002 being appropriate one.

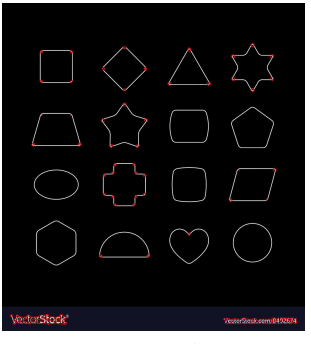


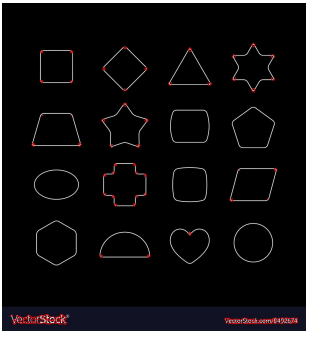
(0.0008) (0.0015) (0.0020)

Figure 20: Einstein corner detection using above thresholds with 0.0015 being appropriate one.

(0.00095) (0.001) (0.0020)

Figure 21: Einstein corner detection using above thresholds with 0.001 being appropriate one.


(0.00085)


(0.001)

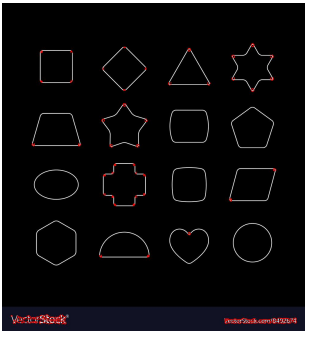

(0.0020)

Figure 22: Toy image corner detection using above thresholds with 0.001 being appropriate one.