CANNY EDGE DETECTOR

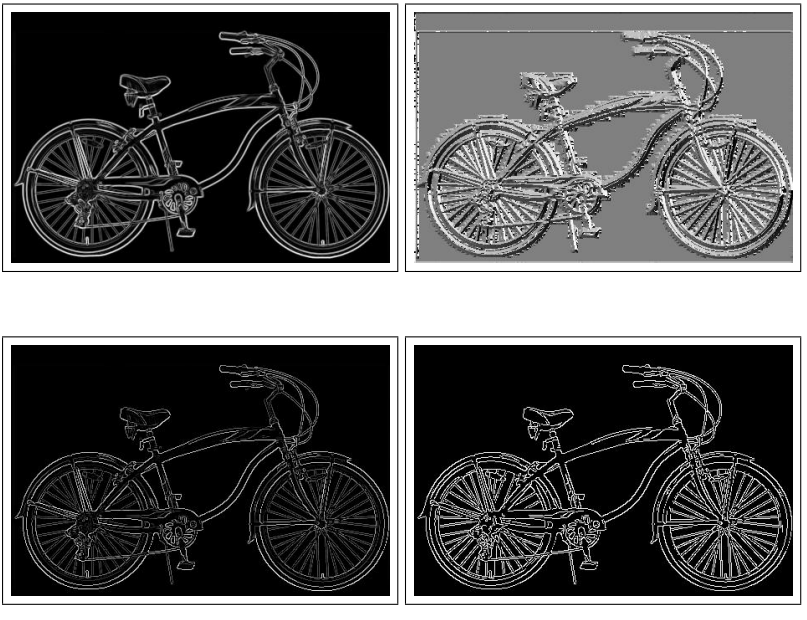
DEEPAK PANT
2017CSB1072
Indian Institue of Technology Ropar

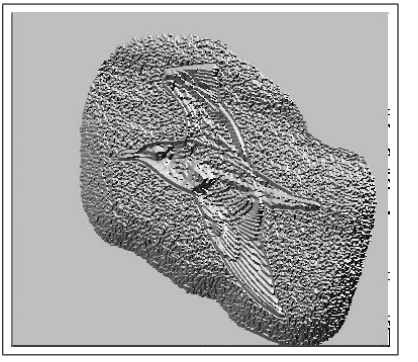
Abstract

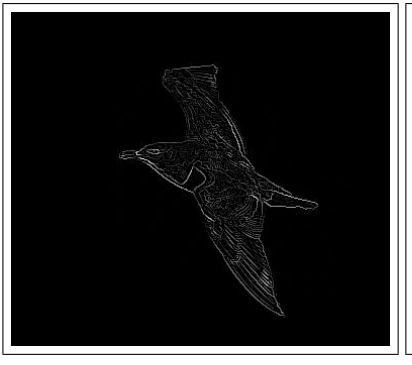
Canny edge detection is an algorithm to find edges in a image. Steps are taken to reduce noise. It involves computing smooth gradient of images, then applying non-maximal suppression to thin the images. To reduce noisy edges hystersis thresholding is applied.

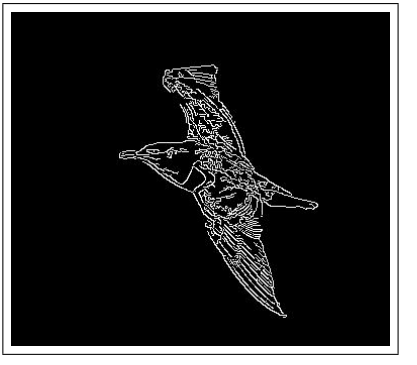
Firstly, I computed the luminance of given RGB image and then convolved it with sobel filter(which acts as a convolution of derivative operator with gaussian filter). Now the image has thick edges. Non-maximal suppression makes these edges one pixel thick. Edges which are less connected are then removed by hysteresis thresholding.

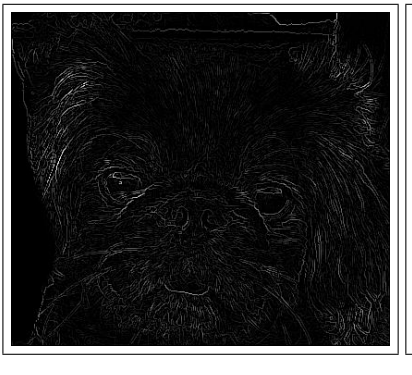
1 Output Images

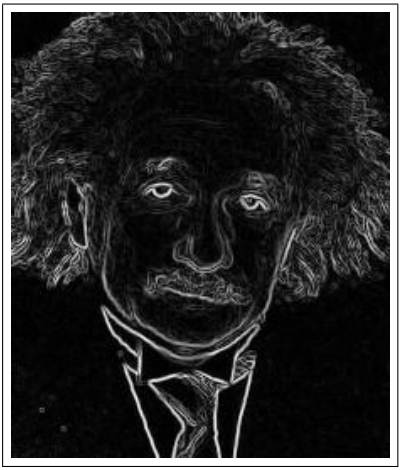


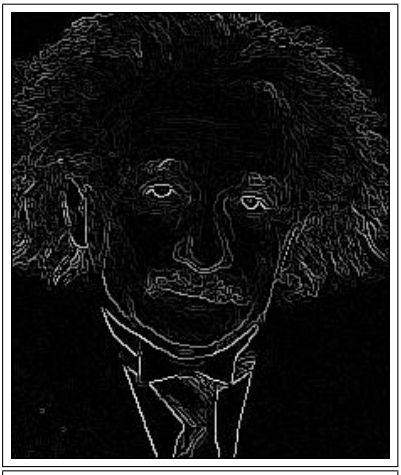


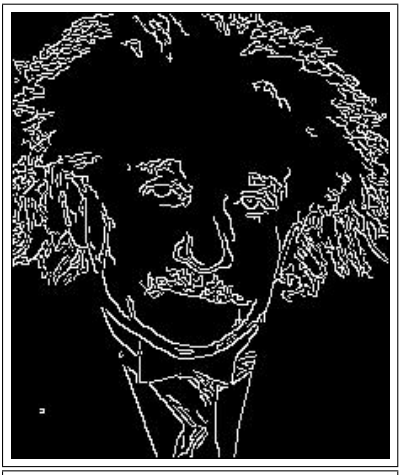


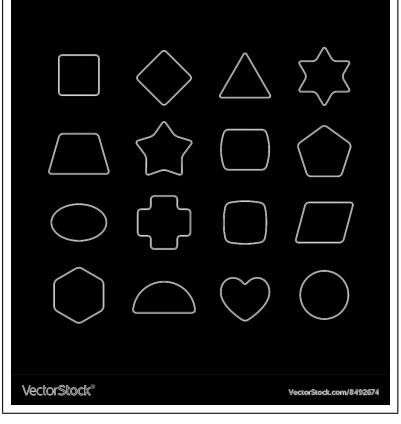


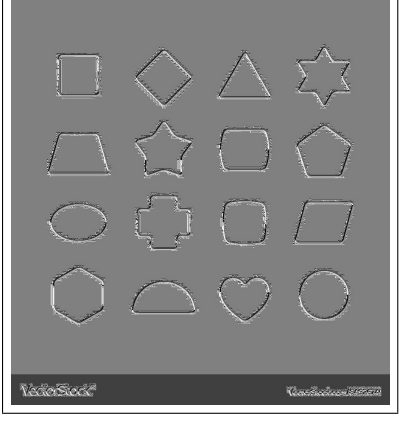


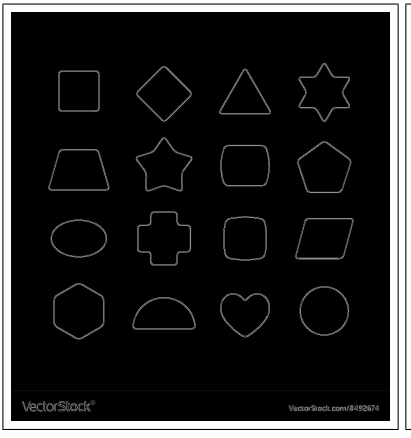


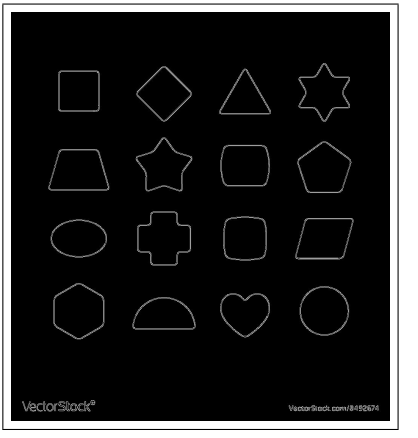





VectorStock
VectorStock.com/8492674


VectorStock


VectorStock®
VectorStock.com/8492674


VectorStock®
VectorStock.com/8492674

2 Algorithms

Algorithm 1 NonMaximalSuppressor(gradstr, graddir)

```
thinimage = np.zeros(gradstr.shape)

while i < height do
    while j < width do

        if closestdir[i, j] == 0 then
            if (gradstr[i, j] > gradstr[i, j+1]) and (gradstr[i, j] > gradstr[i, j−1]) then
                thinimage[i, j] = gradstr[i, j]
            else
                thinimage[i, j] = 0
            end if
        end if

        if closestdir[i, j] == 45 then
            if (gradstr[i, j] > gradstr[i + 1, j + 1]) and (gradstr[i, j] > gradstr[i1, j1]) then
                thinimage[i, j] = gradstr[i, j]
            else
                thinimage[i, j] = 0
            end if
        end if

        if closestdir[i, j] == 90 then
            if (gradstr[i, j] > gradstr[i+1, j]) and (gradstr[i, j] > gradstr[i−1, j]) then
                thinimage[i, j] = gradstr[i, j]
            else
                thinimage[i, j] = 0
            end if
        end if

        if closestdir[i, j] == 135 then
            if (gradstr[i, j] > gradstr[i + 1, j − 1]) and (gradstr[i, j] > gradstr[i − 1, j + 1]) then
                thinimage[i, j] = gradstr[i, j]
            else
                thinimage[i, j] = 0
            end if
        end if

    end while
end while
return thinimage
```

Algorithm 2 HysteresisThresholding(img)

```
lowr = 0.15
hir = 0.35
tlow = max(img)*lowr
thigh = max(img)*hir
hysimg = np.copy(img)

while i < height do
    while j < width do
        if img[i, j] > thigh then
            hysimg[i,j] = 1
        end if
        if img[i, j] < tlow then
            hysimg[i,j] = -1
        else
            hysimg[i,j] = 0
        end if

        Apply BFS to find those weak pixels that are 8-connected to strong pixels
        Classify these pixels as strong pixels
    end while
end while
return hysimg
```

3 Observations

Canny edge detector showed correct output only if it was given good threshold values. Each image had to be given different pair of thresholds as some had strong edges and other had weak edges. I found that we could keep a low and high ratios of the maximum value of a pixel intensity of the image. The ratio between high and low was around 2.5-3.5.

In hysteresis thresholding, I suppressed some edges with very low intensities. Of the remaining some were weak and some were strong. Those pixels which are 8-connected to the strong pixels are then classified strong. In the end I suppressed the remaining weak pixels. For traversal through the image I used BFS. I used a multi source BFS which starts at each strong pixels and then propagates outward till it cannot reclassify weak pixels.

The convolution function I used in this assignment was the one built in scipy. When I ran the same code on the same image, I found the convolve I had developed earlier was very slow compared to the one already present in library.

4 Conclusion

The canny edge detector showed good output in case of tuning the thresholds. For each image different pair of thresholds were required. The high ratio was found to be around 0.35 and the lower one around 0.10. Hysteresis thresholding removed unnecessary edges. This canny edge detector was still found somewhat inferior to the one present in OpenCV library.