Canny Edge Detection

Kamal Sharma
Entry_no.:2017csb1084

Indian Institute of Technology, Ropar
Punjab, India

Abstract

Edges are primary image artifacts for extraction by low-level processing techniques, and the starting point for many computer vision projects.Canny is well defined and reliable method of edge detect. This report presents Canny edge detection algorithm implementation. [3].

1 Introduction

The **Canny edge detector** uses a multi-stage algorithm to detect a wide range of edges in images. It was developed by John F. Canny in 1986. The idea today is to build an algorithm that can **sketch the edges** of any object present on a picture, using the Canny edge detection algorithm. Canny edge detection is useful for extracting structural information from objects and reduce the amount of processing data for computer vision applications.

2 Methodology and Algorithms

Canny edge detection algorithm comprises of 5 steps: [1]
Reduce noise
First I converted image to to gray-scale image and then to get rid of noise I applied Gaussian blur on image to smooth it. Sigma is taken as 1 for the gauss filter of scipy.ndimage.
Gradient calculation
In this step, I calculated gradient magnitude and gradient direction. Edges is supposed to be detected pixels´ intensity change. To detect edge, I applied sobel filter of 5*5 size that highlight change in intensity in both directions (horizontal(x) and vertical(y)).
Filter used is :
for horizontal direction

$$\begin{bmatrix} -1 & -2 & 0 & +2 & +1 \\ -4 & -8 & 0 & +8 & +4 \\ -6 & -12 & 0 & +12 & +6 \\ -4 & -8 & 0 & +8 & +4 \\ -1 & -2 & 0 & +2 & +1 \end{bmatrix}$$

for vertical direction

$$\begin{bmatrix} -1 & -4 & -6 & -4 & -1 \\ -2 & -8 & -12 & -8 & -2 \\ 0 & 0 & 0 & 0 & +0 \\ +2 & +8 & +12 & +8 & +2 \\ +1 & +4 & +6 & +4 & +1 \end{bmatrix}$$

These filter are applied to find gradient in both directions (F_x, F_y)
Grad_magnitude $=\sqrt{F_x^2 + F_y^2}$
Grad_direction $=arctan(\frac{F_y}{F_x})$

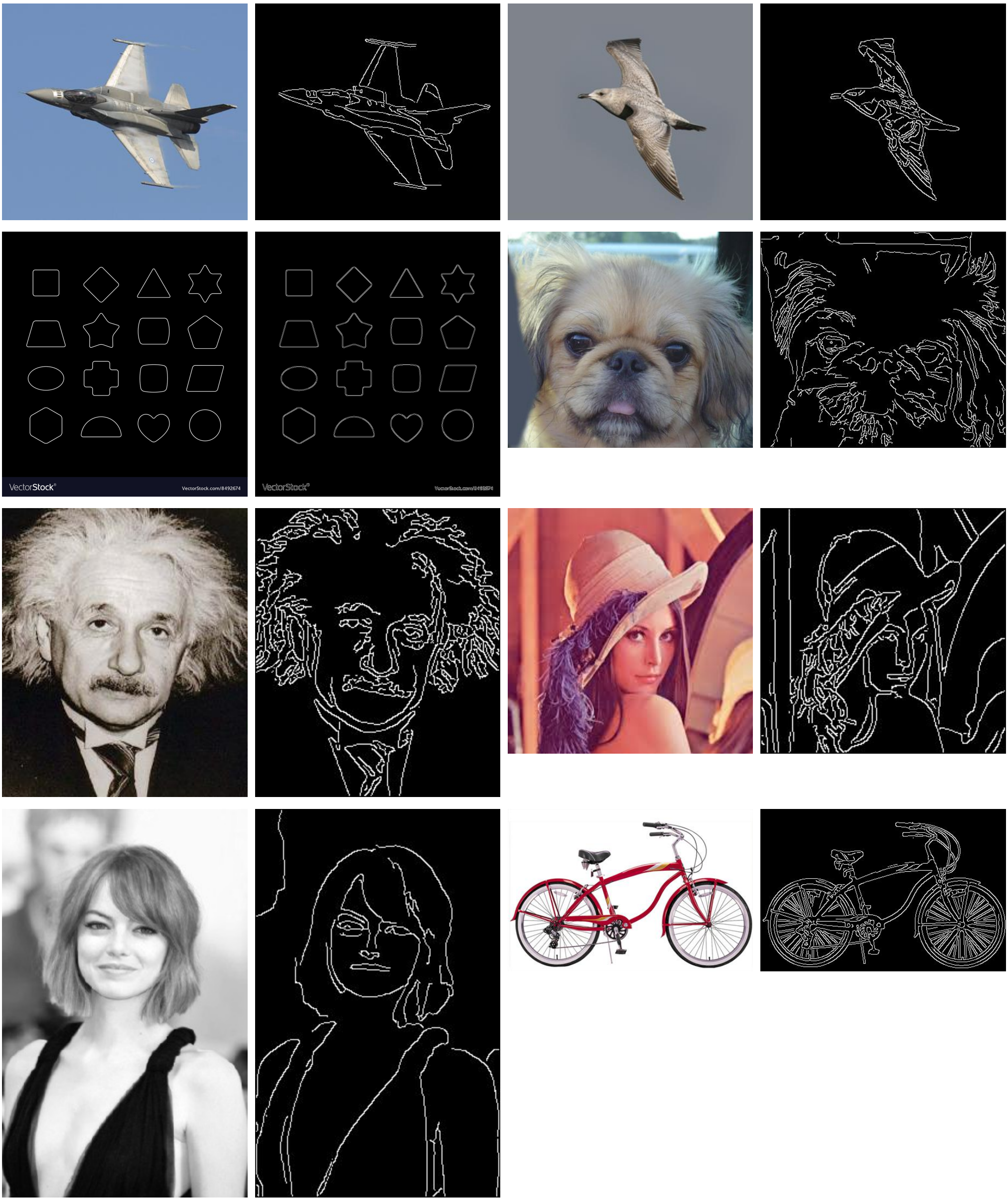


Figure 1: Canny edge detection output

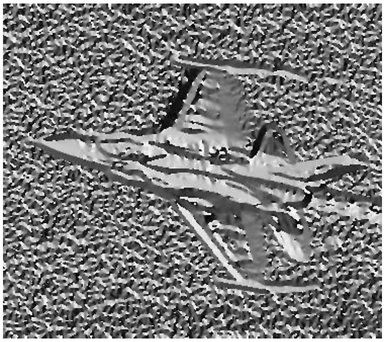
(a)original_image

(b)smoothen_image

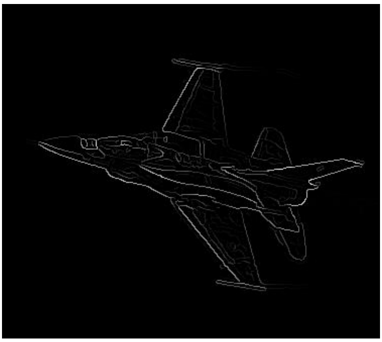
(c)F_x(sobel)

(d)F_y(sobel)

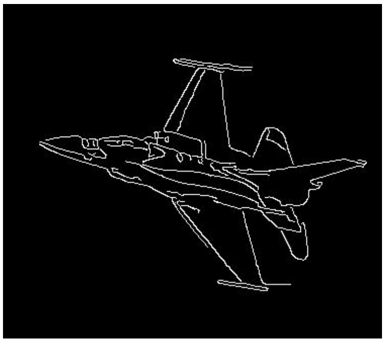
(e)Gradient_magnitude



(f)Gradient_ydirection



(g)image_after_nms



(h)final_image

Figure 2: Intermediate steps images

Non-maximum suppression

For each pixel, I found direction in (0, $\pi/4$, $\pi/2$, $3\pi/4$) that is closest to the orientation at that pixel. If the edge strength Grad_Mag[y, x] is smaller than at least one of its neighbors along the direction, set I[y, x] = zero, otherwise, set I[y, x] = Grad_mag[y, x].After thinning, image should not have thick edges.

See Algorithm 1 NMS

Double threshold

High threshold is set to find the strong pixels (form edge) and low threshold is set to find non-relevant pixels (donot form edge). Weak pixels are those having value between low and high threshold.

Th = np.max(cpy) * Th_ratio and Tl = Th * Tl_ratio

See Algorithm 2 Hysteresis_Thresholding.

Edge Tracking by Hysteresis [2]

Weak pixels that are connected in a chain to a strong pixel are need to be considered only.If any of the 8 neighboring pixels have greater magnitude than high threshold value than that pixel is considered into edge.

Algorithm 1 NMS

Input: Mag, Dir ▷ Grad_magnitude and Grad_direction
Output: img_after_nms

initialise img_after_nms with same shape as of Mag
M, N *are dimensions of Mag.shape*
for $i \leftarrow 1$ to M-1 **do**
  for $j \leftarrow 1$ to N-1 **do**
    $D \leftarrow Dir[i, j]$
    if D < 0 **then**
      $D \leftarrow D + \pi$ ▷ π = 180
    end if
    $adj1,\ adj2 \leftarrow 1,\ 1$ ▷ adj1, adj2 are pixels in same direction
    if $(D >= 0 \ and\ D <= \pi/8)\ or\ (D >= 7\pi/8\ and\ D <= \pi)$ **then**
      $adj1,\ adg2 \leftarrow Mag[i, j+1],\ Mag[i, j-1]$
    else if $(\ D >= \pi/8\ and\ D <= 3\pi/8\ )$ **then**
      $adj1,\ adg2 \leftarrow Mag[i+1, j+1],\ Mag[i-1, j-1]$
    else if $(D >= 3\pi/8\ and\ D <= 5\pi/8\ )$ **then**
      $adj1,\ adg2 \leftarrow Mag[i+1, j],\ Mag[i-1, j]$
    else if $(\ D >= 5\pi/8\ and\ D <= 7\pi/8\ )$ **then**
      $adj1,\ adg2 \leftarrow Mag[i+1, j-1],\ Mag[i-1, j+1]$
    end if
    if $(\ Mag[i, j] > adj1\ )\ and\ (\ Mag[i, j] > adj2\ )$ **then**
      $img\_after\_nms[i, j] \leftarrow Mag[i, j]$
    end if
  end for
end for

Algorithm 2 Hysteresis_Thresholding

Input: img, Th_ratio, Tl_ratio
Output: cpy ▷ final image

```
copy img in cpy
Th ← max(cpy) * Th_ratio
Tl ← Th * Tl_ratio
total, prev_total ← 1, 0
M, N are dimensions of cpy.shape
while total != prev_total do
    prev_total ← total
    total ← 0
    for i ← 1 to M-1 do
        for j ← 1 to N-1 do
            if (cpy[i, j] < Tl) then
                cpy[i, j] ← 0
            else if (isPossible(cpy, i, j, Th)) then
             ▷ isPossible returns true if magnitude of cpy[i,j] or any of 8 neighbour > Th
                cpy[i, j] ← 0
                total ← total + 1
            end if
        end for
    end for
end while
if any of the cpy[i, j] != 1 then assign cpy[i, j] = 0
```

3 Results and Observations

After smoothing the noise is reduced. Gradients in both vertical direction and horizontal directions are found by sobel filter as it is prone to sudden change in pixel intensity.
Gradient magnitude image has edges which may be thicker. Hence non maximal suppressing is done to thin the edges. Threshold(high) is set to find which edges are definitely in image and Threshold(low) is set to find weak edges.
Weak edges are taken in consideration if then connects the strong edges. This is done by tracking the strong edges.

4 Conclusion and Takeaway

- I have used 5*5 sobel filter instead of 3*3 for improving the accuracy of edge detection. Filter description is written in methodology.

- Learnt that how to deal with pixels having exactly opposite gradient directions. (I have done this by changing negative grad_direction to grad_direction + pi(180).

- Increasing the higher threshold ratio reduces the strong edges and increasing the lower threshold ratio leads to suppression of number of weak edges.

- Balancing the weak and strong edges can be done by tuning threshold ratios. Say if Thresh_low_ratio is increased less pixels qualify for the weak edges.

References

[1] Written by: Sofiane Sahir at Toward data science. Canny edge detection, 2006. https://towardsdatascience.com/canny-edge-detection-step-by-step-in-python-computer-vision-b49c3a2d8123.

[2] MadhavEsDios. Canny. http://justin-liang.com/tutorials/canny/.

[3] Wikipedia. Canny_edge_detector. https://en.wikipedia.org/wiki/Canny_edge_detector.