Canny Edge Detector From First Principles

Aman Bilaiya
2018csb1069

Indian Institute Of Technology Ropar
Rupnagar, Punjab

Abstract

Feature Extraction is an important tool to draw meaningful insights from the image and perform analysis. Edges and Corners play an important role in various Image Feature Extraction Algorithms and Computer Vision Applications. Hence it becomes essential to detect corners and edges efficiently in an image. This report presents the implementation, observations and results of extracting edge and corners features from images using first principles, using python. This work consists of - Implementing a **Canny edge detector.**

1 Introduction

Understanding and developing any vision related model requires feature extraction from the image for analysis. Edges and Corners are the features that are trivially identified by our human eye. Thus extracting edges and images is of prime importance in computer vision.

The Canny edge detector is an edge detection technique that uses a multi-stage algorithm flow. It was developed by John F. Canny in 1986. It extracts useful structural information from different vision objects and dramatically reduce the amount of data to be processed. It has been widely applied in various computer vision and imaging applications. Canny Edge detector works on the principle that intensity changes suddenly across the edge, however remains uniform along the edge.

2 Methodology

Canny edge detection algorithm comprises of the following steps :-

[1] Preprocessing and computing smoothed gradients: First step is converting RGB image to gray scale. Edge Detection results are highly sensitive to image noise, so to get rid of these noises, input image is smoothened using Gaussian filter. Then the gradient component Ix and Iy along x and y direction respectively are calculated for every single point in the image. But, for faster computation, we can merge these two steps into one by convolving with the x and y Guassian derivatives. This is done by convolving with standard 3*3 sobel x and y filters.

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

The gradient magnitude at a point(x,y) of image determines if point possibly lies on an edge or not. A high gradient magnitude means the sudden intensity change - implying an edge. The gradient direction shows how the edge is oriented.

Algorithm 1 convolveWithGaussianDerivative(img)

Ix $\leftarrow covolve(sobel\_x, img)$
Iy $\leftarrow covolve(sobel\_y, img)$
G $\leftarrow \sqrt{Ix^2 + Iy^2}$
Theta $\leftarrow arctan(\frac{Iy}{Ix})$
return Ix, Iy, Theta

[2] Non Maximal Suppression: The gradient magnitude image(G) obtained above can have thick edges. For each pixel in the gradient magnitude image we look at the pixels in its neighbourhood along the gradient direction at that point. Gradient direction at each pixel is rounded off to four bins - 0°,45° , 90° and 135°. We retain that pixel only if it is a local maxima and suppress other pixels. This results in a thinned image with single pixel edges.

Algorithm 2 Non_maximal_Suppression(Gradient, Theta)

$angle \leftarrow deg(Theta)$
for $i \leftarrow 1 : Gradient\_height - 1$ **do**
  for $j \leftarrow 1 : Gradient\_width - 1$ **do**
    $x = y = 1.0$
    if $angle[i][j] < 0$ **then**
      $angle[i][j] \leftarrow angle[i][j] + 180$
    end if
    if $angle[i][j] \in 0°$ **then**
      $x \leftarrow Gradient[i][j+1]$
      $y \leftarrow Gradient[i][j-1]$
    else if $angle[i][j] \in 45°$ **then**
      $x \leftarrow Gradient[i+1][j-1]$
      $y \leftarrow Gradient[i-1][j+1]$
    else if $angle[i][j] \in 90°$ **then**
      $x \leftarrow Gradient[i+1][j]$
      $y \leftarrow Gradient[i-1][j]$
    else if $angle[i][j] \in 135°$ **then**
      $x \leftarrow Gradient[i-1][j-1]$
      $y \leftarrow Gradient[i+1][j+1]$
    end if
    if $Gradient[i][j] \geq x \; and \; Gradient[i][j] \geq y$ **then**
      $thinned\_edge\_image[i][j] \leftarrow Gradient[i][j]$
    else
      $thinned\_edge\_image[i][j] \leftarrow 0$
    end if
  end for
end for
return thinned_edge_image

[3] Hysteresis Thresholding: Now, we need to mark pixels as edges based on some threshold on gradient magnitude. Using double thresholding, the edges in the thinned image are marked as weak or strong. Those above the high threshold are marked as strong and those between the high and low threshold are marked as weak while those below low threshold are marked as not edges.

Algorithm 3 Hysteresis_Thresholding(Gradient, Th_low, Th_high)

Double Thresholding :
for $i \leftarrow 1 : Gradient\_height - 1$ **do**
  for $j \leftarrow 1 : Gradient\_width - 1$ **do**
    if $Gradient[i][j] \geq Th\_high$ **then**
      $thresholded\_img[i][j] \leftarrow 1.0$
    else if $Th\_low \leq Gradient[i][j] < Th\_high$ **then**
      $thresholded\_img[i][j] \leftarrow 0.5$
    else
      $thresholded\_img[i][j] \leftarrow 0.0$
    end if
  end for
end for

Linking weak & strong edges :
$result \leftarrow thresholded\_img.copy()$
for $i \leftarrow 1 : thresholded\_img - 1$ **do**
  for $j \leftarrow 1 : thresholded\_img - 1$ **do**
    if $thresholded\_img[i][j] == 0.5$ **then**
      if $8\_neighbours(thresholded\_img[i][j]) == 1.0$ **then**
        $result[i][j] \leftarrow 1.0$
      else
        $result[i][j] \leftarrow 0.0$
      end if
    end if
  end for
end for
return result

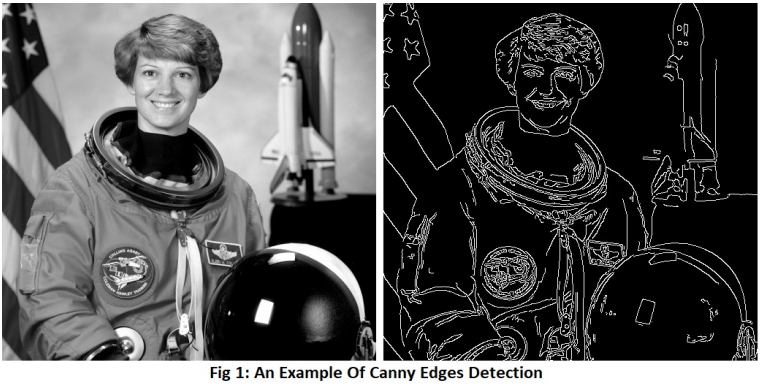


Fig 1: An Example Of Canny Edges Detection

3 Results & Observations

On performing NMS following observations were made:
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.

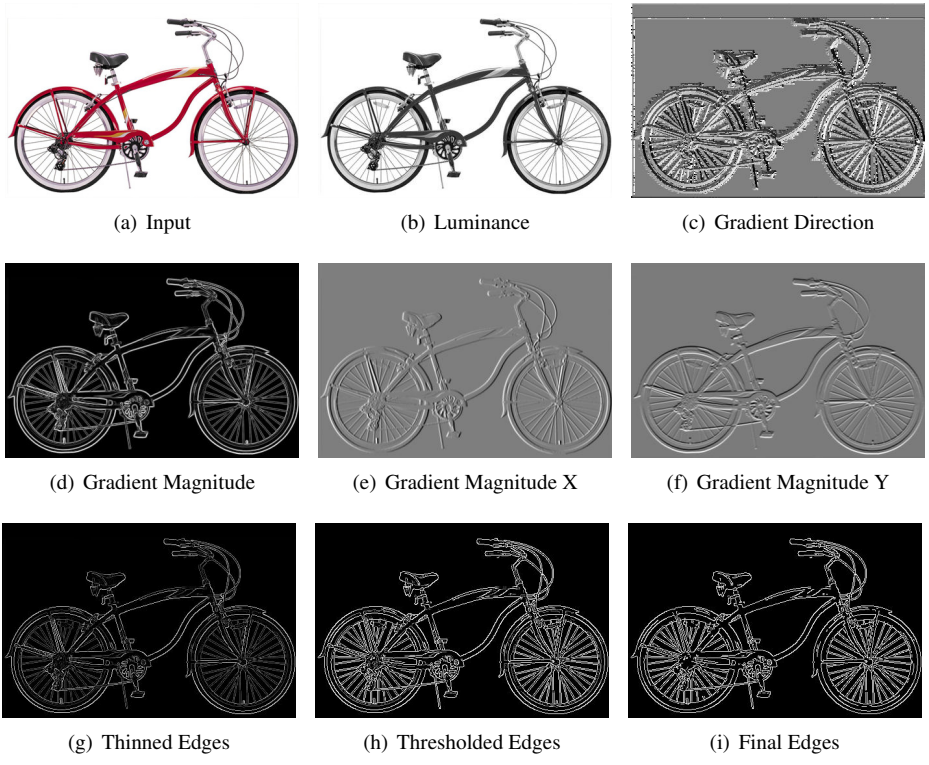
On performing Hysteresis Thresholding following observations were made:
1. On increasing the upper threshold value, some potential strong edges can be missed upon.
2. On increasing the lower threshold value, leads to suppression of number of weak edges.
3. Balancing the weak and strong edges can be done by tuning threshold ratios. The threshold values for images differ from image to image. With proper thresholds chosen, edges are detected quite accurately.
4. If you select large lower threshold value, many potential edges will be left out and if you select small upper threshold, many weak edges will be detected due to noise in the image. It is recommended to select small lower threshold and large upper threshold. Difference between high_Th and low_Th ensures only connected edge detection.

4 Insights and Take Aways

1. Canny edge detector is invariant to rotation because gradient magnitude remains same, only the gradient direction changes.
2. Canny edge detector is invariant to linear translation because gradient magnitude and direction, both remain same.
3. Canny edge detector is invariant to intensity shift because the first derivative involved remains unchanged on linear shift.
4. Canny edge detector is not invariant to scaling. Relative neighbourhood pixels intensity shifts change on scaling and thus the gradient magnitude and direction.
5. Canny edge detector is sensitive to noise in the image. So, Gaussian smoothing is used to remove noise effects on edge detection.

5 References

1) IITM Summer School
2) Wikipedia
3) Step by step algorithm
3) Lectures slides and Online resources



(a) Input (b) Luminance (c) Gradient Direction

(d) Gradient Magnitude (e) Gradient Magnitude X (f) Gradient Magnitude Y

(g) Thinned Edges (h) Thresholded Edges (i) Final Edges

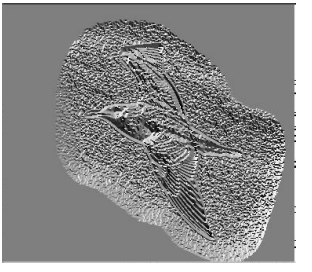
Figure 1: Intermediate and Final images of Bicycle with low_Th = 0.1, high_Th = 0.25

Figures[1-7] are the final and intermediate output images obtained from our Canny Edge detector algorithm :-

With proper thresholds chosen (low_Th = 0.1, high_Th = 0.25), edges are detected close to accurate.

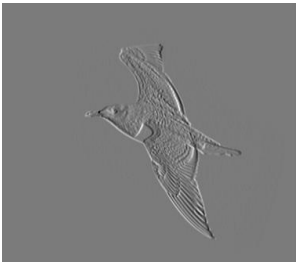
(a) Input

(b) Luminance



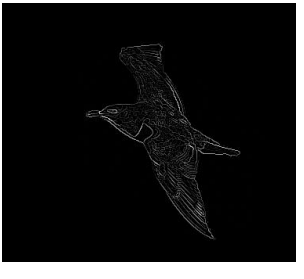
(c) Gradient Direction

(d) Gradient Magnitude

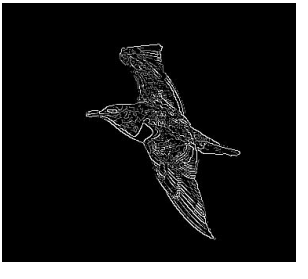


(e) Gradient Magnitude X

(f) Gradient Magnitude Y



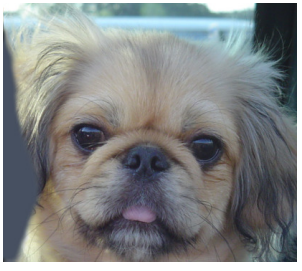
(g) Thinned Edges



(h) Thresholded Edges

(i) Final Edges

Figure 2: Intermediate and Final images of bird with low_Th = 0.1, high_Th = 0.25

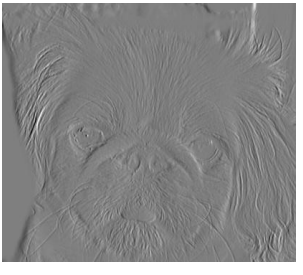


(a) Input

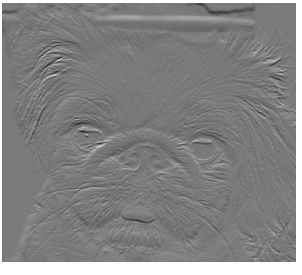
(b) Luminance

(c) Gradient Direction

(d) Gradient Magnitude



(e) Gradient Magnitude X



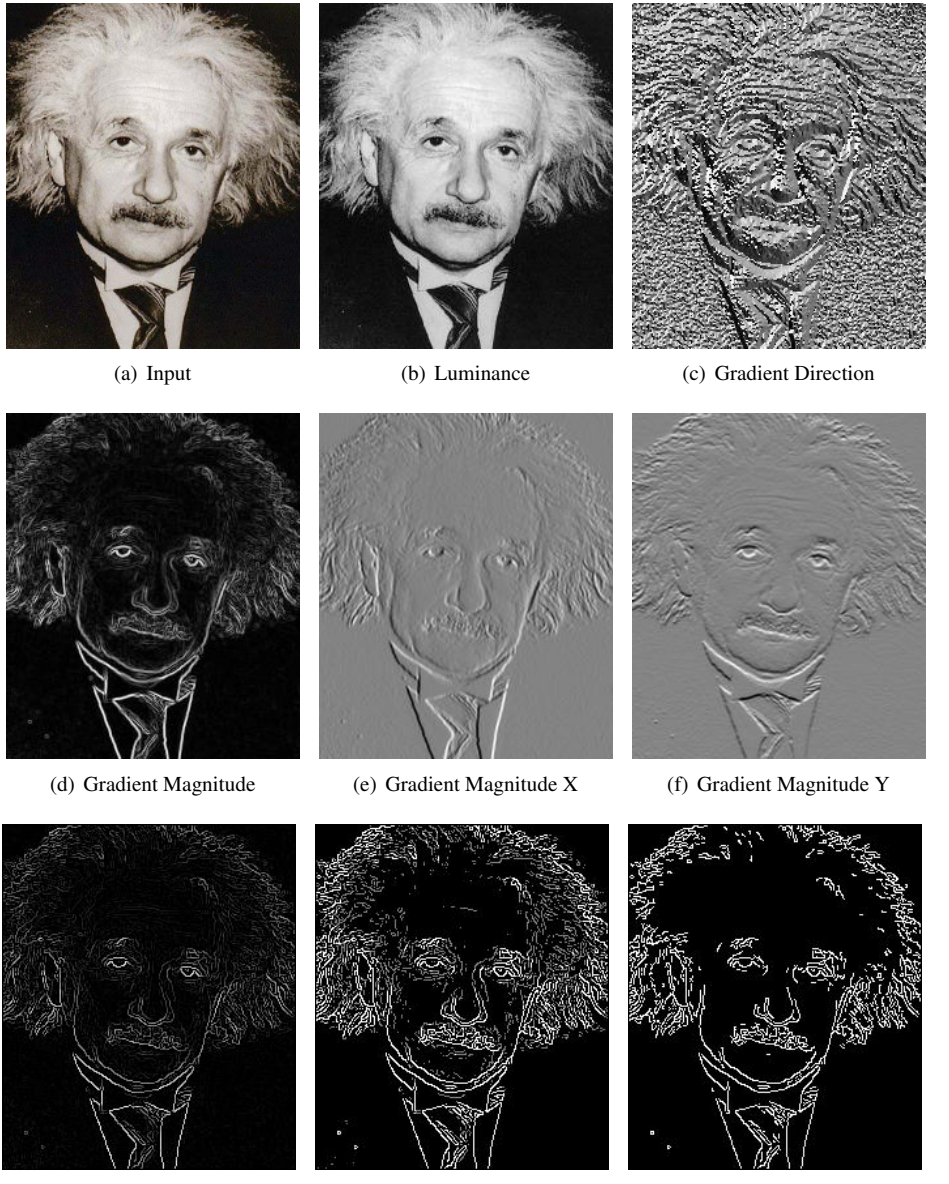
(f) Gradient Magnitude Y

(g) Thinned Edges

(h) Thresholded Edges

(i) Final Edges

Figure 3: Intermediate and Final images of dog with low_Th = 0.1, high_Th = 0.25

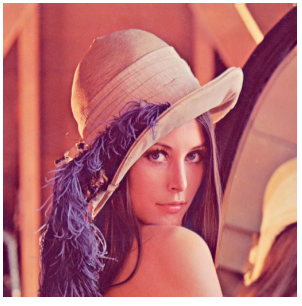


(a) Input (b) Luminance (c) Gradient Direction

(d) Gradient Magnitude (e) Gradient Magnitude X (f) Gradient Magnitude Y

(g) Thinned Edges (h) Thresholded Edges (i) Final Edges

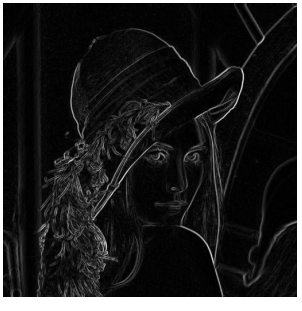
Figure 4: Intermediate and Final images of einstein with low_Th = 0.1, high_Th = 0.25



(a) Input

(b) Luminance

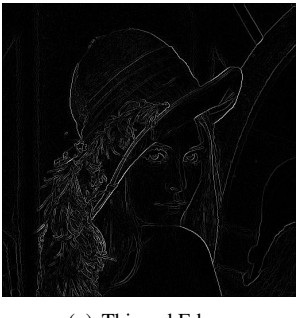
(c) Gradient Direction



(d) Gradient Magnitude

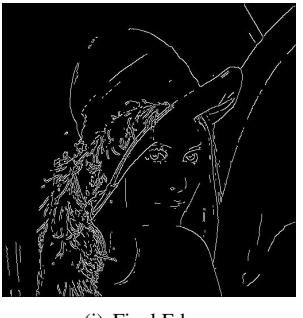
(e) Gradient Magnitude X

(f) Gradient Magnitude Y



(g) Thinned Edges

(h) Thresholded Edges

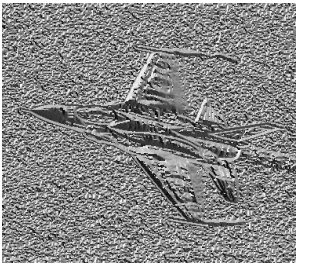


(i) Final Edges

Figure 5: Intermediate and Final images of leena with low_Th = 0.1, high_Th = 0.25

(a) Input

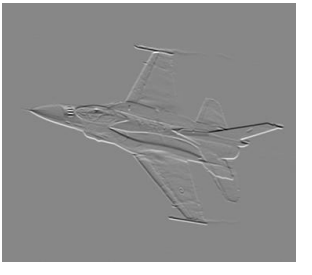
(b) Luminance



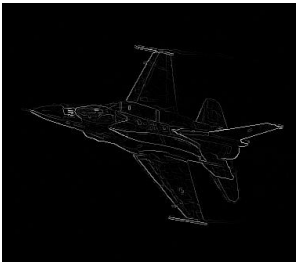
(c) Gradient Direction

(d) Gradient Magnitude

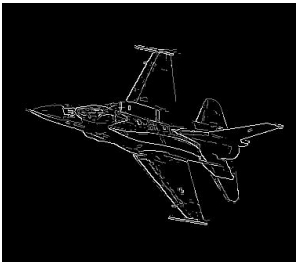
(e) Gradient Magnitude X



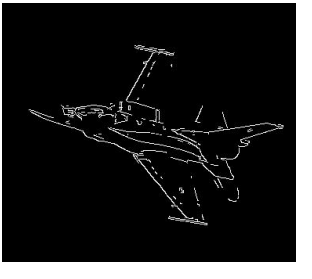
(f) Gradient Magnitude Y



(g) Thinned Edges

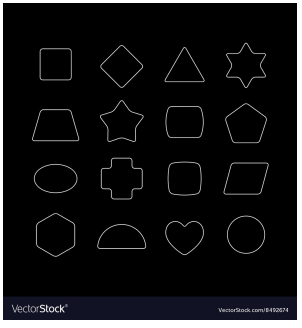


(h) Thresholded Edges

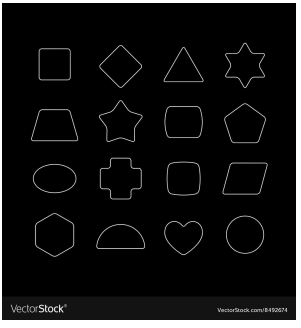


(i) Final Edges

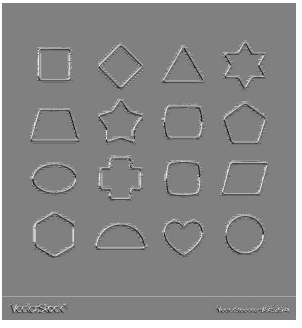
Figure 6: Intermediate and Final images of plane with low_Th = 0.1, high_Th = 0.25



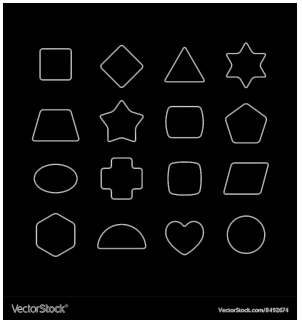
(a) Input



(b) Luminance

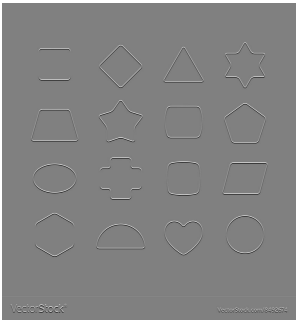


(c) Gradient Direction

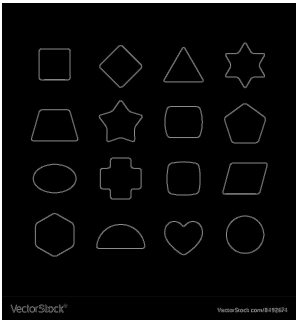


(d) Gradient Magnitude

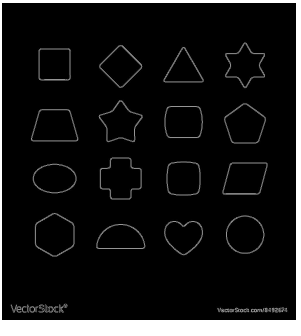
(e) Gradient Magnitude X



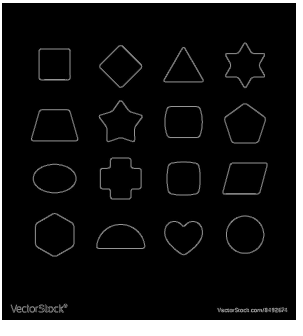
(f) Gradient Magnitude Y



(g) Thinned Edges



(h) Thresholded Edges



(i) Final Edges

Figure 7: Intermediate and Final images of toy¡$magewithlow\_Th = 0.1, high\_Th = 0.25$