Feature Extraction : Edges and Corners

Sahil Sidheekh
2017CSB1104
Indian Institute Of Technology, Ropar

Abstract

This document demonstrates the observations and results of extracting features - Edges and corners from images using **first principles**, in **python** .This work consists of two sub parts - Implementing a **Canny edge detector** and a **Harris corner detector**.

1 Introduction

Understanding and developing any vision related model requires extraction of features from the image for analysis. Edges and Corners are the features that are trivially identified by our human vision. Thus extracting edges and images is of prime importance in computer vision. We thus discuss two important Algorithms for detecting edges and corners in an image, as described in the section below.

2 Theory and Intuition

2.1 Canny Edge Detection

Canny edge detection is a technique to extract useful structural information from different vision objects. The canny edge detection Algorithm consists of the following sub-processes

2.1.1 Computing smoothed gradients

A Gaussian filter is applied to smooth the image in order to remove the noise and then the gradients of the image are computed along the x and y directions. The gradient magnitudes and directions are thus obtained.

2.1.2 Non-maximal Suppression

The gradient magnitude image 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 and retain that pixel only if it is a local maxima. This results in a thinned image with single pixel edges.

2.1.3 Hysteresis Thresholding

Using two thresholds, the edges in the thinned image are marked as weak or strong - those above the high threshold are marked strong and those between the high and low threshold are marked weak while those below low threshold are discarded.

© 2012. The copyright of this document resides with its authors.
It may be distributed unchanged freely in print or electronic forms.

2.1.4 Linking

The hysterisis thresholded image is analyzed and all weak pixels connected to a strong pixel are included in the final output and others are discarded.

2.2 Harris Corner Detection

A corner is a point of intersection of edges. Thus, while an edge has a change in gradient in any one direction, a corner has a change of gradients along multiple directions. This is the principle used in harris corner detector. The processes involved are :

2.2.1 Compute Image Gradients

The image gradients along x and y directions are computed by convolving with appropriate filters.

2.2.2 Harris Corner Score Calculation and Thresholding

Using a window of size (2m+1)X(2m+1) the gradient image is scanned and the covariance matrix is computed. Using taylor series expansion, this computation is equivalent to :

$$C[x,y] = \frac{1}{(2m+1)^2} \sum_{u=-m}^{m} \sum_{v=-m}^{m} \begin{bmatrix} I_x^2[x+u,y+v] & I_xI_y[x+u,y+v] \\ I_xI_y[x+u,y+v] & I_y^2[x+u,y+v] \end{bmatrix}$$

The harris corner score is calculated as half the harmonic mean of the eigen values of the matrix C computed above, for each pixel. The Harris corner detection paper uses an approximation of this score as equivalent to as :

$$det(C) - k(Trace(C)^2)$$

using k = 0.04.
We use the same formula to calculate the corner scores and using an appropriate threshold, find possible corner points.

2.2.3 Non maximum suppression

Within the eight-connected neighbourhood of each pixel, the local maxima which satisfies the threshold is set as a corner point and rest are discarded.

3 Implementation

The implementation is in python and consists of the following 2 major modules :

canny.py - This module contains the class Canny which performs the above mentioned steps for calculating the edges from an input grayscale image. The outputs are stored to the folder : results/canny

harris.py - This module contains the class Harris which performs the above mentioned steps for calculating the corners from an input grayscale image. The outputs are stored to the folder : results/harris

Algorithm 1: Canny Edge Detection(img)

Result: Edge Image
$I_x = \frac{d}{dx}(img)$
$I_y = \frac{d}{dy}(img)$
result = img
$\delta I_{mag} = \sqrt{I_x^2 + I_y^2}$
$\delta I_{dir} = \arctan(\frac{I_y}{I_x})$
for $i \leftarrow 0$ **to** *img.height* **do**
  for $j \leftarrow 0$ **to** *img.width* **do**
    if $\delta I_{mag}(i,j)$ *is local maxima along* $\delta I_{dir}(i,j)$ **then**
      $result(i,j) = \delta I_{mag}(i,j)$
    else
      $result(i,j) = 0$
    end
  end
end
for $i \leftarrow 0$ **to** *img.height* **do**
  for $j \leftarrow 0$ **to** *img.width* **do**
    if $result(i,j) >= \tau_{high}$ **then**
      $result(i,j) = E_{strong}$
    else
      if $result(i,j) < \tau_{low}$ **then**
        $result(i,j) = 0$
      else
        $result(i,j) = E_{weak}$
      end
    end
  end
end
for $i \leftarrow 0$ **to** *img.height* **do**
  for $j \leftarrow 0$ **to** *img.width* **do**
    if $result(i,j) = E_{weak}$ *and* (i,j) *connects* E_{strong} **then**
      $result(i,j) = 1$
    else
      $result(i,j) = 0$
    end
  end
end

Algorithm 2: Harris Corner(img)

Result: Corner Points

$I_x = \frac{d}{dx}(img)$

$I_y = \frac{d}{dy}(img)$

$$C[x,y] = \frac{1}{(2m+1)^2} \sum_{u=-m}^{m} \sum_{v=-m}^{m} \begin{bmatrix} I_x^2[x+u,y+v] & I_xI_y[x+u,y+v] \\ I_xI_y[x+u,y+v] & I_y^2[x+u,y+v] \end{bmatrix}$$

for $i \leftarrow 0$ **to** *img.height* **do**
  for $j \leftarrow 0$ **to** *img.width* **do**
    score(i,j) = $det(C[i,j])$ - $0.04 * Trace(C[i,j])^2$
    if $score(i,j) > \tau_{harris}$ *and* (i,j) *is localmaximum* **then**
      $result(i,j) = 1$
    else
      $result(i,j) = 0$
    end
  end
end

4 Results and Observations

4.1 Canny Edge Detector

The gradient magnitudes and directions for the input images where calculated using the derivative of a gaussian kernel of size (5 X 5) and sigma = 1.4. The gradient directions where quantized to the angles : 0°, 45°, 90°, 135° and 180°. The gradient magnitude images upon performing non maximal suppression using the quantized gradient directions where found to have thinner edges of unit pixel width and it was observed that this resulted in the edges becoming **discontinuos**, possibly due to **quantization** of gradient directions. Hysterisis thresholding was performed on the image and after a lot of experimentation the thresholds were set as :

$$T_{high} = 1.33 * (I_{mean}) \tag{1}$$

$$T_{low} = 0.5 * (T_{high}) \tag{2}$$

i.e within a tolerance of 25% of the mean intensity of the thinned image. The weak pixels which were attached to some strong pixels were found to be marked as a part of the output image after linking. It was observed that while canny edge detector gives reasonably good information about edges, the relevance of the edges finally outputted were subject to the threshold values which were partially manually set. As the optimal threshold for each image was different, manual effort is still needed to get good results from the canny edge detector.

4.2 Harris Corner Detection

For the Harris corner detection, the optimal window size was found to be (9 X 9) manually, when using a gaussian with sigma = 2 as the window function. The corner response was once again found to be largely dependent on the threshold kept for the harris corner score calculated using the formula given above. As decomposing the covarience matric to find the eigen value and then calculating the corner score was computationally expensive, the approximate method of using the parametrized difference of determinant and square of trace of the covariance matrix gave reasonable output,using k=0.04, with lesser computation. After experimentation, the threshold for the corner score was set as :

$$T_{harris} = 0.1 * (E_{max}) \tag{3}$$

where E is the corner response matrix for the input image. This particular threshold ratio was found to give reasonable corner response across multiple images, thus it was set as default because having a generally well functioning threshold would be better than fine tuning for each test image for future sample input images.
For the toy image, shapes with rounded corners were **not detected** as corner points. This is an expected drawback of harris corner detector as it is **not scale invariant**.

The corner and edge responses including the intermediate images are included below for the sample test input data and were reasonably accurate and informative.
(**Please Note** that for the toy image, some shape edges in the final output for canny edge detector may be missing because this information is lost during resizing the image to fit the report, as the information is strictly single pixel wide. However these edges are actually present in the original image and can be verified by running the file test.py.)

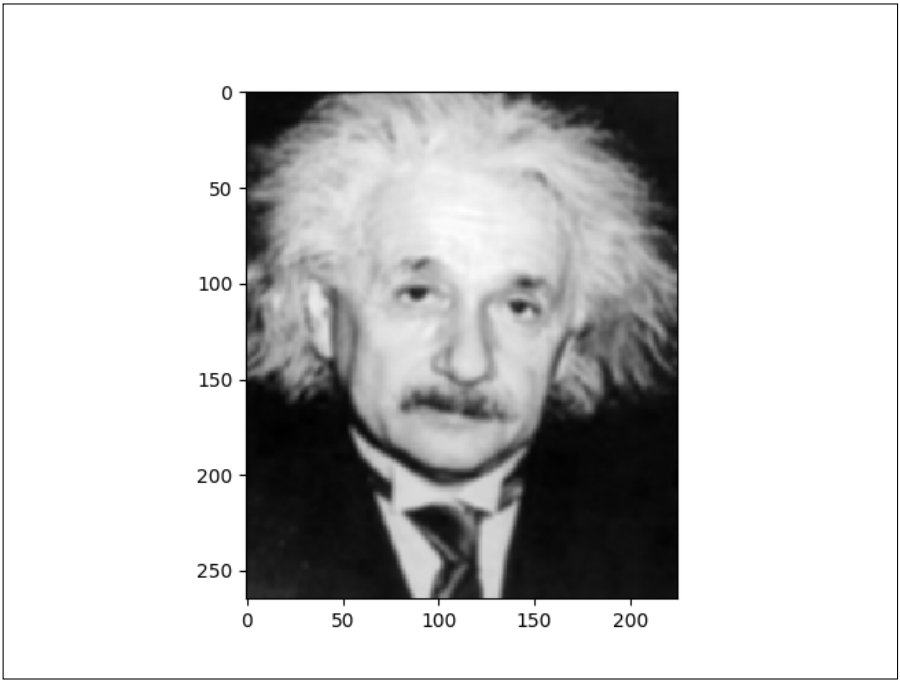

Figure 1: Einstein - Canny 1.0 : Original Image

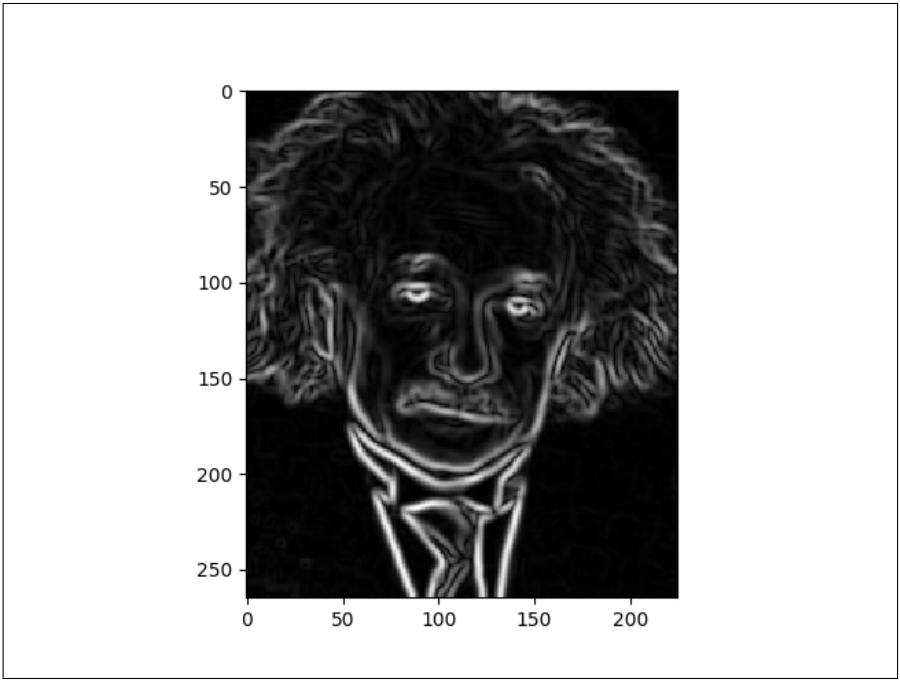

Figure 2: Einstein - Canny 1.1 : Gradient Magnitude Image

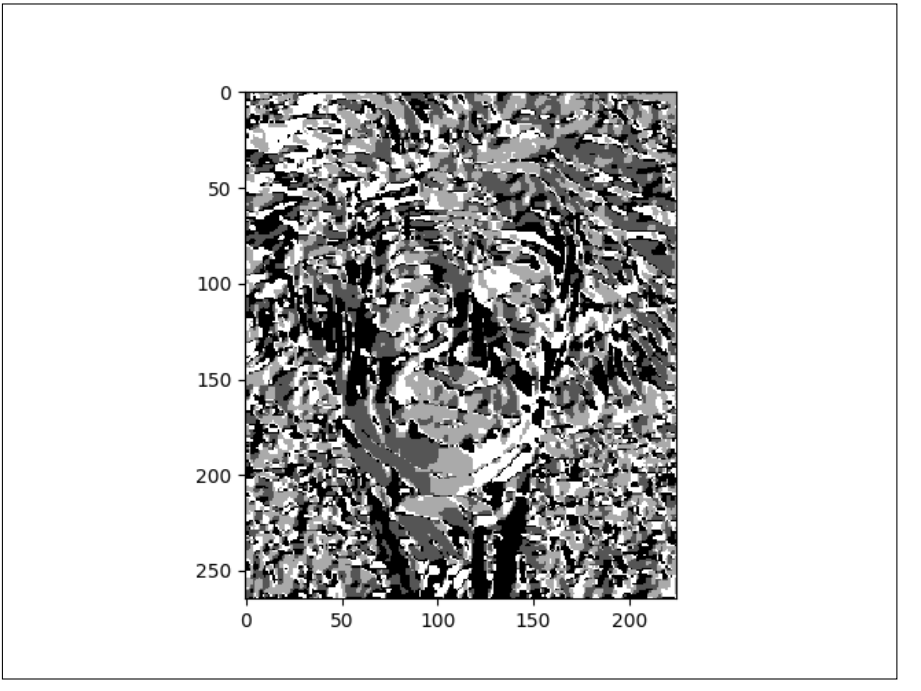

Figure 3: Einstein - Canny 1.2 : Gradient Direction Image (Quantized)

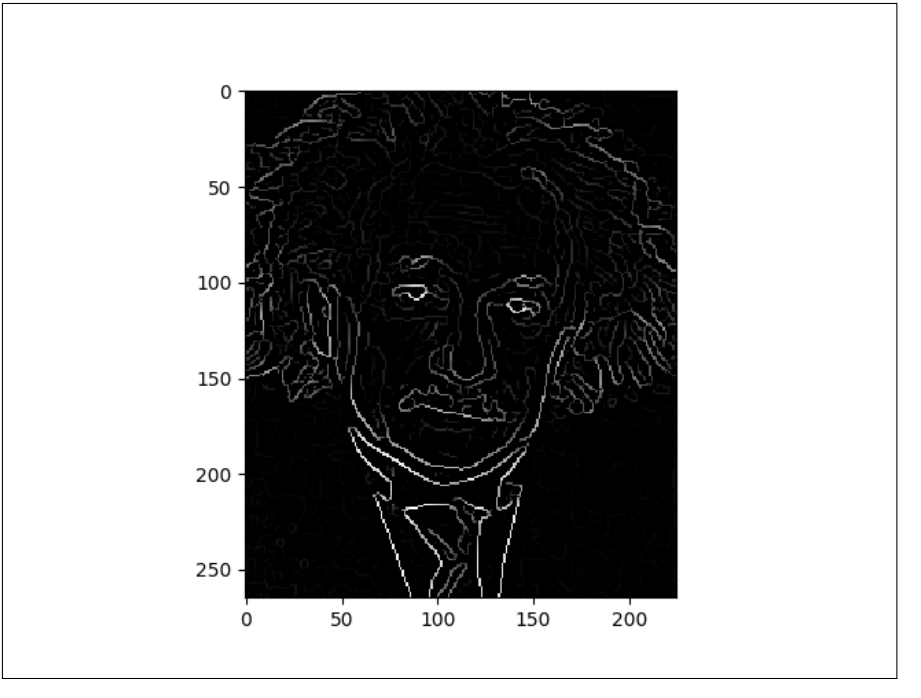

Figure 4: Einstein - Canny 1.3 : Thinned Image (Using NMS)

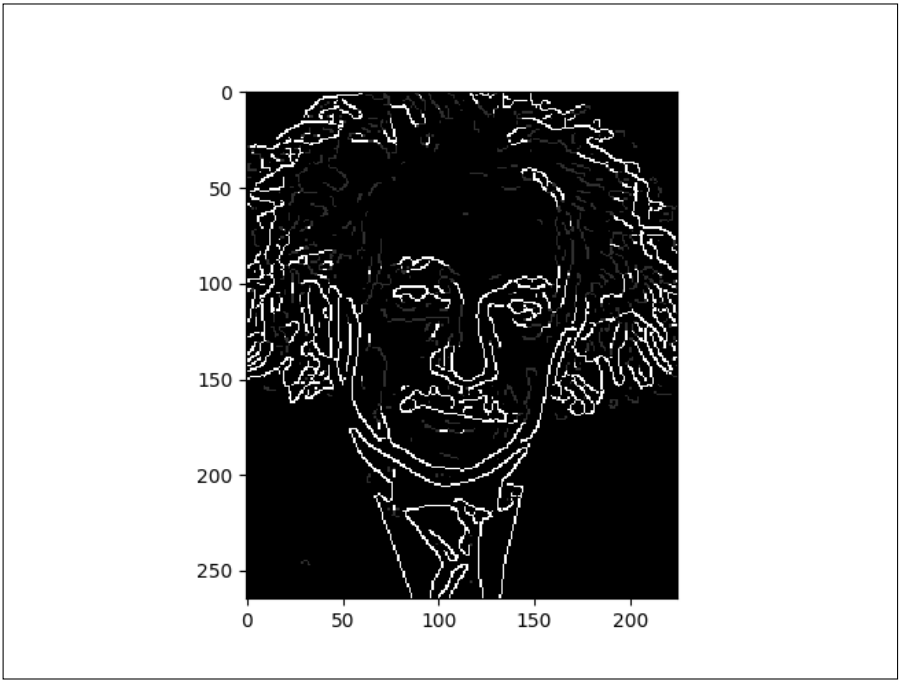

Figure 5: Einstein - Canny 1.4 : Thresholded Image

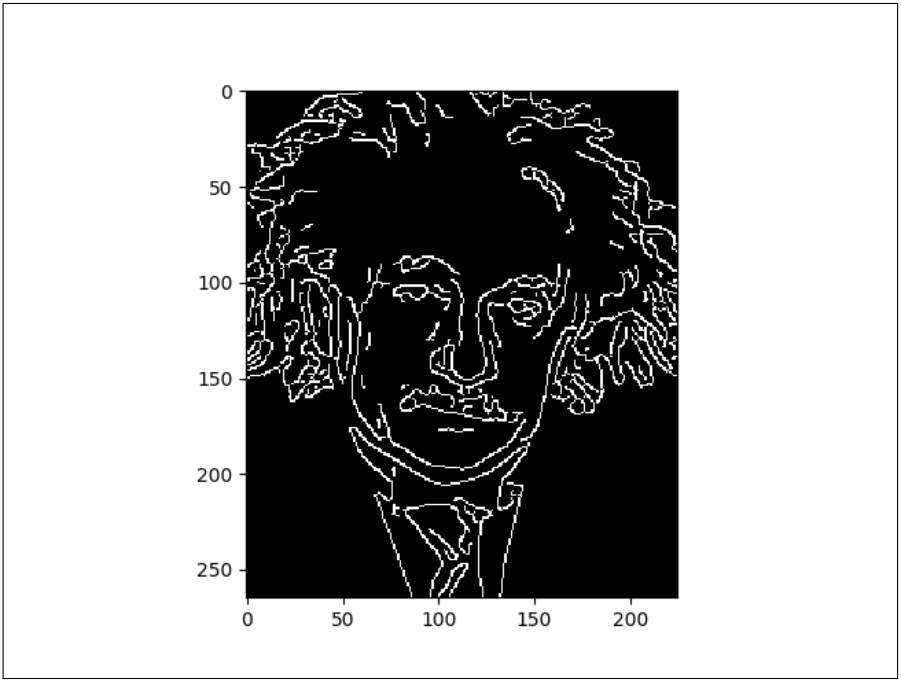

Figure 6: Einstein - Canny 1.5 : Final Image after Linking

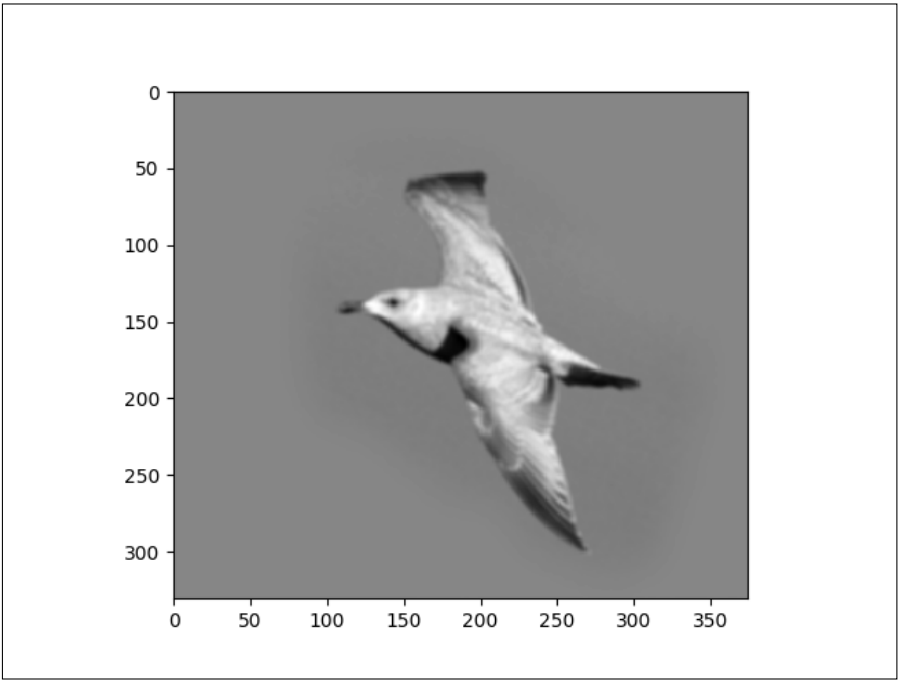

Figure 7: Bird - Canny 2.0 : Original Image

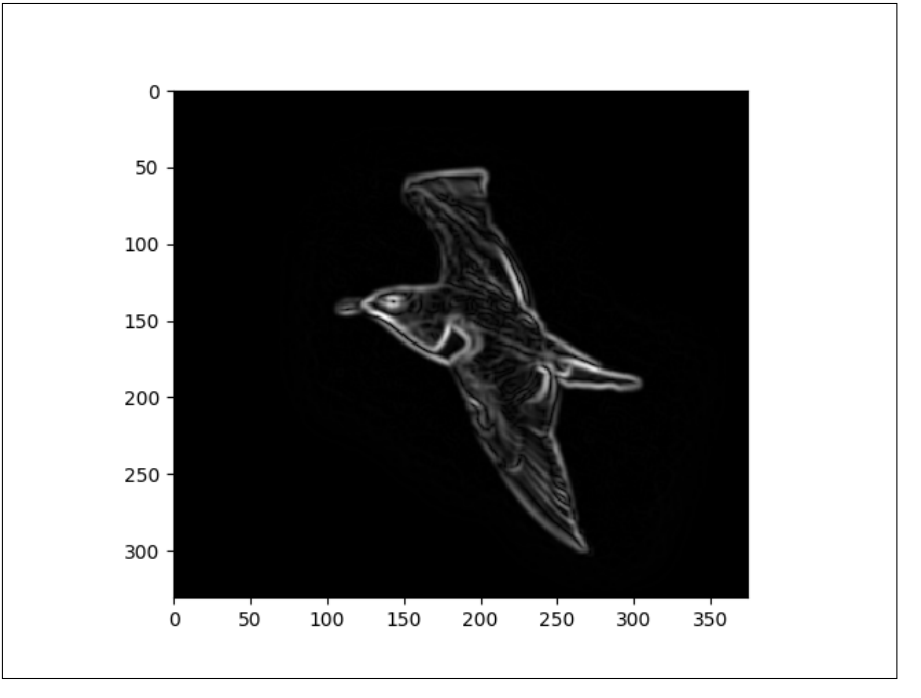

Figure 8: Bird - Canny 2.1 : Gradient Magnitude Image

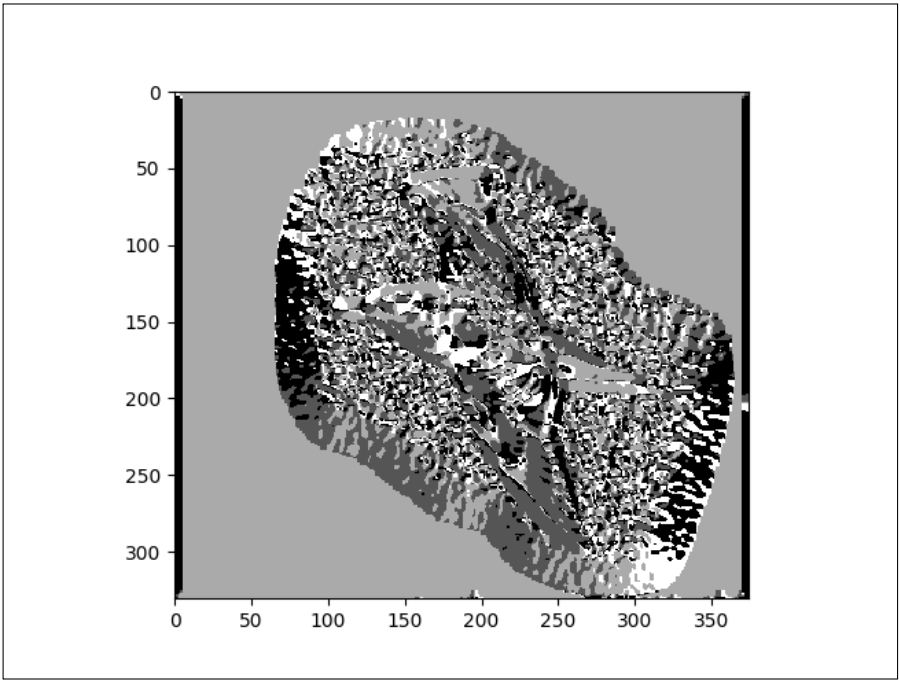

Figure 9: Bird - Canny 2.2 : Gradient Direction Image (Quantized)

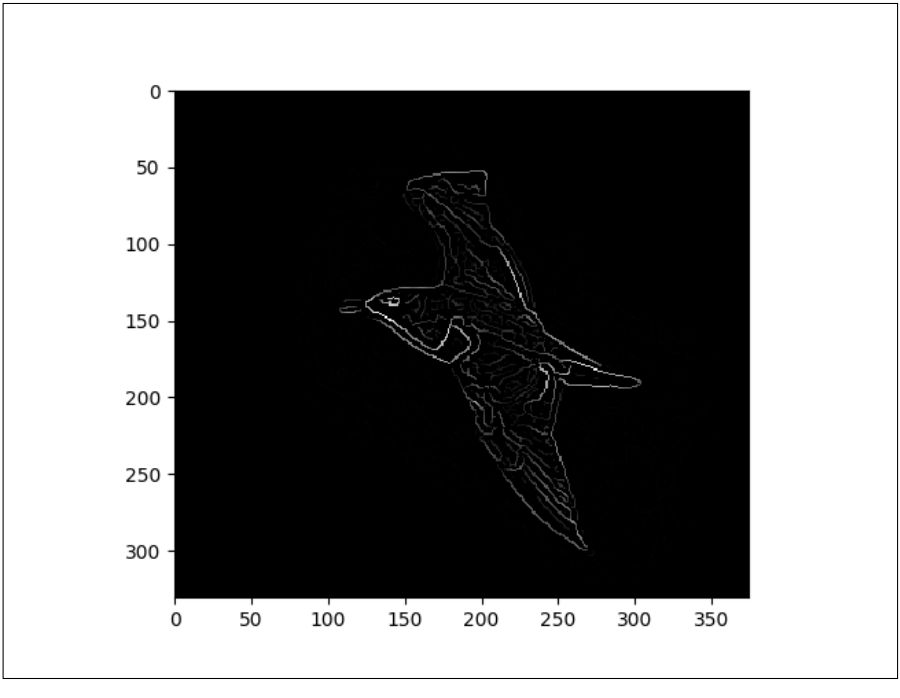

Figure 10: Bird - Canny 2.3 : Thinned Image (Using NMS)

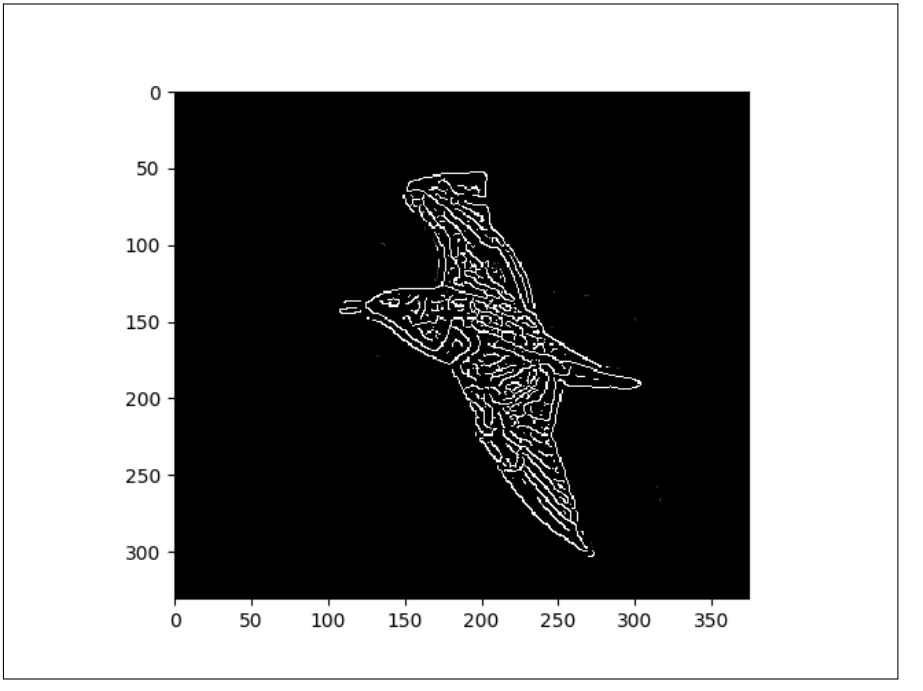

Figure 11: Bird - Canny 2.4 : Thresholded Image

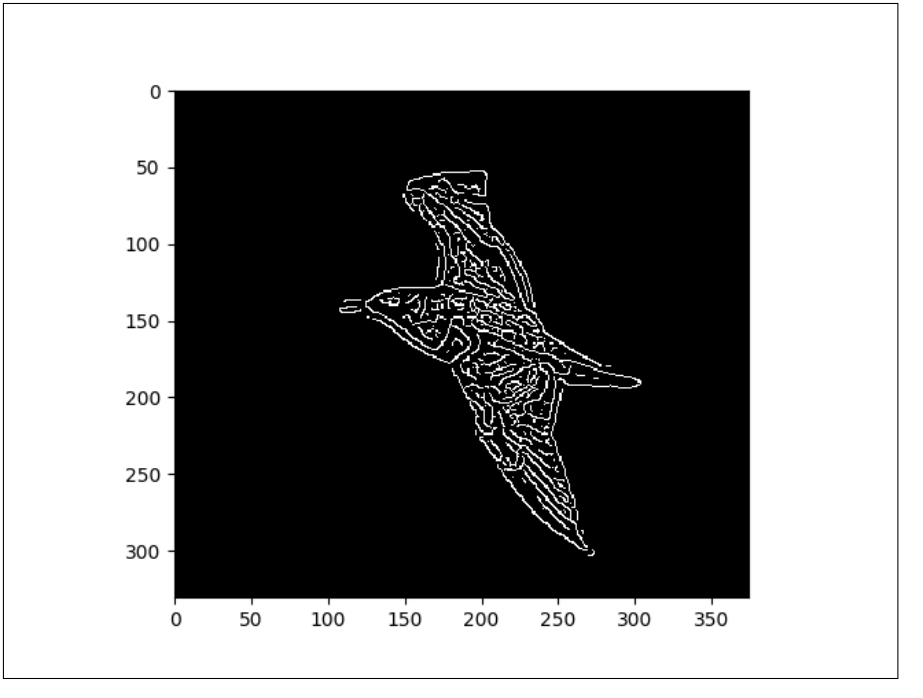

Figure 12: Bird - Canny 2.5 : Final Image after Linking

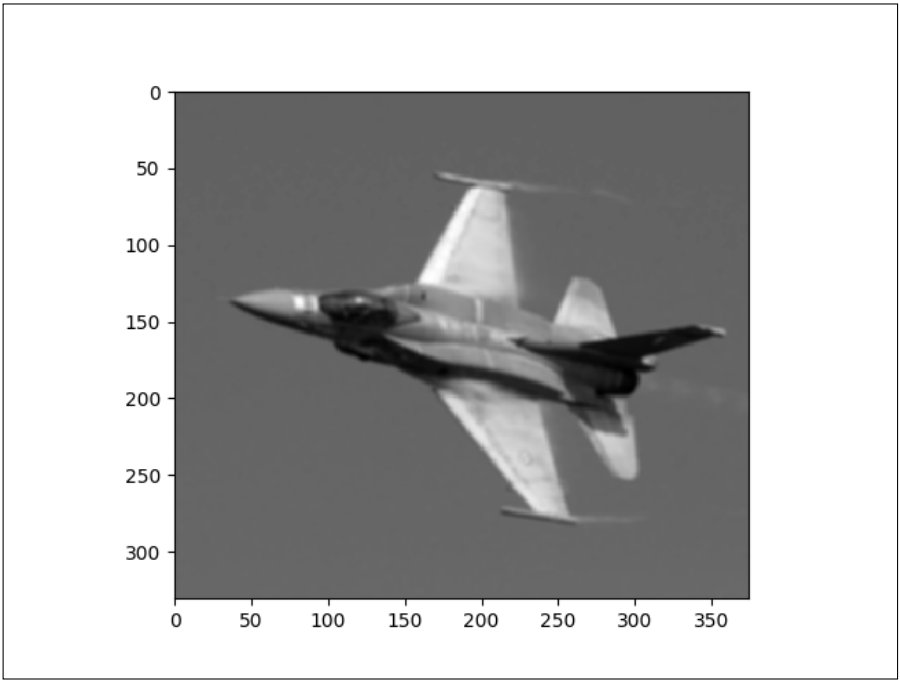

Figure 13: Plane - Canny 3.0 : Original Image

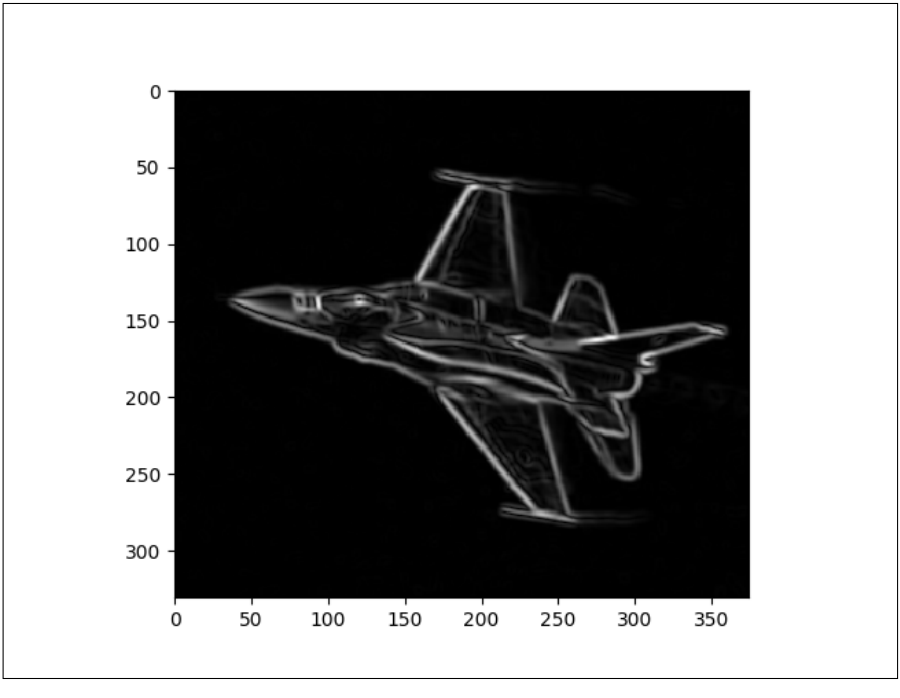

Figure 14: Plane - Canny 3.1 : Gradient Magnitude Image

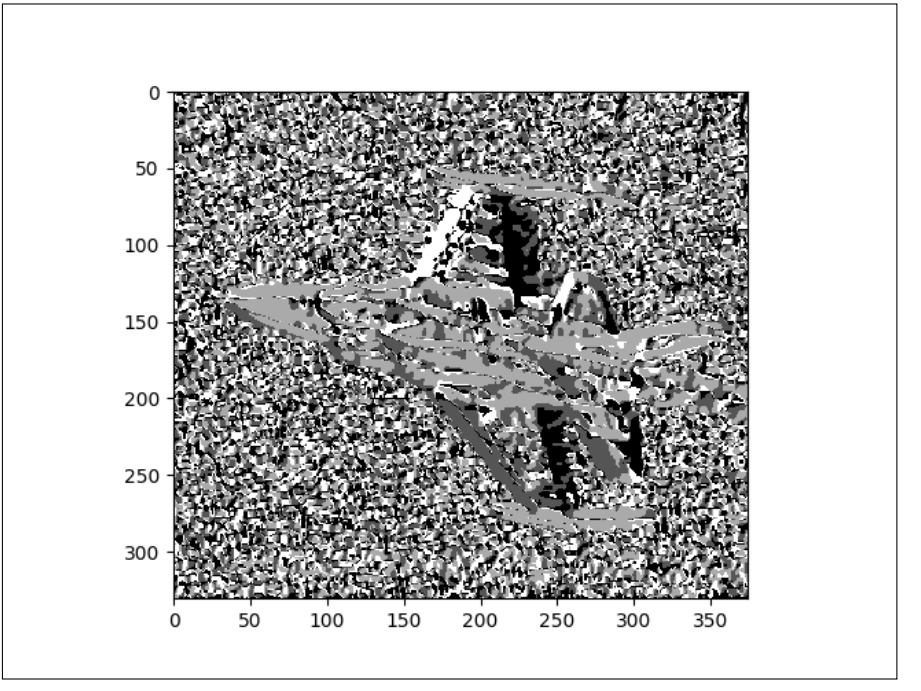

Figure 15: Plane - Canny 3.2 : Gradient Direction Image (Quantized)

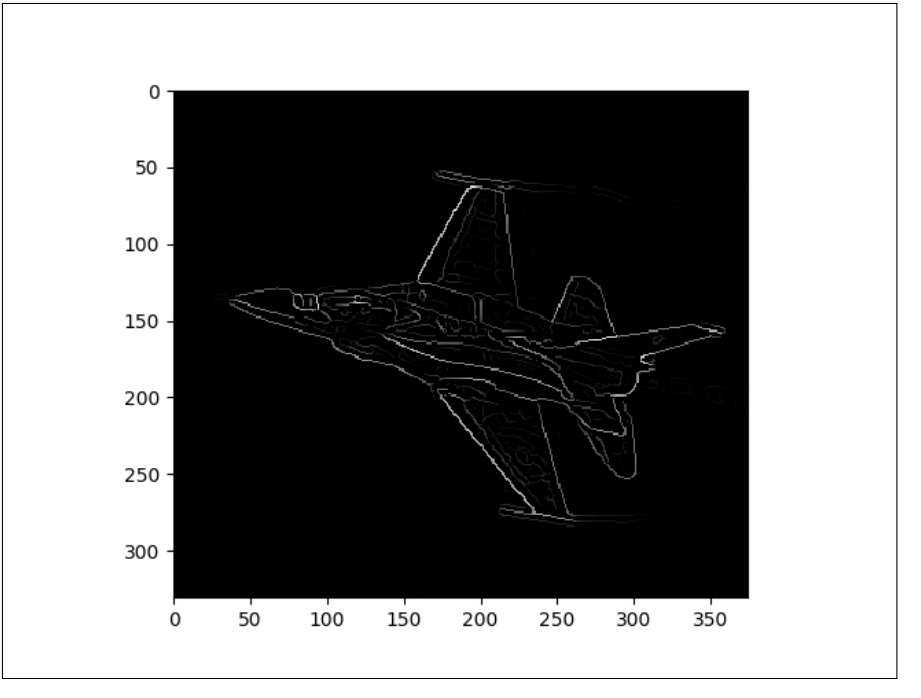

Figure 16: Plane - Canny 3.3 : Thinned Image (Using NMS)

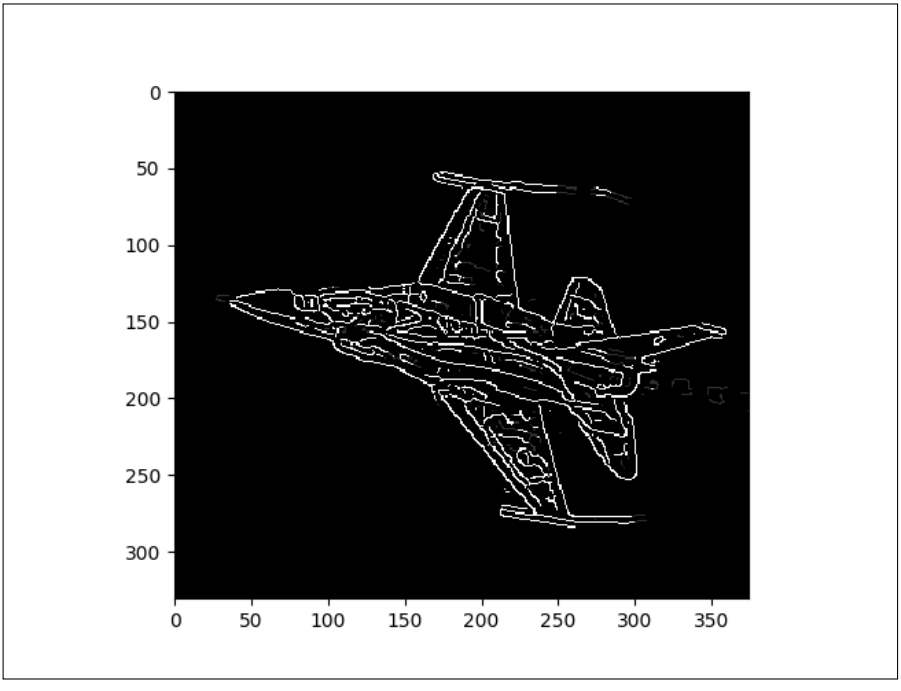

Figure 17: Plane - Canny 3.4 : Thresholded Image

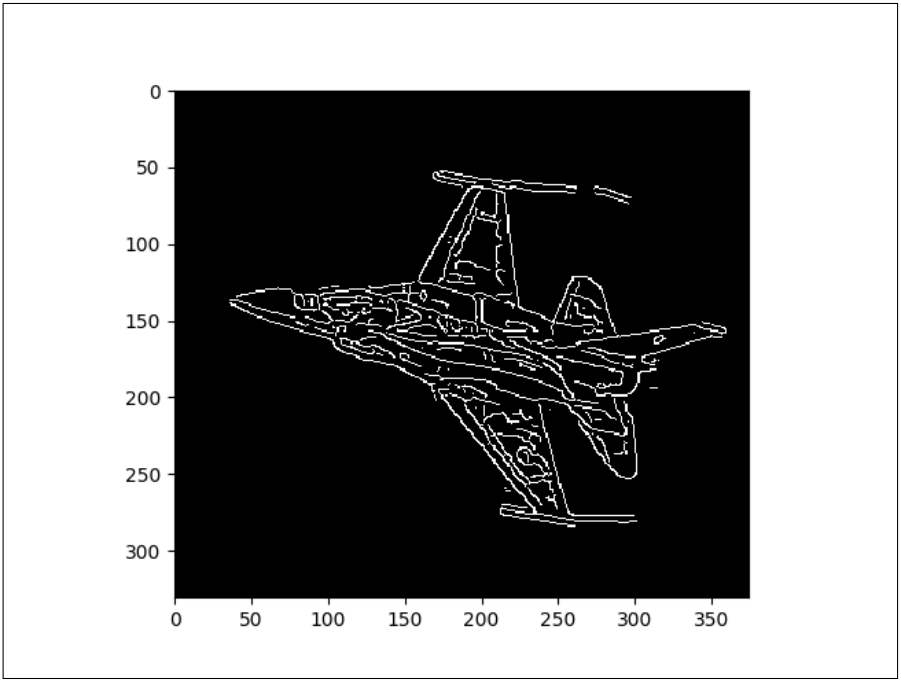

Figure 18: Plane - Canny 3.5 : Final Image after Linking

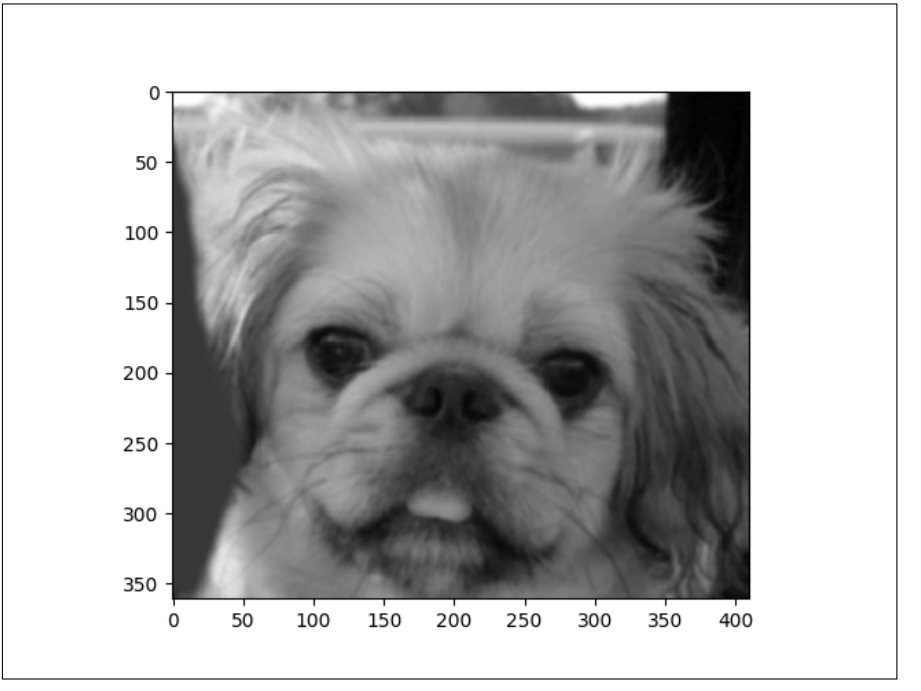

Figure 19: Dog - Canny 4.0 : Original Image

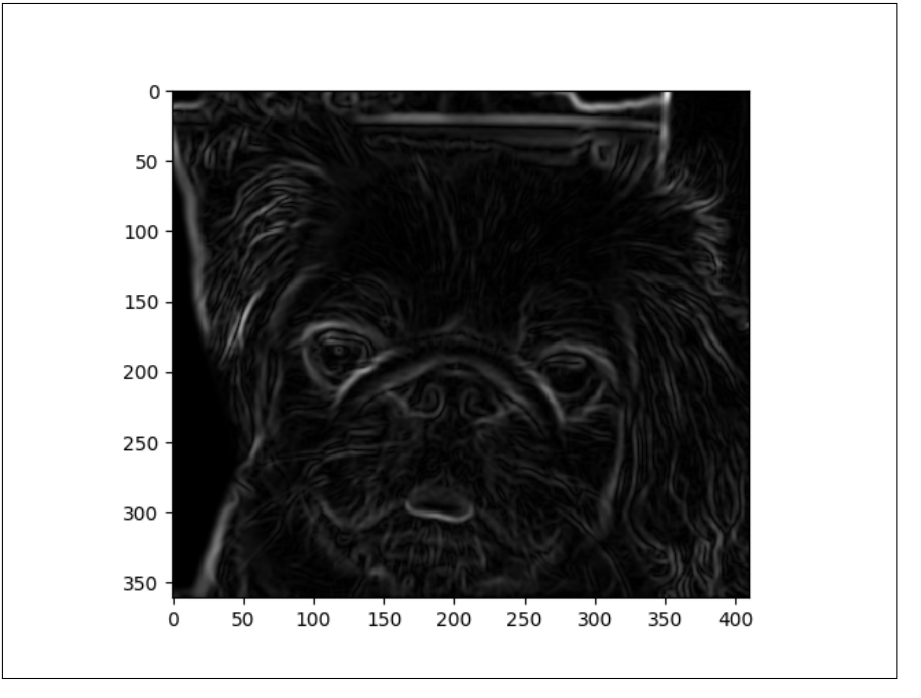

Figure 20: Dog - Canny 4.1 : Gradient Magnitude Image

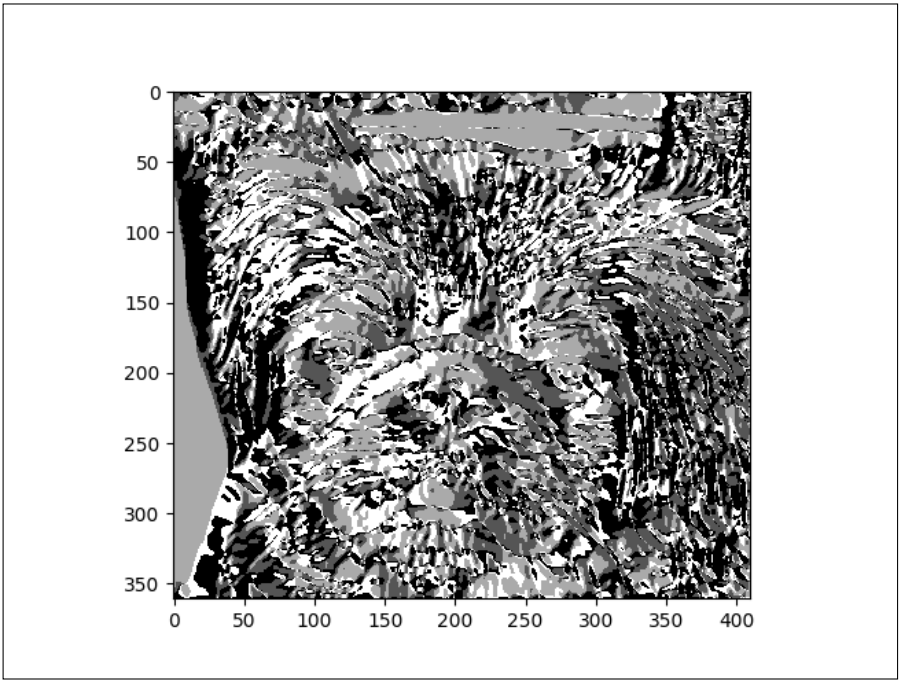

Figure 21: Dog - Canny 4.2 : Gradient Direction Image (Quantized)

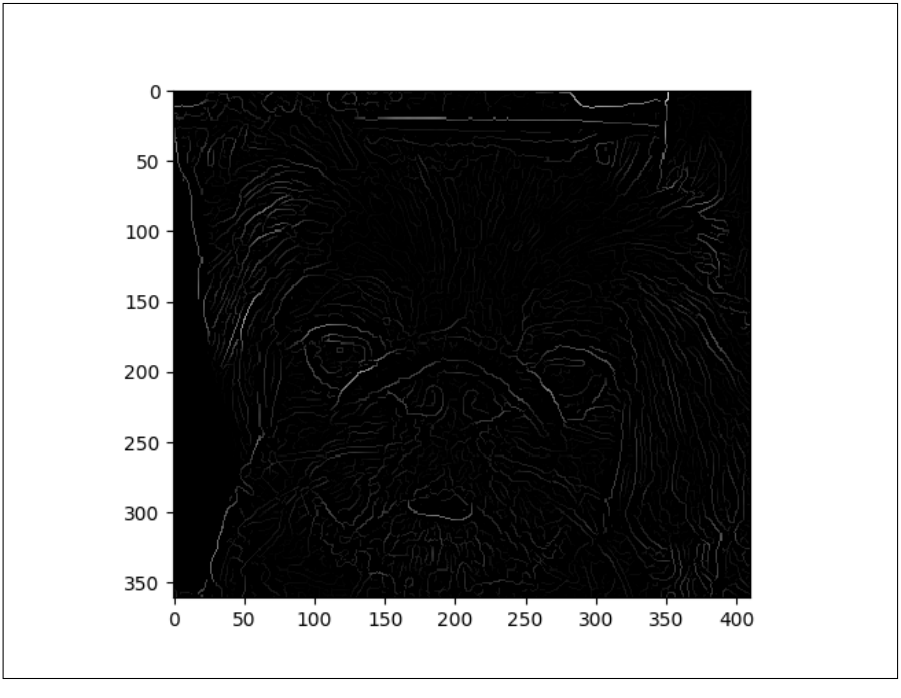

Figure 22: Dog - Canny 4.3 : Thinned Image (Using NMS)

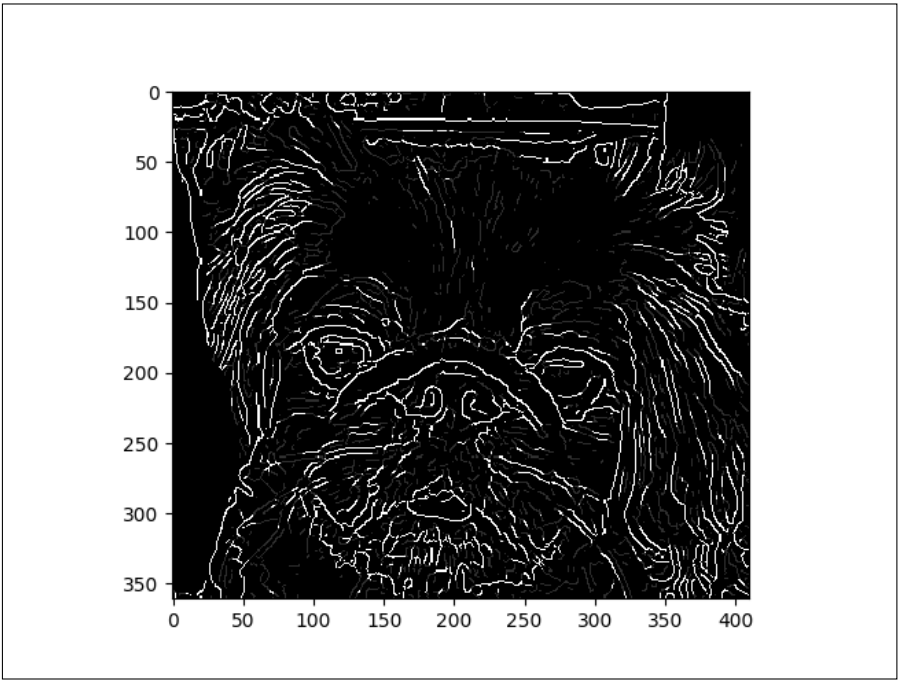

Figure 23: Dog - Canny 4.4 : Thresholded Image

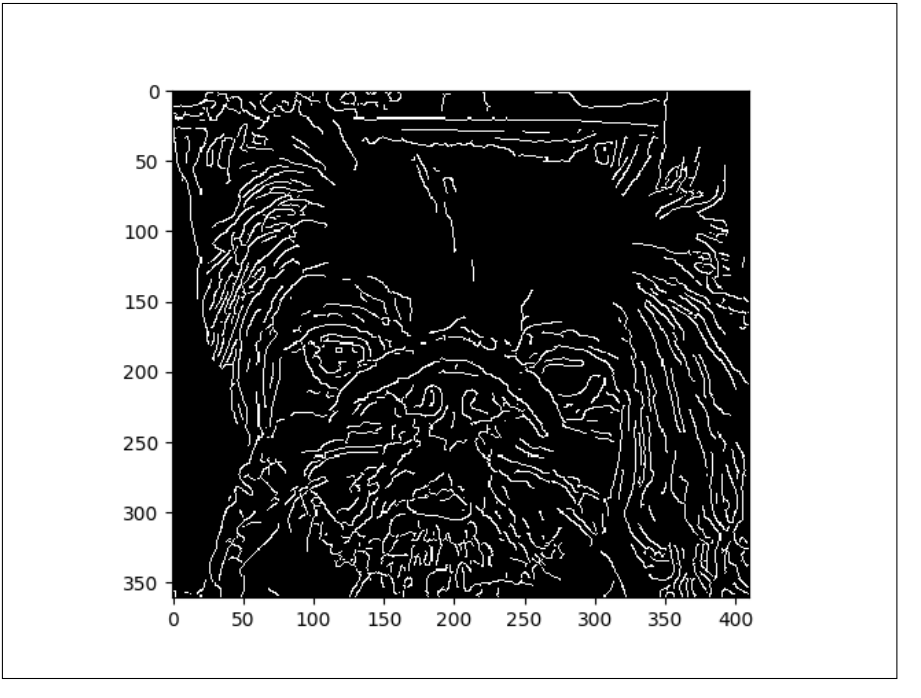

Figure 24: Dog - Canny 4.5 : Final Image after Linking

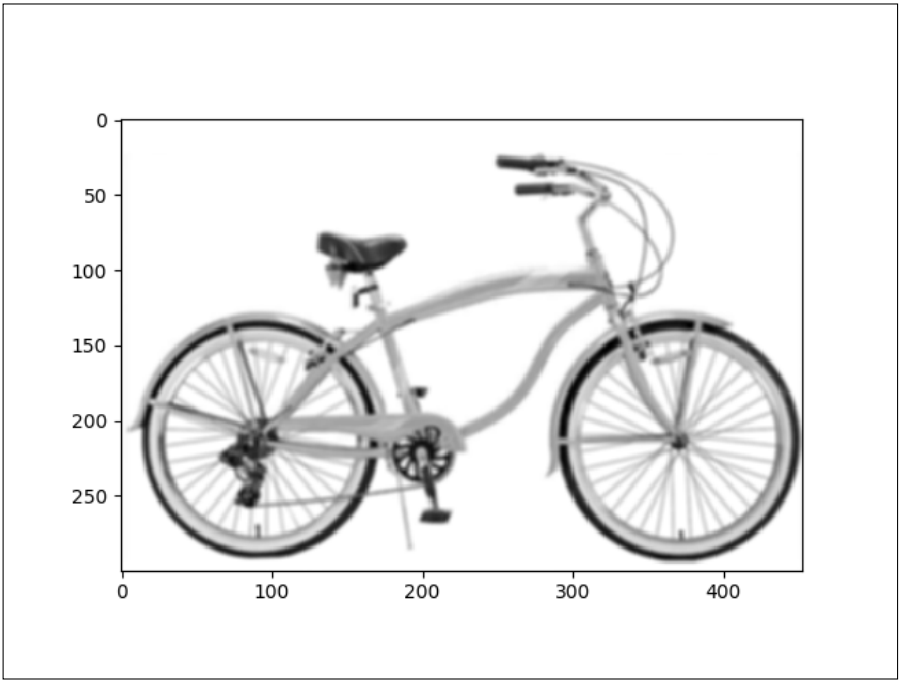

Figure 25: Bicycle - Canny 5.0 : Original Image

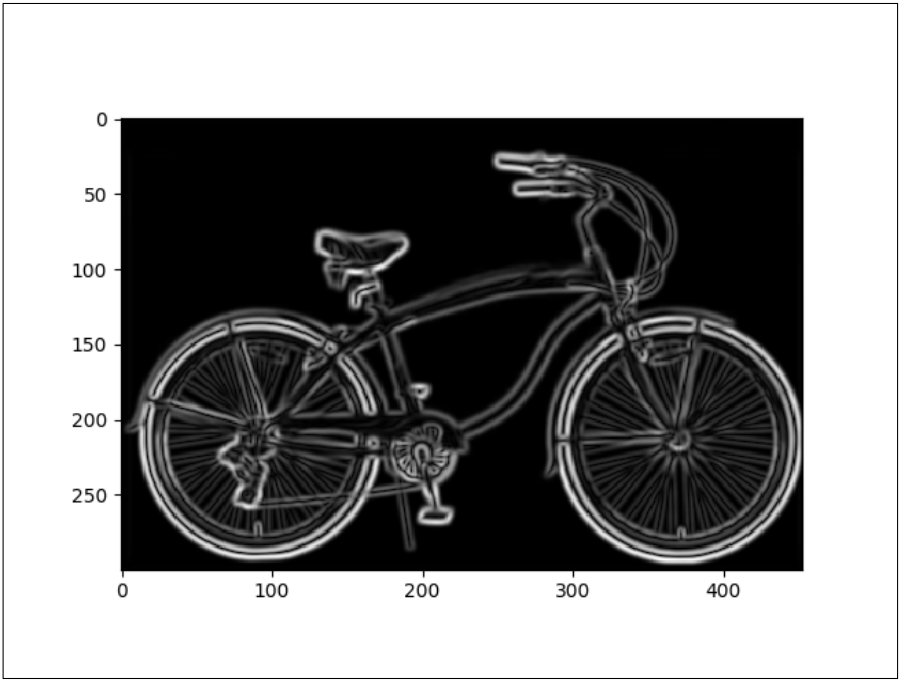

Figure 26: Bicycle - Canny 5.1 : Gradient Magnitude Image

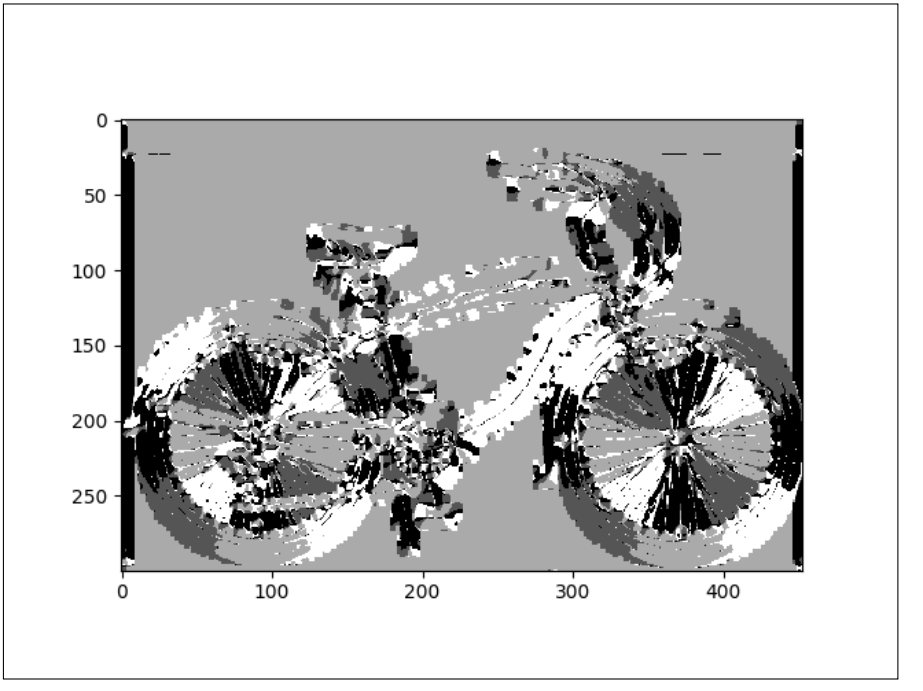

Figure 27: Bicycle - Canny 5.2 : Gradient Direction Image (Quantized)

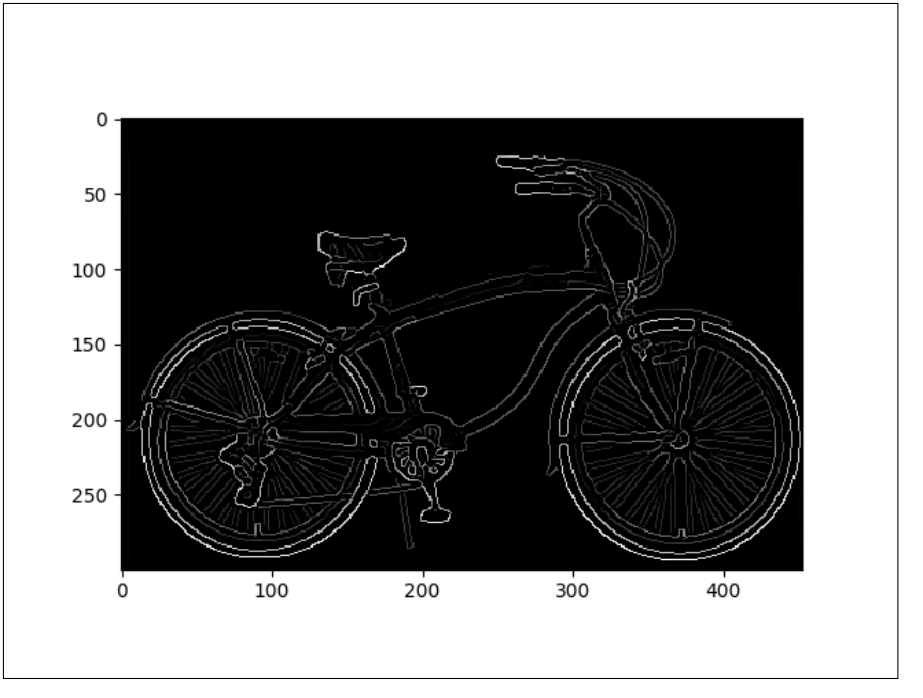

Figure 28: Bicycle - Canny 5.3 : Thinned Image (Using NMS)

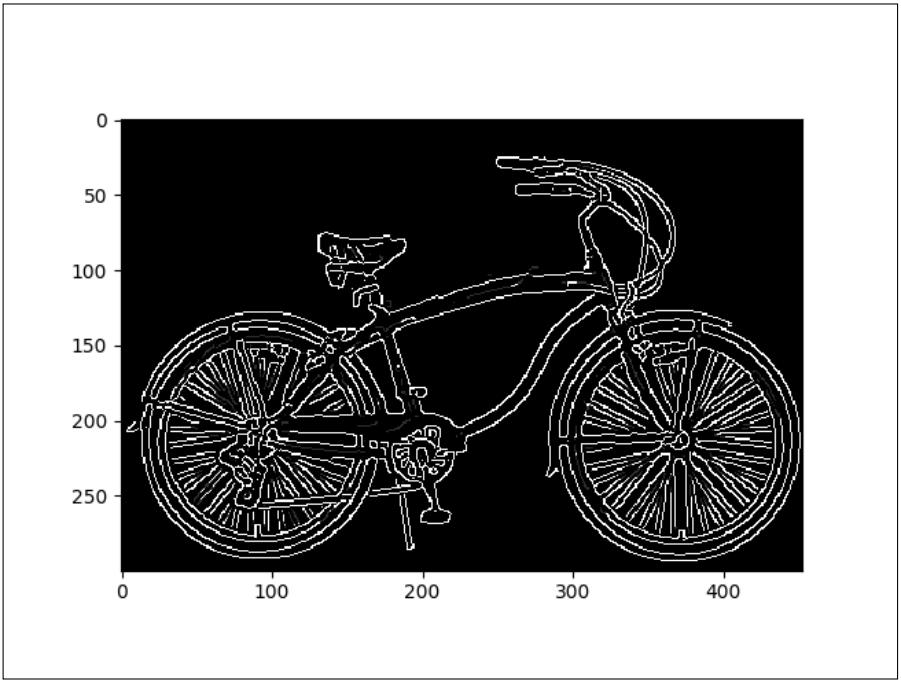

Figure 29: Bicycle - Canny 5.4 : Thresholded Image

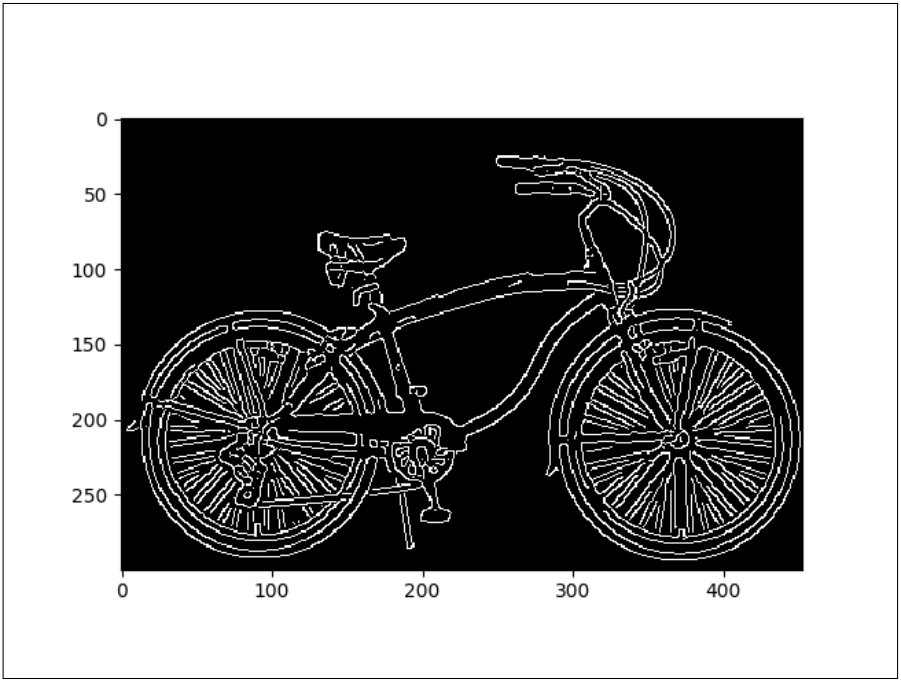

Figure 30: Bicycle - Canny 5.5 : Final Image after Linking

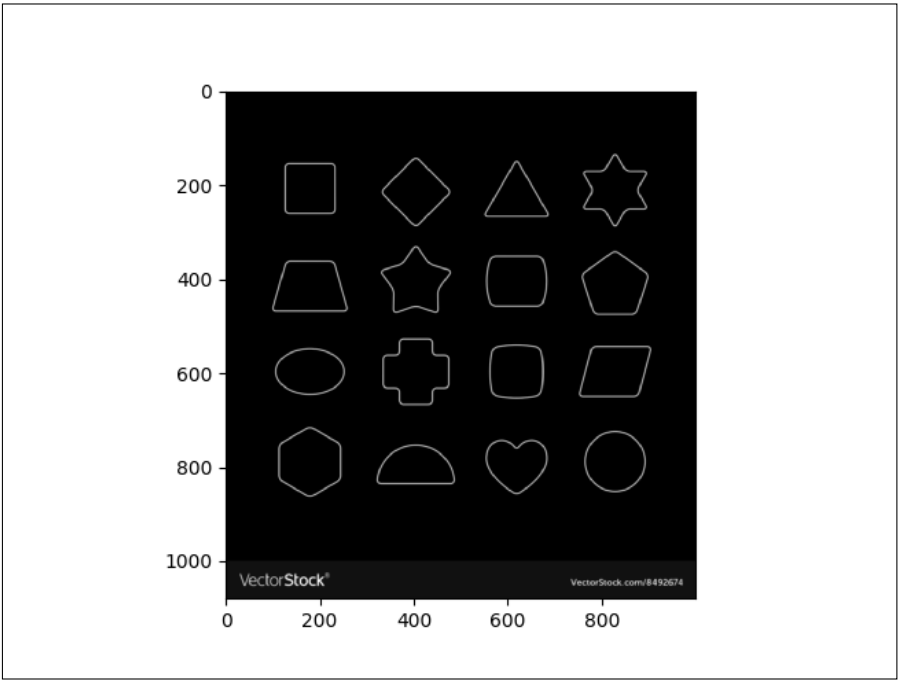

Figure 31: Toy - Canny 6.0 : Original Image

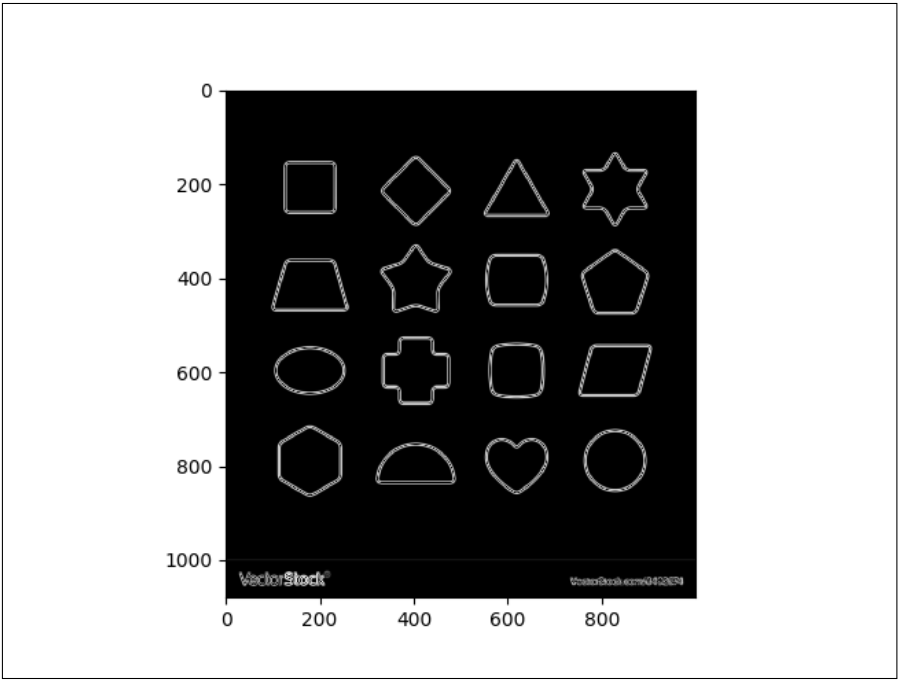

Figure 32: Toy - Canny 6.1 : Gradient Magnitude Image

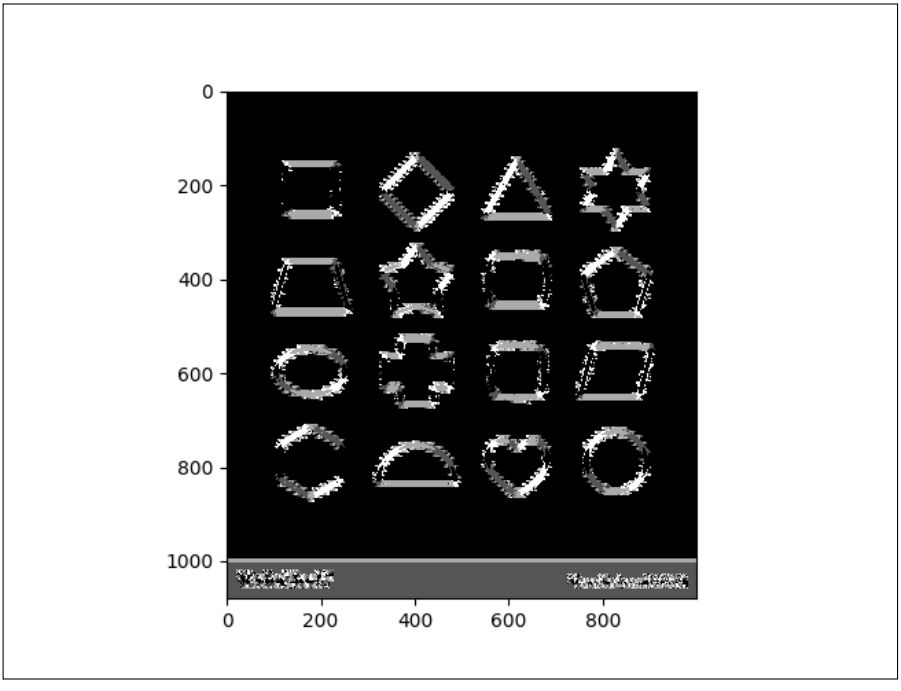

Figure 33: Toy - Canny 6.2 : Gradient Direction Image (Quantized)

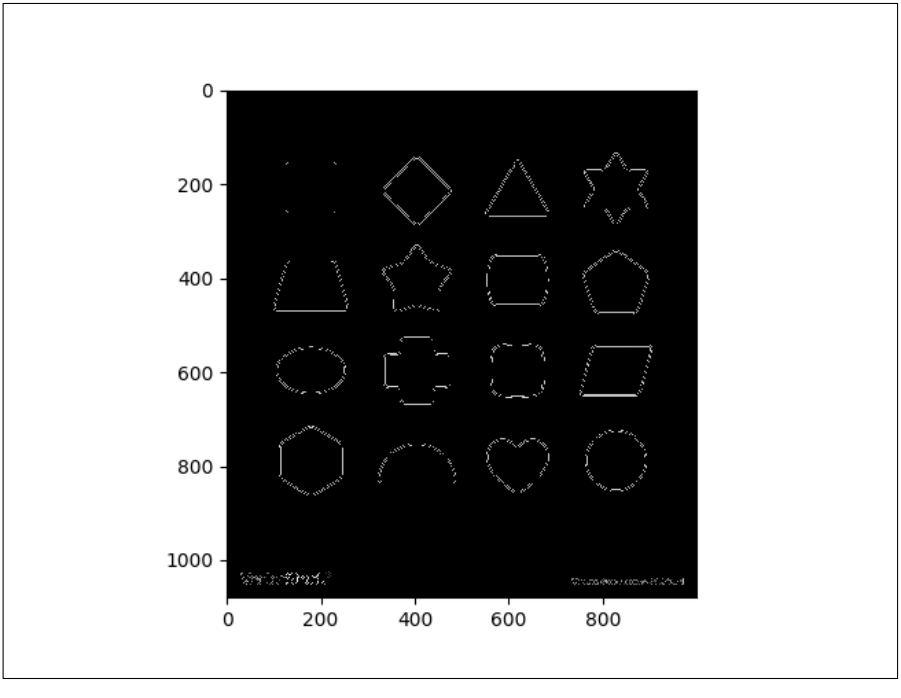

Figure 34: Toy - Canny 6.3 : Thinned Image (Using NMS)

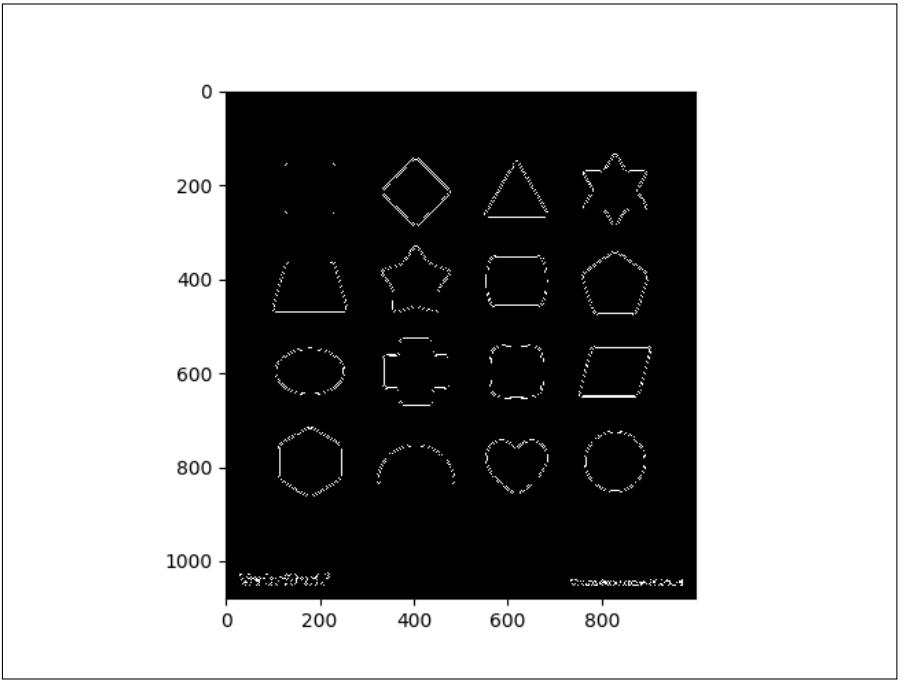

Figure 35: Toy - Canny 6.4 : Thresholded Image

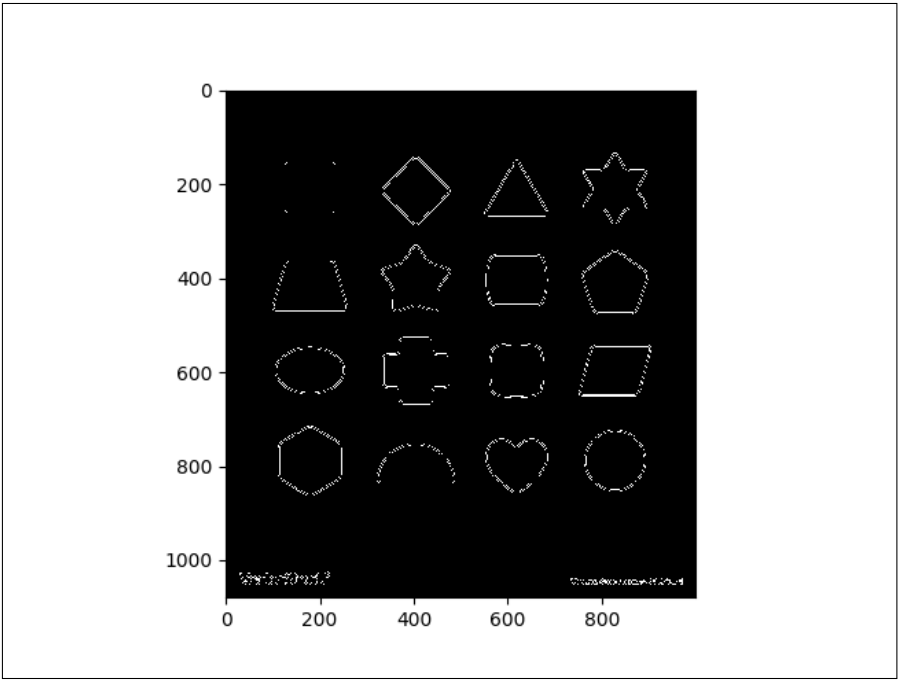

Figure 36: Toy - Canny 6.5 : Final Image after Linking

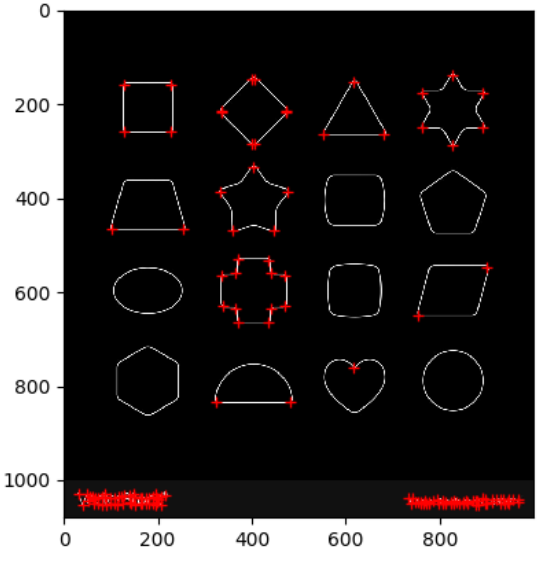

Figure 37: Toy - Harris Corner

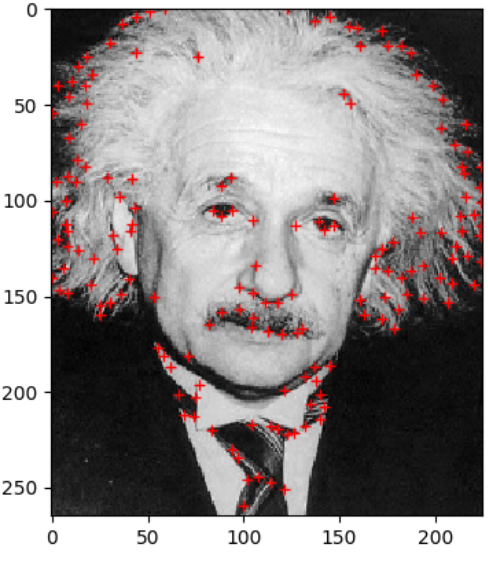

Figure 38: Einstein - Harris Corner

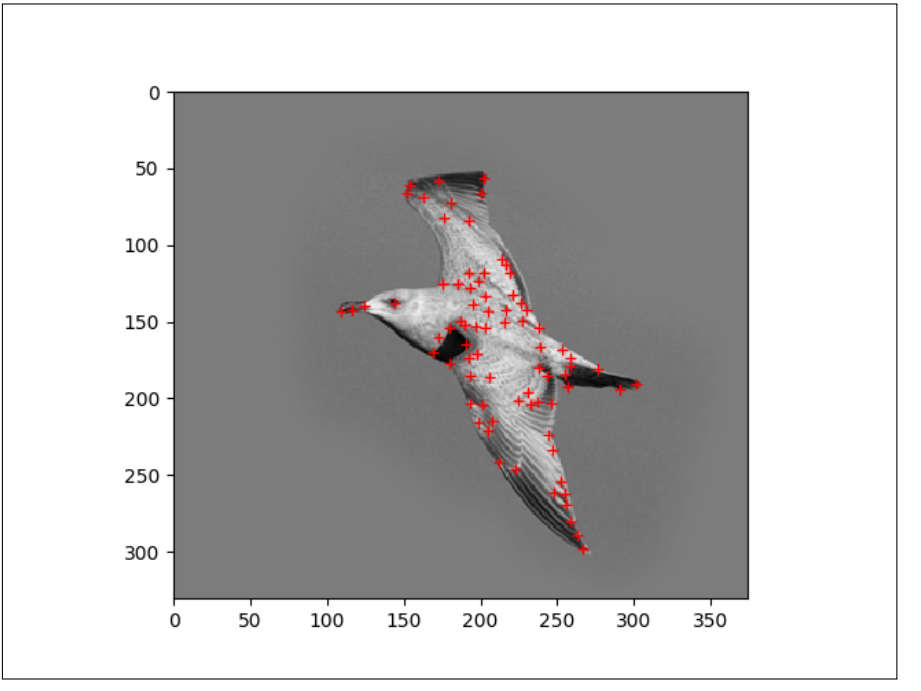

Figure 39: Bird - Harris Corner

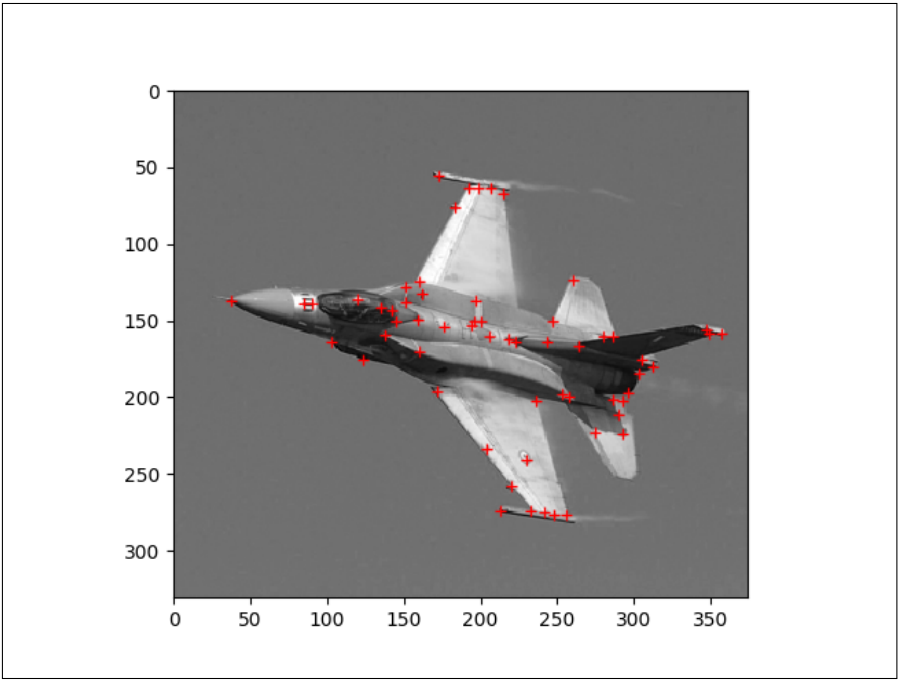

Figure 40: Plane - Harris Corner

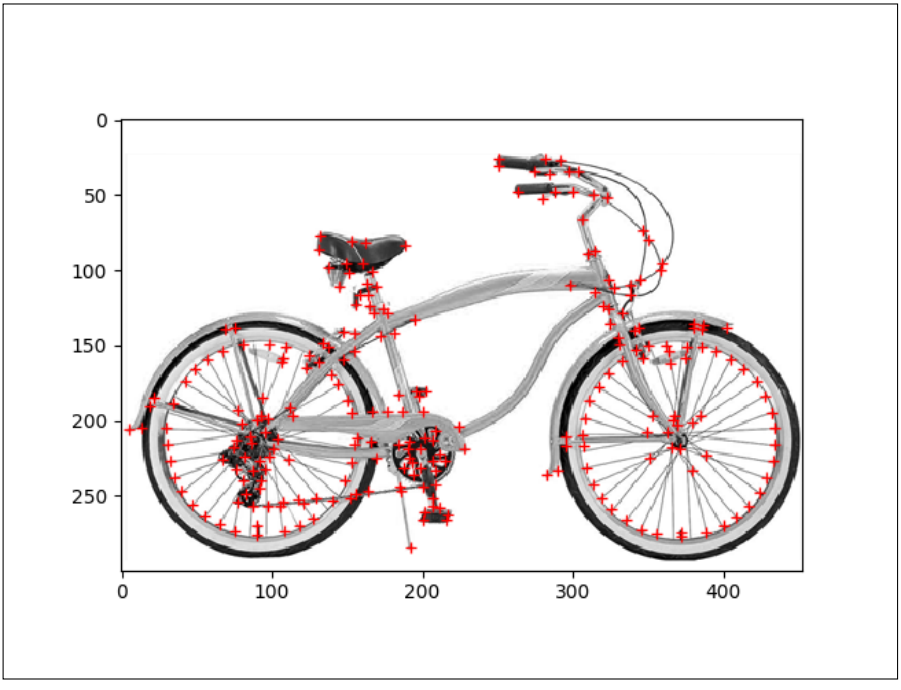

Figure 41: Bicycle - Harris Corner

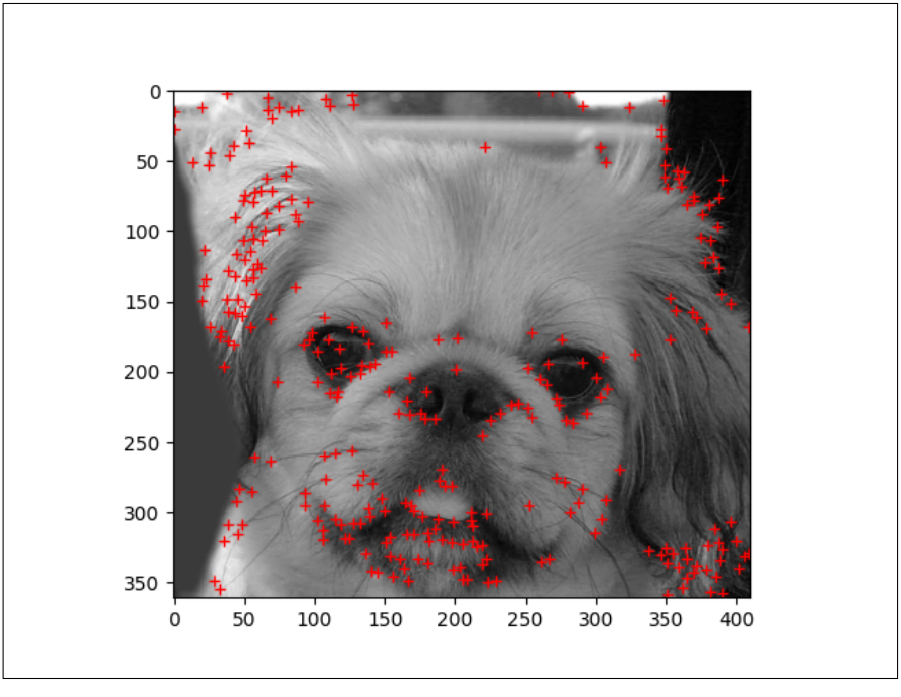

Figure 42: Dog - Harris Corner