University of Pennsylvania
CIS 581: Computer Vision and Computational Photography, Fall 2021

Project Nicholas Gurnard: Canny Edge Detection

Name: *Nicholas Gurnard*

Academic Honesty Declaration: By submitting this document, I certify that all solutions below are my own work in collaboration with my partner(s) listed below. I have listed below all individuals (other than the course instructor/TAs) with whom I have discussed any of the problems and/or solutions; nevertheless, while I may have discussed some of the problems/solutions with these collaborators, I have written this document independently, on my own.

Collaborators:

1. Ankit Billa – ankitb@seas.upenn.edu

Folder of Result Images:

This folder provides a series of images where the threshold values are adjusted to find the optimal image threshold values. Looking at the images within this folder will give a good intuition of the limitations of the canny edge detection algorithm (discussed more below). The value for the low threshold is always 0.4 times the high threshold. The naming convention for each image is "lowThresh_highThresh_image#.png".

`https://drive.google.com/drive/folders/1-61mW8nx_wGTRHjywjtCpgp8eq1LXxXE?usp=sharing`

Canny Edge Detection Algorithm:

There are 5 steps to the canny edge detection algorithm:

1. Filter images by the derivatives of the Gaussian kernel
   (a) The image is smoothed with the Gaussian kernel so that edges within the image are smoothed at the pixel level. This is done via a convolution of the image with the Gaussian kernel and its derivatives: $I \circledast dx \circledast G$. The filters for the gradient along the x and y directions and the Gaussian kernel G are as follows for this project:

$$dx = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

$$dy = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

$$G = \frac{1}{159} * \begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix}$$

2. Compute the magnitude of the gradient
   (a) The result of step 1 should give the the smoothed derivatives of the image in both the x (I_x) and the y (I_y) directions. The magnitude of the overall gradient of the image is then: $Magnitude = \sqrt{I_x^2 + I_y^2}$
3. Compute edge orientation
   (a) The gradient consists of 2 vectors I_x and I_y, so the orientation of the gradient is then arctan $\frac{I_y}{I_x}$
4. Detect local maximum
   (a) The edges that are smoothed from the Gaussian kernel are several pixels wide, so non-maximum suppression is used to clearly define a precise edge line. At the pixel level, if a set of pixels lie on an edge then only the greatest pixel value is preserved while the other pixel values are suppressed to zero.
5. Edge linking (hysteresis)
   (a) Hysteresis utilizes a set of 2 threshold values, high and low, so that edges that are caused by noise are filtered out of the image. The high threshold is used to preserve only the most prominent edges in the image while filtering out all of the noise, thus generating a strong edge map. However, a lot of real edges are broken with this step. So, a low threshold is used to create a map of "uncertain" edges who may or may not be due to noise. To check if these uncertain edges are cause by noise or are actually a part of a strong edge, the nearby pixels are analyzed. If the nearby pixels are part

of the strong edge map, then that uncertain pixel is likely an edge (not noise) and is added to the final edge map. This process is done iteratively until all uncertain edges connected to an existing strong edge are reinstated to the final edge map.

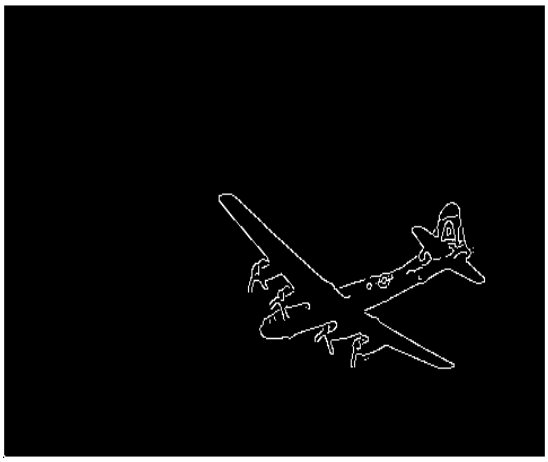
Limitations of Canny Edge Detection:

Canny edge detection is a powerful edge detection algorithm when assessing a few images, however it has some limitations. For each image, there is a set of 2 thresholds that must be optimized in order to get a sensible result. Canny edge detection cannot be a "catch all" for a large set of images, because the thresholds are distinct for each image and must be manually tuned for the best results.

In addition, canny edge detection struggles with edges that are soft or do not have enough contrast with the surrounding pixels. For example, edges that exist in a shadowy region in an image often are not detected because the pixels are not "different" enough to be considered a strong edge. If the pixels' values are too similar, it will struggle to consider these softer edges a distinct edge, especially if one does not want to introduce a lot of noise in the image.

Image 1 – Image #3096

(a) Original Image



(b) Image After Canny Edge Detection

Figure 1

Threshold Values:
Low = 16, High = 40

Observations:
For this image, the threshold values were tuned so that all of the noise in the image was removed at the expense of a good edge around the propellers. The overall edge of the airplane was solid.

Image 2 – Image #16068

(a) Original Image (b) Image After Canny Edge Detection

Figure 2

Threshold Values:
Low = 20, High = 50

Observations:
For this image, the threshold values were tuned so that much of the noise in the image was removed at the expense of a good edge around the some of the face and back. The overall edges were good, however there was a trade-off between less noise in the grass and the background with some of the edges on the zebra.

Image 3 – Image #21077

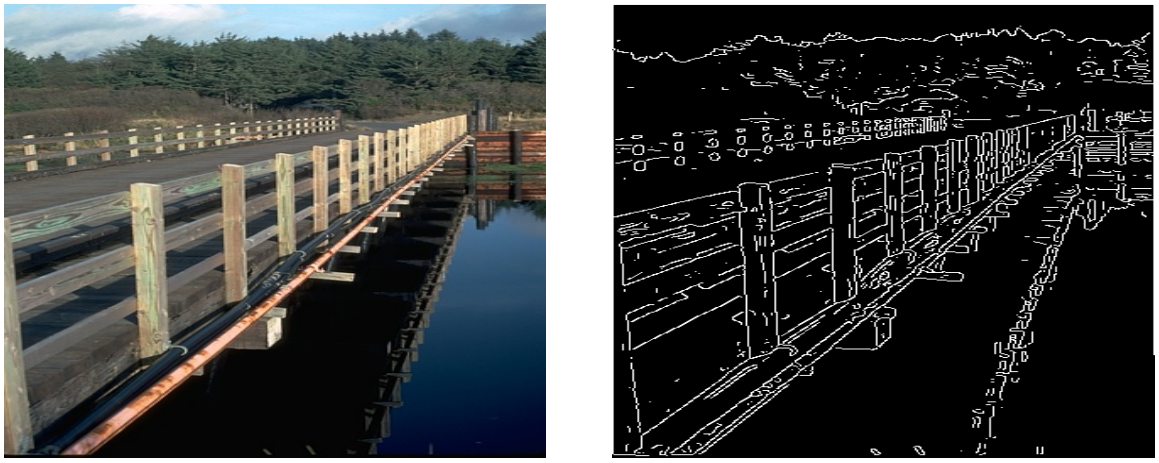
(a) Original Image (b) Image After Canny Edge Detection

Figure 3

Threshold Values:
Low = 20, High = 50

Observations:
For this image, the threshold values were tuned so that much of the noise in the road was removed without sacrificing good edges for the lettering in the background. The edges around the care and its shadow was very solid.

Image 4 – Image #22013



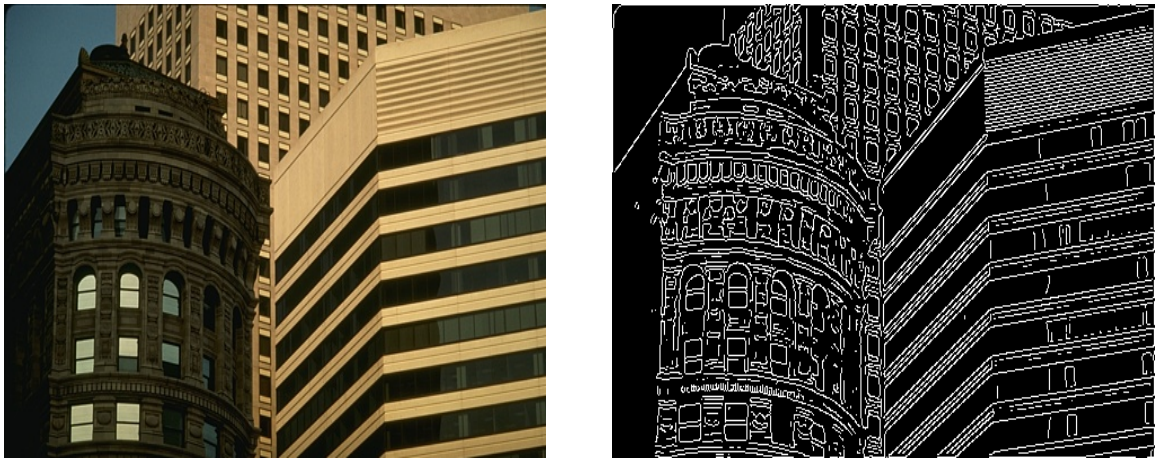
(a) Original Image (b) Image After Canny Edge Detection

Figure 4

Threshold Values:
Low = 22, High = 55

Observations:
For this image, the threshold values were tuned so that much of the noise in the background was removed to ensure the fence's edges in the foreground were prominent. As a consequence, some of the edges in the back fence were removed.

Image 5 – Image #48017



(a) Original Image (b) Image After Canny Edge Detection

Figure 5

Threshold Values:
Low = 12, High = 30

Observations:
For this image, the threshold values were tuned so that as many edges for windows and details on the building were preserved. If the high threshold was set very low, the shadowed side of the left building would have edges, however the rest of the image would become too noisy.

Image 6 – Image #55067

(a) Original Image (b) Image After Canny Edge Detection

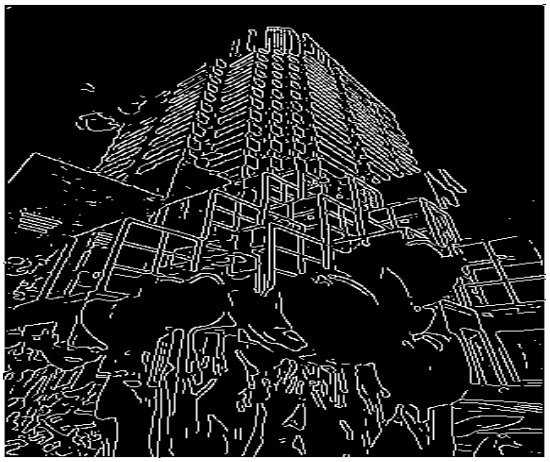
Figure 6

Threshold Values:
Low = 4, High = 10

Observations:
For this image, the threshold values were tuned so that the mountain edges were precise. However, some of the mountains did not have a significant contrast between the other in front of/behind it. In order to capture all mountain edges, the high threshold was set numerically low, but not too low such that there was a lot of noise.

Image 7 – Image #86000

(a) Original Image



(b) Image After Canny Edge Detection

Figure 7

Threshold Values:
Low = 16, High = 40

Observations:
For this image, the threshold values were tuned so that the very prominent building edges were preserved. The edges that were against the sky in the top left and on the very right of the building were difficult to preserve without introducing too much noise. This is because the foreground has many small edges, so making the high threshold too low would result in a busy foreground for only a few edges on the building, which was not a worthy trade-off.

Image 8 – Image #118035

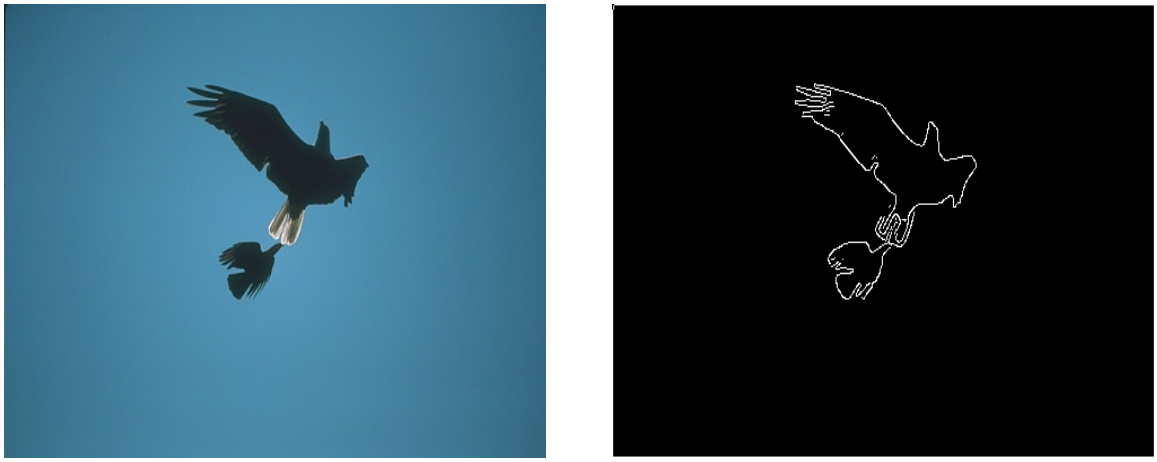
(a) Original Image (b) Image After Canny Edge Detection

Figure 8

Threshold Values:
Low = 2, High = 5

Observations:
For this image, the threshold values were tuned very low since the original image is very washed out. Most edges are very strong, however intricate details are hard to pick up.

Image 9 – Image #135069



(a) Original Image (b) Image After Canny Edge Detection

Figure 9

Threshold Values:
Low = 10, High = 25

Observations:
For this image, the threshold values were tuned so that the edges on the tail were as detailed as possible. The stark contrast between the birds and the sky made the edges very precise.

Image 10 – Image #189080

(a) Original Image (b) Image After Canny Edge Detection

Figure 10

Threshold Values:
Low = 12, High = 30

Observations:
For this image, the threshold values were tuned so that most of the contours on the face had an edge (wrinkles, eyes, cheeks, etc.) There was a a small amount of noise in order to get all the details of the face. The chin was hardest to get an edge for because the contrast between the chin and the neck was not very big, meaning the pixels were similar in value.

Image 11 – Image #201080

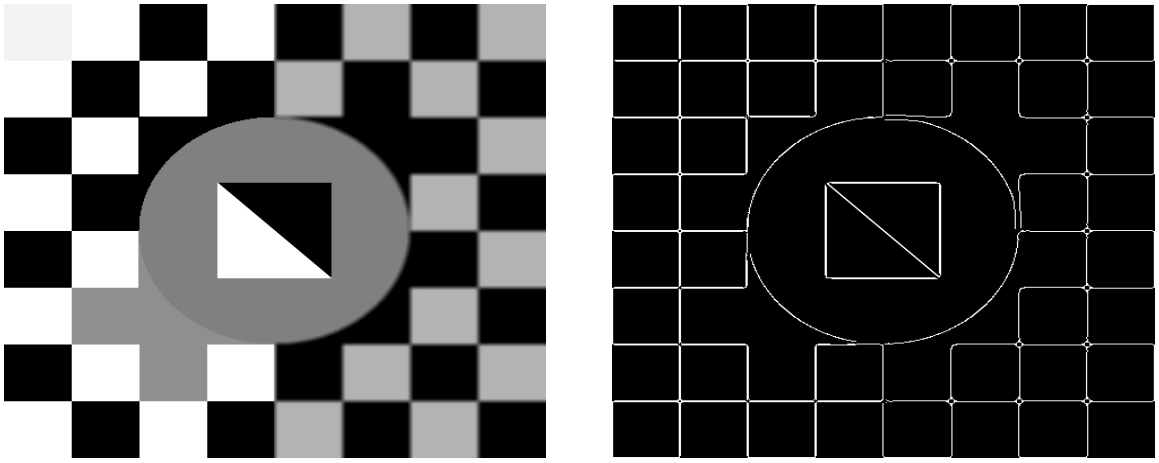
(a) Original Image (b) Image After Canny Edge Detection

Figure 11

Threshold Values:
Low = 16, High = 40

Observations:
For this image, the threshold values were tuned so that most of the building edges were solid. The very left side of the building had pixel values that were not highly contrasted from those of the sky, so it was difficult to maintain an edge there without introducing too much noise. Many edges in the grass were also preserved.

Image 12 – Image #I1



(a) Original Image (b) Image After Canny Edge Detection

Figure 12

Threshold Values:
Low = 8, High = 20

Observations:
For this image, the threshold values were tuned so that all possible edges were highlighted. Even less prominent images, like the top left grey box, were detected. The higher the contrast (ex: black vs. white), the better the edge. For this image, some of the edges were 2 pixels wide because of the way the Gaussian smoothing works. When convolving the image with dx or dy in the x and y directions, the result will be an edge 2 pixels wide with the *same value*. Thus, when you perform non-maximum suppression, the 2 pixel edge will not always be reduced to a 1 pixel edge (depending on the orientation of the gradient) since they are the *same value*, and thus neither edge will be suppressed. This is an edge case in the canny edge detection algorithm that, in practice, is highly unlikely in organic images. Even then, a 2 pixel wide edge is still precise.