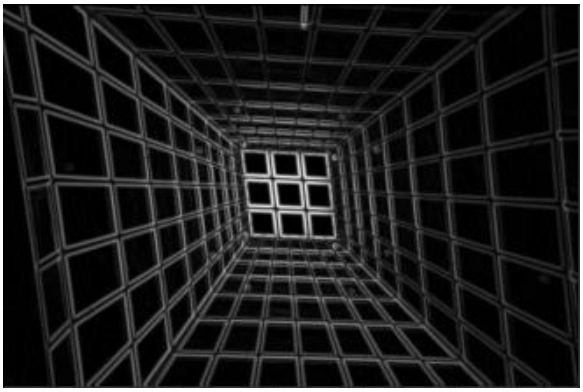
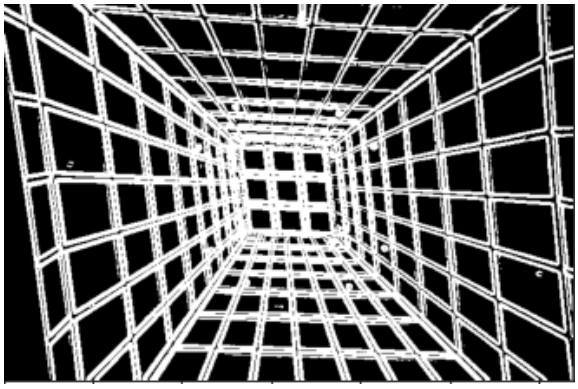
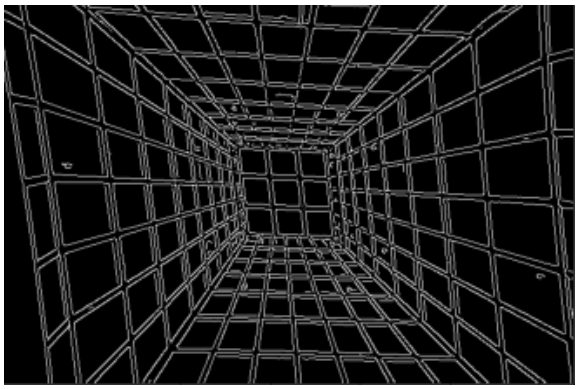






ELEC 474

Prelab 3 – Edge Detection

General Edge Detection

Kevin Huang and Michael Greenspan

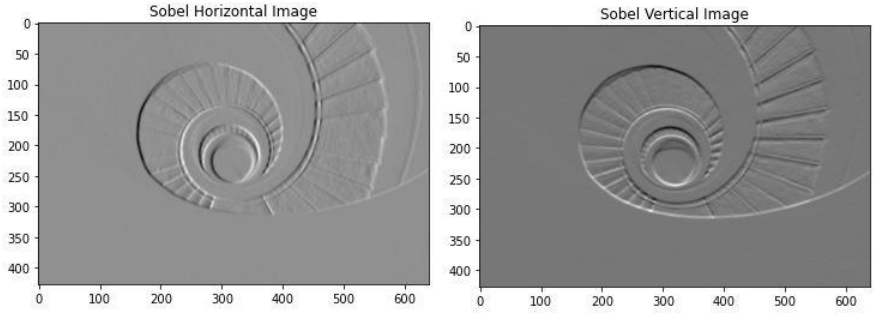
Contents

1. General Edge Detection 1
2. Canny Edge Detection 2
3. Comparison of General and Canny Edge Detectors 3
4. Requirements and Notes 3
5. Submission 3

1. General Edge Detection

For this prelab you will be implement the General Edge detection method from Slide 6 of the Edge Detection lecture slides. The algorithm is as follows:

1. Smooth your input image $\boldsymbol{I}$ using the **GaussianBlur()** method.
2. Define your orthogonal kernel masks. For this prelab use **Sobel's** Edge Detection Masks.
3. Convolve both your kernel masks over your input image using **filter2D()** to retrieve the $\boldsymbol{x}$- and $\boldsymbol{y}$-Gradient images $\boldsymbol{I_x}$ and $\boldsymbol{I_y}$. As an example, here are the Sobel kernel's applied to "staircase.jpg":



$\boldsymbol{G_x}$ $\boldsymbol{G_y}$

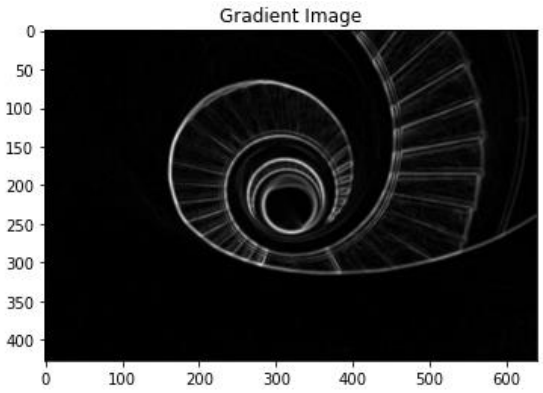
4. Now find your **Gradient Magnitude** $\boldsymbol{G(i,j)}$ of your image by combining both the $\boldsymbol{x}$-derivative $\boldsymbol{I_x}$ and $\boldsymbol{y}$-derivative $\boldsymbol{I_y}$ images. This can be estimated using:

$$G(i,j) = \sqrt{I_x^2(i,j) + I_y^2(i,j)}$$

$$OR$$

$$G(i,j) = |I_x(i,j)| + |I_y(i,j)|$$

Here's the Gradient magnitude image $\boldsymbol{G(i,j)}$ on "staircase.jpg":

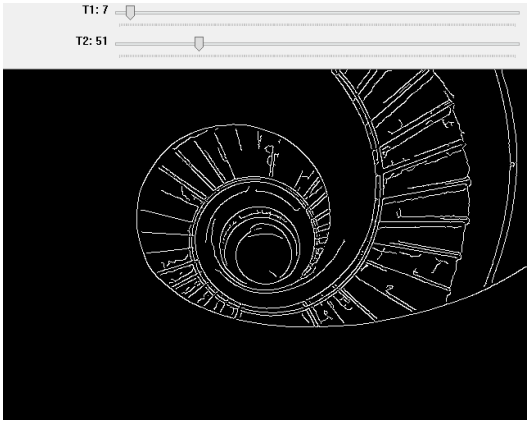


5. Threshold your gradient image by marking all pixels in $\boldsymbol{I}(i,j)$ as edges if $\boldsymbol{G}(i,j) > \boldsymbol{\tau}$.

Here the output edge image of "staircase.jpg" with $\boldsymbol{\tau} = 18$:

2. Canny Edge Detection

For this part you will be using OpenCV's **Canny()** function to perform Canny Edge Detection. Follow these steps to perform edge detection. Use **cv2.Canny()** function with both bottom and top bound thresholds ($\boldsymbol{\tau_1}, \boldsymbol{\tau_2}$) to retrieve your edge image. Here's an example of Canny executed on the "staircase.jpg" edge image with threshold values $\boldsymbol{\tau_1} = 7$ and $\boldsymbol{\tau_2} = 51$.



3. Comparison of General and Canny Edge Detectors

Compare the output of the General and Canny edge detectors on the staircase image. Define and implement a metric that indicates the ratio of the number of similar "edge" and "non-edge" pixels in both images. For example, if both images were identical, the metric should return a value of 1, and if half of the pixels were identical, it should return a value of 0.5.

Plot the value of your metric, for the selected (static) values of the Canny ($\boldsymbol{\tau_1}, \boldsymbol{\tau_2}$) thresholds, as the General $\boldsymbol{\tau}$ threshold varies.

4. Requirements and Notes

For this prelab your program must:

- Implement threshold **trackbars (createTrackbar())** for both Binary and Canny Edge detector to update your edge image live in a **cv2.namedWindow()**.
- When updating your threshold(s) and image using a **trackbar**; be sure to declare them as Python's **global** type so they can be used outside of your **trackbar** function.
- When double thresholding, be sure to **pre-define** your threshold values and you should update your canny image after **trackbar** usage using a separate function
- **OUTPUT** a plot of the similarity metric between the General and Canny Edge detection methods as the General $\boldsymbol{\tau}$ threshold varies.

5. Submission

The submission for this lab should include a .zip of:

- .ipynb file that:
  - Your code General and Canny Edge Detection using Trackbars
  - Tested your code on "staircase.jpg", "lines.jpg", and an image of your choosing
  - Output edge images **WITH their thresholds** noted (commented) of "staircase.jpg", "lines.jpg", and an image of your choosing
  - A plot of the similarity metric for the "staircase.jpg" image.
- An image of your choosing

Your code will be run in Jupyter Lab to test for functionality.

The marking rubric is as follows:

Item	mark
1. General Edge Detection method correct and fully functional	1
2. Comparison metric correct	1
3. Plot of comparison metric correct	0.5
4. Submission format correct	0.5
Total:	3