Canny Edge Detection Algorithm Explained

Furkan Atak

Abstract—This paper aims to describe the methods used in canny edge detection algorithm which is a state of the art algorithm based on the paper[1] " A Computational Approach to Edge Detection" published in 1986 by John Canny himself. Mathematical background of the algorithm is enlightened without diving into much detail. Finally, a python code snippet is provided to visualize and track the steps used in the algorithm.

Index Terms—Edge Detection, Canny, Algorithm, Image Processing, Computer Vision

1 Introduction

Edge detection is an important part of an computer vision system. It both reduces the amount of data to be processed and provides useful structural data. This make edge detection a must-to-use image processing in variety of applications such as machine learning. Canny edge detection algorithm follows 5 steps in order:

- Gaussian filter
- Gradient calculation
- Non-Maximum Suppression
- Double threshold
- Edge tracking by hysteresis

Each step is explained in a subsection. After each process a stock image from skimage.data module is provided to visualise the after effect. Algorithm is applied on a gray-scaled version of the selected image. Gray-scaling can reduce the amount of channel of the RGB image according to formula:

$$Y' = 0.2989 * R + 0.5870 * G + 0.1140 * B$$

August 19, 2022

1.1 Gaussian Filter

As a part of the algorithm Gaussian filter is used since at the later steps gradients are taken. The noise on the image disturbs the gradient matrix. To mitigate the this effect Gaussian filter is applied. Smoothing the image applying Gaussian filter, Gaussian kernel is convoluted with the image. The equation for a Gaussian filter kernel of size (2k+1)×(2k+1) is given by:

$$H_{ij} = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{(i-(k+1))^2 + (j-(k+1))^2}{2\sigma^2}\right)$$

$$; 1 \leq i, j \leq (2k+1)$$

So in essence, Gaussian kernel is a Gaussian distribution which the origin is at the center of the square 2-D matrix.

As can be seen from the fig1, matrix entry values are distributed in Gaussian. The center pixel is white meaning high in intensity and around edges it gets fainted meaning when kernel is convoluted with the image the current pixel value smoothed out with neighbour pixels therefore reducing sharp changes in pixel values(noise)

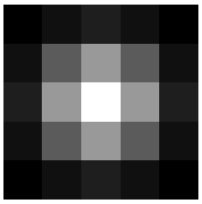


Fig. 1: 5x5 Gaussian filter kernel

1.2 Gradient Calculation

Gradient calculation allows algorithm to detect the edge pixels. Since at the edge points the change in the pixel values are higher than the change in the contexture in the object itself. This is the main idea for edge detection. At later steps algorithm also defines threshold values for those gradients to select the pixel either as an edge or not.

Gradient operation is applied both in x and y direction using Sobel filter kernels

$$K_x = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}, K_y = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$$

The convolution of kernel with the image gives I_x and I_y 2-D matrices which have the same shape as the original image.Then the magnitude G and slope θ matrix of the gradient are calculated as follows:

$$|G| = \sqrt{I_x^2 + I_y^2}$$

$$\theta(x, y) = arctan(\frac{I_y}{I_x})$$

(a) Gray-scaled

(b) Sobel Filtered

Fig. 2: Comparison of two image after applying sobel filter

1.3 Non-Maximum Suppression

As can be seen from the previous end result of the Sobel filtered image, edge thicknesses are not uniform. The purpose of this step is to convert the "blurred" edges in the image of the gradient magnitudes to "sharp" edges. Basically, this is done by preserving all local maxima in the gradient image and deleting everything else. The algorithm is for each pixel in the gradient image:

1) Compare the edge strength of the current pixel with the edge strength of the pixel in the positive and negative gradient directions.
2) If the edge strength of the current pixel is the largest compared to the other pixels in the mask with the same direction (e.g., a pixel that is pointing in the y-direction will be compared to the pixel above and below it in the vertical axis), the value will be preserved. Otherwise, the value will be suppressed.

When applying the algorithm for each pixel gradient direction θ values are rounded to the nearest 45°, corresponding to the use of an 8-connected neighborhood.

In more accurate implementations, linear interpolation is used between the two neighboring pixels that straddle the gradient direction. This is not the case for the code provided in the appendix. But skimage module uses linear interpolation.

The below figure shows what is meant by the algorithm clearly. Most of the gradient directions are rounded to the south-north direction. The pixel that is selected as an edge(marked with white borders) are local maxima in that direction. Other pixels are suppressed.

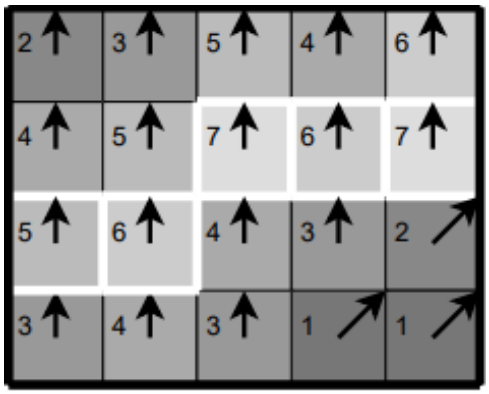
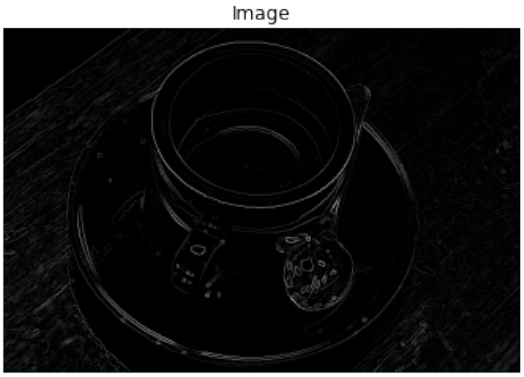


Fig. 3: Example image on how non-max suppression works

(a) Sobel Filtered



(b) Non-maximum suppressed

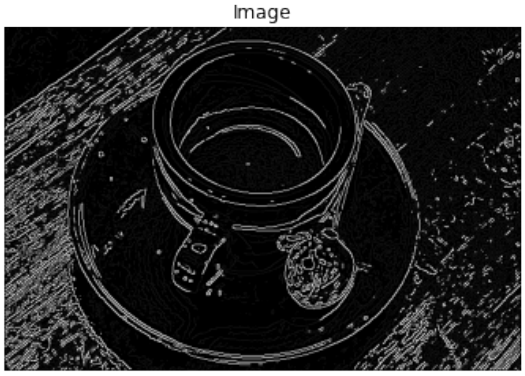
Fig. 4: Comparison of two image after applying non-max suppression

1.4 Double Threshold

After non-maximum suppression edge pixels have still non-uniform pixel intensity values. Even though probably most of the pixels that are selected to be edge are true ones, some of the edge pixels can be noise or color variations. To account for these spurious responses, edge pixels are categorized according to their intensity values. This is done by using double threshold meaning edge pixels stronger than some threshold are labeled as *strong*, and pixel values less than a lower threshold are labeled as *non-relevant*. These

pixels are suppressed. Pixel values that are between these threshold values are labeled as *weak* to be decided whether to be suppressed or preserved at the later step. Threshold values are set empirically.

(a) Non-max suppressed



(b) Double threshold-ed

Fig. 5: Comparison of two image after applying double threshold.

1.5 Edge Tracking by Hysteresis

After the previous step, there are still weak pixel values to be determined whether as an edge or not. The main idea when deciding the faith of the weak pixels is to track the current strong edges. Since some edge lines are disconnected by weak pixels, they should be connected. Edge tracking can be implemented by BLOB-analysis (Binary Large OBject). The edge pixels are divided into connected BLOB's using 8-connected neighbourhood. BLOB's containing at least one strong edge pixel are then preserved, while other BLOB's are suppressed.

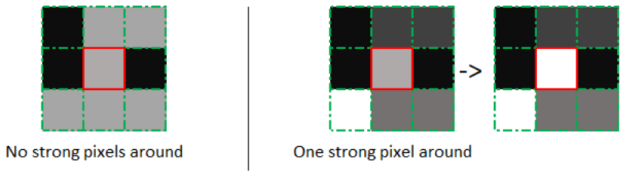
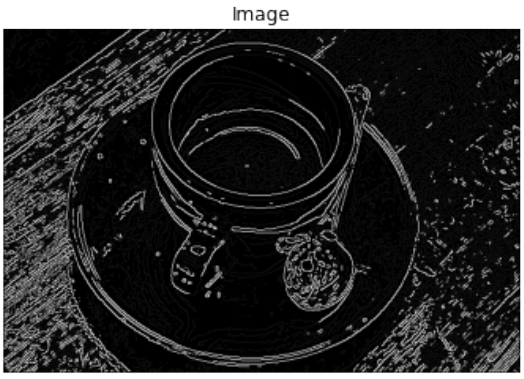
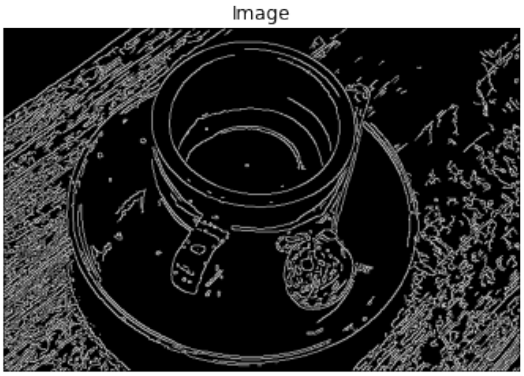


Fig. 6: Example image on how BLOB-analysis work



(a) Double threshold-ed



(b) Hysteresis applied

Fig. 7: Comparison of two image after applying BLOB analysis

2 Conclusion

After all steps are applied used in the canny edge detection algorithm gray-scaled image becomes a binary image. So that by successfully detecting edges, the algorithm reduced the amount of data at the same time supplied useful information about the image. Even though the Canny edge detection algorithm provides a relatively simple but precise methodology for the edge detection problem, many improvements are done to the algorithm which enhanced the accuracy and robustness of the detection. Improved versions can be studied for further analysis.

Appendix A

Applying the algorithm from scratch using python in jupyter notebook :
https://github.com/furkanatak/
Canny-Edge-Detection-Algorithm

References

[1] Canny, J., A Computational *Approach To Edge Detection*, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.
[2] https://www.cse.iitd.ac.in/~pkalra/col783-2017/canny.pdf
[3] https://en.wikipedia.org/wiki/Canny_edge_detector
[4] https://towardsdatascience.com/canny-edge-detection-step-by-step-in-python-