MSAI 495 MP5: Canny Edge Detector from scratch

Marthinus Johannes Nel

A Canny edge detector is a multi-stage algorithm that detects edges in images by applying a series of filters to reduce noise, calculate image gradients, suppress non-maximum edges, and using a high and low threshold with recursion to determine final edges.

Below is a description of the 6 functions used, in order, to implement canny edge detection:

1) **Gaussian smoothing**: This function preforms Gaussian smoothing on a gray scale image with a given kernel size and standard deviation. This reduces the noise in the image.

2) **Image gradient**: This function takes in the smoothed gray scale image and calculates the images gradient and magnitude. Sobel operators are used to compute the horizontal and vertical edges of the image (Ix, Iy). The magnitude of the image gradient is calculated by taking the square root of the sum of the squares of the horizontal and vertical edges and the direction of the image gradient is calculated using the arctan2 of the vertical and horizontal edges.

3) **Compute thresholds**: This function computes high and low thresholds based on the magnitude image of the image gradient. First a histogram is created from the magnitude image and then the cumulative histogram is generated and normalized. The first bin of the cumulative distribution that is greater than or equal to the percentage of non-edge pixels is determined to be the high threshold. The low threshold is calculated by multiplying the high threshold with the given ratio.

4) **Non-maximum suppression**: This is used to thin out the edges obtained from applying gradient-based edge detection. It works by iterating through each pixel in the magnitude image and comparing the gradient direction of the pixel to its neighboring pixels. The output is obtained by only keeping pixels whose magnitude is greater than or equal to its two neighboring pixels in the direction of the gradient.

5) **Find edges**: The algorithm loops through the suppressed image and assigns pixel values to each image based on the high and low threshold values. Pixels with values above the high threshold are considered strong edges, those between the high and low thresholds are considered weak edges, and those below the low threshold are not considered edges. The output images have different pixel values to differentiate between strong and weak edges.

6) **Edge linking**: The input to the function is an image containing both strong (255) and weak (125) edges. The function loops through each pixel in the image and checks if the pixel is a weak edge. If it is, it checks if any of its eight neighboring pixels has a strong edge. If it does, the weak edge is made strong by setting its value to 255. If not, the weak edge is deleted by setting its value to 0. The function returns an image containing only strong edges.

Results:

Figure 1 displays the output image of each function described above. (Figure 1: Top, Middle left) The Gaussian smoothing makes the image less sharp and thus reduces the edge noise. (Figure 1: Top, Middle right) The edges is then found by calculating the gradient of the image. The image magnitude displays the approximate edge locations. (Figure 1: Top, Right) Next the Non-Maxima suppression makes the edges in the magnitude tinner and thus the edges more clear. (Figure 1: Bottom left 3 images) The high and low threshold is then applied to the image and the edges are categorized in either strong or weak edges. (Figure 1: Bottom right) Lastly the weak edges are used to connect and extend the strong edges. This is the final result from the Canny edge detection algorithm.

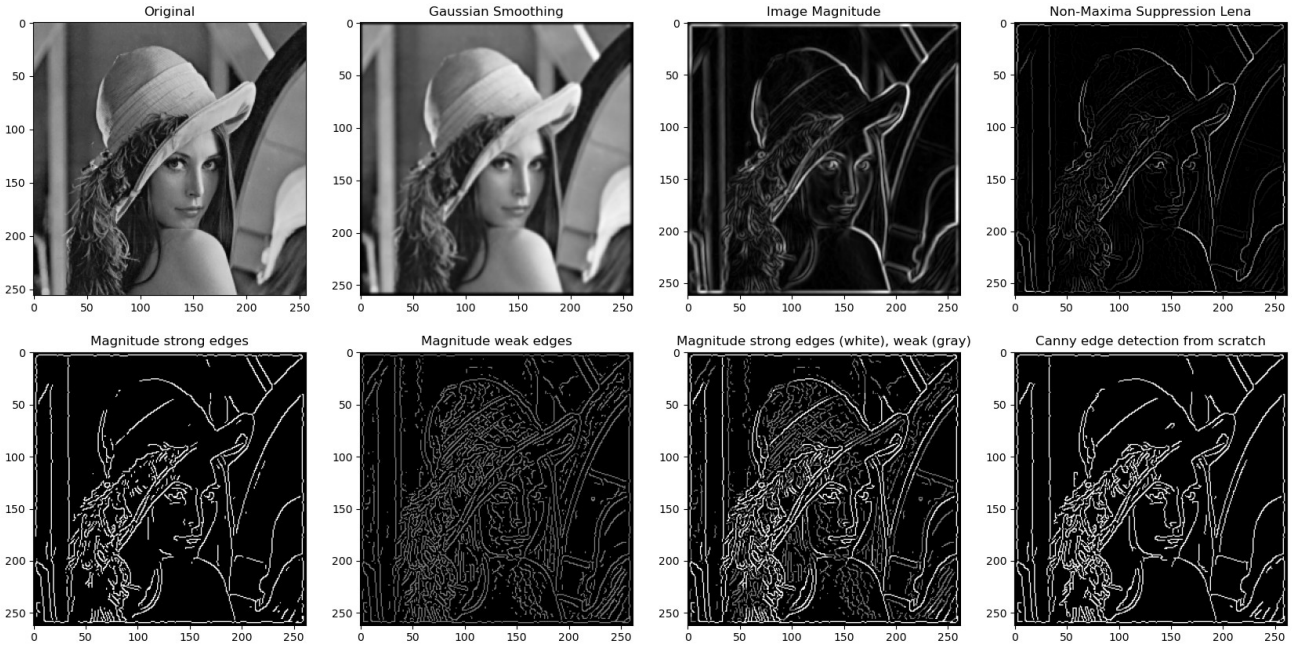

Figure 1: Image at each step in the canny edge detection algorithm.

Different high and low thresholds where experimented with. Decreasing the high threshold increases the amount of edges that are considered strong edges and thus drastically increases the edges and noise in the image. Decreasing the low threshold increases the amount of weak edges that can connect strong edges to generate clean lines. Thus a good choice is a high threshold that does not capture noise, but still captures parts of the main features of an image and then a low threshold that captures all the desired features in an image with some noise.

Figure 2 displays edge detectors that are in the OpenCV package. It is evident that Canny edge detection outperforms the other detection methods in accuracy and noise reduction. The canny edge detection that was implemented from scratch displayed in Figure 1 is comparable in accuracy to the builtin Canny edge detection algorithm in OpenCV.

Figure 2: OpenCV built in edge detection algorithms