Zele Zoltán

Image and Signal Processing

Homework assignment

Task: Edge detection

Detect connected edges on images! Implement one of the edge detection methods (Prewitt, Sobel, Laplace, Laplacian of Gaussian (LoG), Canny), with edge linking, and edge thinning, for grayscale images. The expected output is a black-and-white (binary) image with thick edges.

My choice was to implement the Canny edge detector.

Solution

Since the canny edge detector uses the first derivative to find potential edges, it is important that we first smooth the image as the derivative is very sensitive to noise and it can cause false detection.

This can be done like this for example:

$$\mathbf{B} = \frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} * \mathbf{A}.$$

where A is our original image, and B is the filtered image. The matrix represents a 5x5 Gaussian filter, which is often used to remove gaussian noise.

The next step is to calculate the gradient at each point. For this I used the Sobel operator. The operator uses two 3x3 kernels to approximate the vertical and horizontal derivatives.

$$\mathbf{G}_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G}_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \mathbf{A}$$

Now the gradient and the it’s direction can be calculated as follows:

$$\mathbf{G} = \sqrt{\mathbf{G}_x^{\,2} + \mathbf{G}_y^{\,2}}$$

$$\mathbf{\Theta} = \operatorname{atan2}(\mathbf{G}_y, \mathbf{G}_x)$$

The edge direction angle is rounded to one of four angles representing vertical, horizontal and two diagonals (0°, 45°, 90°, 135°).

The next step is to use the technique called non-maximum suppression. The technique is used to find the local maximum of an edge at a given direction. The resulting image will have thin edges as the result.

Next I used double tresholding to find real edges. There is a larger and a smaller treshold representing strong and weak edges. Strong edges are automatically marked as real edges and if we find a weak edge neighbouring a strong edge, then it will be marked as a real edge as well. To find continous edges, this step is iterated a few times.

Usage

The implementation can be found in the **edgedetection.m** matlab script file. To use the function you have to pass the image path to the function as an argument. For example: **edgedetection('lena.png')**.

If you run the script it will show after each step how the image is being processed, and finally the resulting edges detected.