Edge Detection Methods with Parallelization

Mingdao Che
mdche@bu.edu

Pat Rick Ntwari
pntwari@bu.edu

Fuyao Wang
fuyao@bu.edu

1.Introduction

It is generally understood that a successful edge detection of an image has three general criteria. The first is the detection of edges with low error rate, which means that the detection should accurately catch as many edges shown in the image as possible. The second is that the edge point detected from the operator should accurately localize on the center of the edge. Lastly, a given edge in the image should only be marked once, and where possible, image noise should not create false edges. These general criteria are satisfied by the calculus of variations – a technique which finds the function which optimizes a given functional and pioneered by the Australian computer scientist John F. Canny.

Canny edge detection is a technique to extract useful structural information from different vision objects and dramatically reduce the amount of data to be processed. It has been widely applied in various computer vision systems. This report outlines an application of this process, compares several variations in the technique's implementation and proposes a parallelization approach to speed up the process.

2. Edge Detection Process

2.1 Gaussian filter

Since edge detection results are easily affected by the noise in the image, it is essential to filter out the noise to prevent false positives. There are several methods of smoothing an image, including the mean, median and Gaussian filtering. The Canny technique uses the Gaussian, which outputs a weighted average of each pixel's neighborhood, and this *weight* is biased towards the central pixels. The filter kernel is convolved with the image. The equation general equation for a Gaussian filter in 2D form is as follows:

$$H_{i,j} = \frac{1}{2\pi\sigma^2} e^{\left(-\frac{x^2+y^2}{2\sigma^2}\right)}$$

where σ signifies the standard deviation, much like in a bell (Gaussian) curve. Σ controls the degree of blurring. In the computing context, a kernel of size (2k+1)×(2k+1) is computed using the following kernel:

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

where $i \geq 1$ and $(2k+1) \geq j$ and k is the distance (radius) from the central pixel.

This application uses two Gaussian approximations, one with σ=1 and another with σ=1.4 for comparison and evaluation reasons. Theoretically, the larger the standard deviation, the more blurred. In reality, the selection of σ is much less trivial. The kernel size for both of these standard deviation sizes is 5x5. The σ=1 approximation used is reproduced below:

$$\frac{1}{273}\begin{pmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{pmatrix},$$

where the values are normalized for the edges to be 1, and the sum of which is 273.

The selection of the size of the Gaussian kernel will affect the performance of the detector. The larger the size is, the lower the detector's sensitivity to noise. Additionally, the localization error to detect the edge will slightly increase with the increase of the Gaussian filter kernel size. A 5×5 is judged a good size for most cases.

2.2 Find intensity gradient

The Gaussian filter generates a grayscale image with each edge pointing in a variety of directions. Normally, the Canny algorithm utilizes different operators to detect horizontal, vertical and two diagonal edges in the blurred image. The edge detection operators (such as Roberts, Prewitt, or Sobel) return a value for the first derivative in the horizontal direction (G_x) and the vertical direction (G_y). From this the edge gradient and direction can be determined:

$$\mathbf{G} = \sqrt{\mathbf{G}_x^2 + \mathbf{G}_y^2}$$
$$\mathbf{\Theta} = \operatorname{atan2}(\mathbf{G}_y, \mathbf{G}_x),$$

where G can be computed using the hypot function and atan2 is the arctangent function with two arguments. The edge direction angle is rounded to one of four angles representing vertical, horizontal, and the two diagonals (0°, 45°, 90° and 135°). An edge direction falling in each angle region will be set to a specific angle value, for instance, θ in [0°, 22.5°] or [157.5°, 180°] maps to 0°.

2.3 Non-maximum suppression

Non-maximum suppression is an edge thinning technique. It is applied to find "the largest" edge. After applying the gradient calculation, the edge extracted from the gradient value is still quite blurred. With respect to criterion 3(A given edge in the image should only be marked once, and where possible, image noise should not create false edges), there should only be one accurate response to the edge. Thus non-maximum suppression can help to suppress all the gradient values (by setting them to 0) except the local maxima, which indicates locations with the sharpest change of intensity value. The algorithm for each pixel in the gradient image is:

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.

In some implementations, the algorithm categorizes the continuous gradient directions into a small set of discrete directions and then moves a 3x3 filter over the output of the previous step (that is, the edge strength and gradient directions). At every pixel, it suppresses the edge strength of the center pixel (by setting its value to 0) if its magnitude is not greater than the magnitude of the two neighbors in the gradient direction. For example,

- if the rounded gradient angle is 0° (i.e. the edge is in the north-south direction) the point will be considered to be on the edge if its gradient magnitude is greater than the magnitudes at pixels in the east and west directions,
- if the rounded gradient angle is 90° (i.e. the edge is in the east-west direction) the point will be considered to be on the edge if its gradient magnitude is greater than the magnitudes at pixels in the north and south directions,
- if the rounded gradient angle is 135° (i.e. the edge is in the northeast-southwest direction) the point will be considered to be on the edge if its gradient magnitude is greater than the magnitudes at pixels in the northwest and south-east directions,
- if the rounded gradient angle is 45° (i.e. the edge is in the northwest-southeast direction) the point will be considered to be on the edge if its gradient magnitude is greater than the magnitudes at pixels in the northeast and southwest directions.

In more accurate implementations, linear interpolation is used between the two neighboring pixels that straddle the gradient direction. For example, if the gradient angle is between 89° and 180°, interpolation between gradients at the north and northeast pixels will give one interpolated value, and interpolation between the south and southwest pixels will give the other (using the

conventions of the last paragraph). The gradient magnitude at the central pixel must be greater than both of these for it to be marked as an edge.

Note that the sign of the direction is irrelevant, i.e. north-south is the same as south-north and so on.

2.4 Double threshold and hysteresis

The non-maximum suppression stage of the Canny technique outputs an image with crisper edges, but the edges are still considered *probable* edges. There still needs to be a way to decide what pixels are in fact part of an edge and which are not. The double threshold is supplemented by hysteresis (or edge tracking) to offer a solution.

The step splits all the remaining pixels into the groups. Some pixels are established to be absolute edges, others are established to be absolutely not edges, and the rest lie in a limbo of *maybe*. To group these, there needs to be numerical criteria based on the gradient level of these pixels. An edge pixel must have a relatively high gradient, while a non-edge pixel has the contrary. The gradient criterion over which a pixel is considered to be on an edge, combined with the criterion under which a pixel is disregarded as not partaking in an edge, makes up the "double" threshold. Pixels whose gradients are between these two values are kept for now. They are re-evaluated at a later stage.

It is not immediately clear what thresholds to choose. The choice resides with the developer and depends on the usage case. This application developed two ways, whose final outputs are shown in Section 5.3, with all other parameters constant. The first is as follows:

$$Min = \frac{Max}{2} \qquad Max = Median$$

The second was as follows:

$$Min = 1Q \qquad Max = 3Q$$

Both of these ways depend on how pixel gradients compare to each other. The first should in theory be a less strict version because it uses the median gradient as the high threshold. The second uses the third quartile as the high threshold and thus will let few pixels be considered strong edges. Using a bigger high threshold presents some benefits in cleaning up edges, but presents problems, potentially making an object unrecognizable.

Edge tracking hysteresis is an essential supplement to the double threshold stage because it provides a way to distinguish between the many "maybe" pixels. For each weak-edge pixel, a strong-edge pixel must be present within a certain radius for the pixel to also be considered part of an edge. This radius is developer-designated. Much like the rest of the stages, it involves significant computational looping, and comparison with neighbouring pixels.

3. Comparisons of different operators

3.1 Sobel Operator

The Sobel operator is a discrete operator, which computes the gradient for intensity changes at each point in an image. The operator consists of a pair of 3*3 convolution kernels as shown.

$$S_x = \begin{matrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{matrix} \qquad S_y = \begin{matrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{matrix}$$

3.2 Roberts Operator

The Roberts operator performs a simple, quick to compute, two-dimensional (2-D) spatial gradient measurement on an image. The operator consists of a pair of 2*2 convolution kernels as shown.

$$R_x = \begin{matrix} 1 & 0 \\ 0 & -1 \end{matrix} \qquad R_y = \begin{matrix} 0 & 1 \\ -1 & 0 \end{matrix}$$

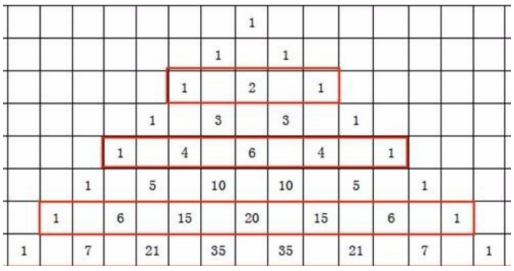
3.3 Prewitt Operator

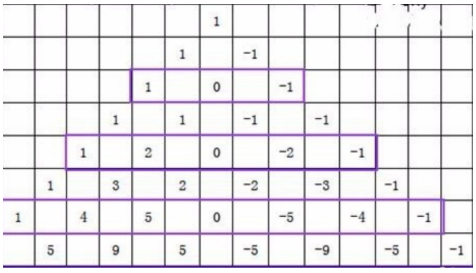
Prewitt operator is a discrete differentiation operator which functions similar to the Sobel operator, by computing the gradient for the image intensity function. The Prewitt edge detection operator is used for detecting vertical and horizontal edges of images.

$$P_x = \begin{matrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{matrix} \qquad P_y = \begin{matrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{matrix}$$

3.4 5X5 Sobel Operator

For further usage, we also generate a 5*5 sobel kernel to compare its performance with other kernels. This Kernel is generated by Pascal Triangle, which is defined as a triangular array of numbers in which those at the ends of the rows are 1 and each of the others is the sum of the nearest two numbers in the row above (the apex, 1, being at the top).





According to the Pascal Triangles, the 5*5 Sobel can be generated by the convolution of the third line of the two triangles.

$$S_x = \begin{matrix} -1 & -2 & 0 & 2 & 1 \\ -4 & -8 & 0 & 8 & 4 \\ -6 & -12 & 0 & 12 & 6 \\ -4 & -8 & 0 & 8 & 4 \\ -1 & -2 & 0 & 2 & 1 \end{matrix} \qquad S_y = \begin{matrix} 1 & 4 & 6 & 4 & 1 \\ 2 & 8 & 12 & 8 & 2 \\ 0 & 0 & 0 & 0 & 0 \\ -2 & -8 & -12 & -8 & -2 \\ -1 & -4 & -6 & -4 & -1 \end{matrix}$$

3.5 Scharr Operator

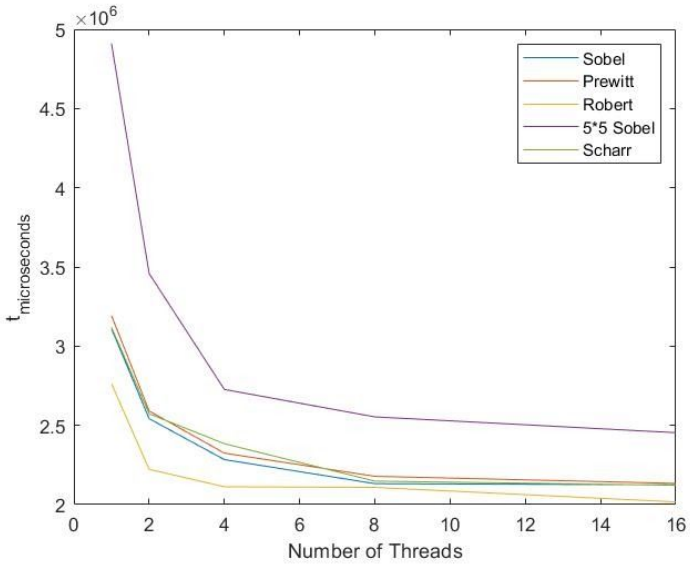
The Sobel–Feldman operator, while reducing artifacts associated with a pure central differences operator, does not have perfect rotational symmetry. Scharr looked into optimizing this property. Filter kernels up to size 5 x 5 have been presented there, but the most frequently used one is:

$$Scharr_x = \begin{matrix} -3 & 0 & 3 \\ -10 & 0 & 10 \\ -3 & 0 & 3 \end{matrix} \qquad Scharr_y = \begin{matrix} -3 & -10 & -3 \\ 0 & 0 & 0 \\ -3 & -10 & 3 \end{matrix}$$

4. Parallelization

Based on the hardware limitation, our team decided to use openMP as our parallelization library.

We tried 5 different numbers of threads to observe the performance on running time. The plot is shown as follows in the image.



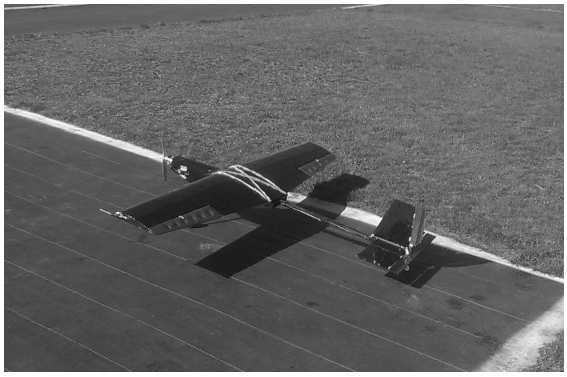
The curve of running time decreases rapidly from 1(serial) to 4, but slows down after 4 threads. In the testing process, there always existed outliers when the number of threads larger than 8. And we assumed the reason is that too many threads are requested by the codes causing the conflicts with the system limitation. The conflict finally slowed down the program. Based on the performance and the resource-saving, the number of threads is set as 4.

Kernel	Serial Time (microseconds)	4 Threads Time (microseconds)	% Change
Sobel	3,110,026	2,283,923	-27%
Prewitt	3,192,802	2,324,641	-27%
Robert	2,762,271	2,112,322	-24%
5x5 Sobel	4,910,604	2,726,721	-44%
Scharr	3,119,187	2,384,453	-24%

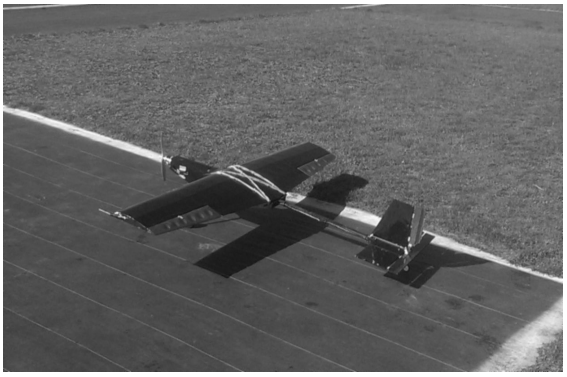
5. Results

5.1 Results of different processes

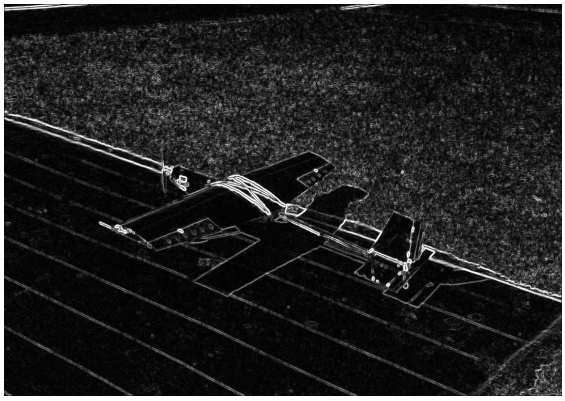
Original (Grayscale)



Blurred

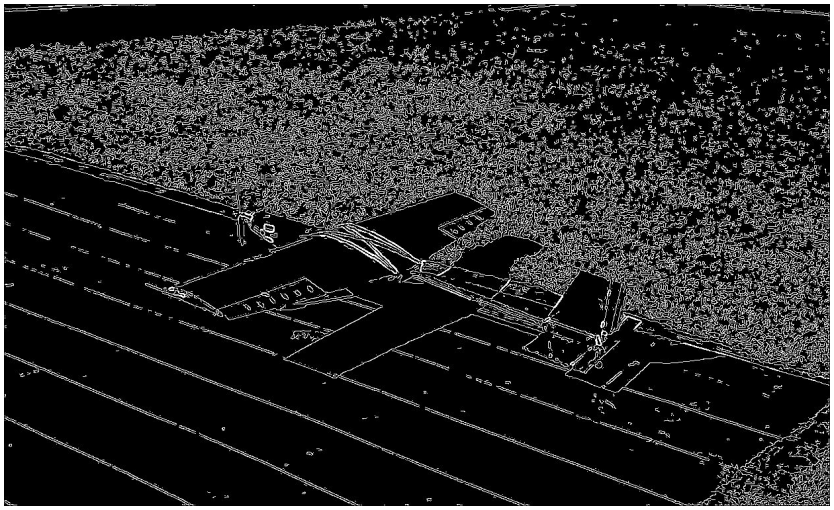


Gradient



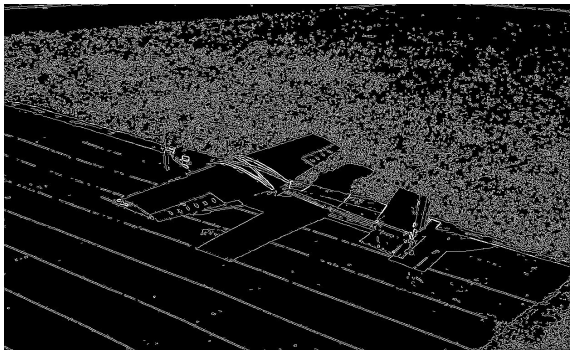
Suppression

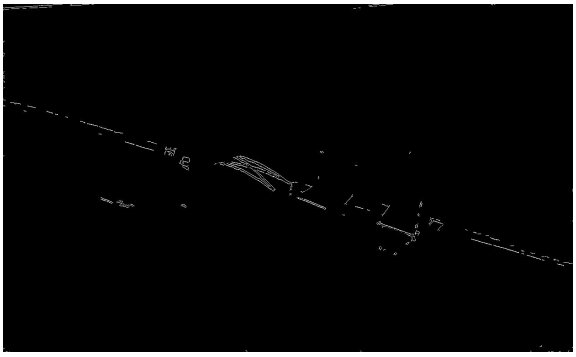
Final



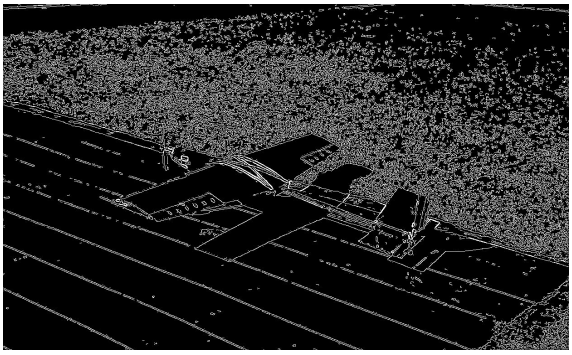
5.2 Comparisons of final edge by kernel (Sobel is standard, all other parameters constant)

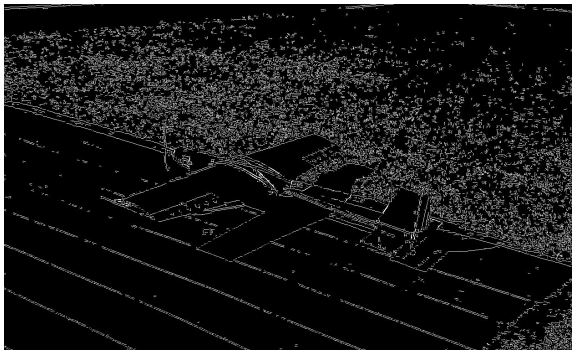
5.2.1 Sobel and Roberts



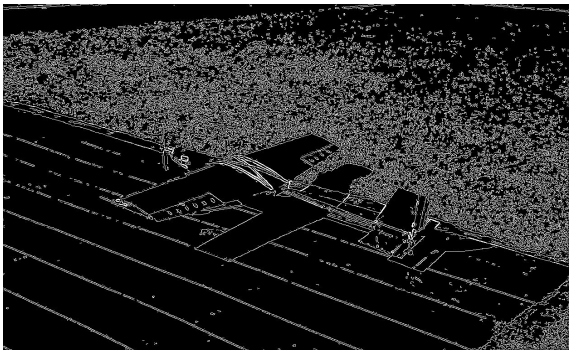


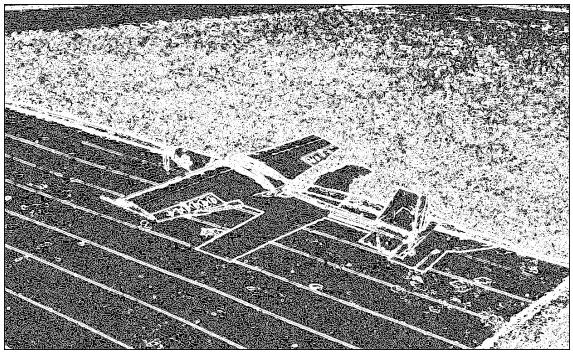
5.2.2 Sobel and Prewitt





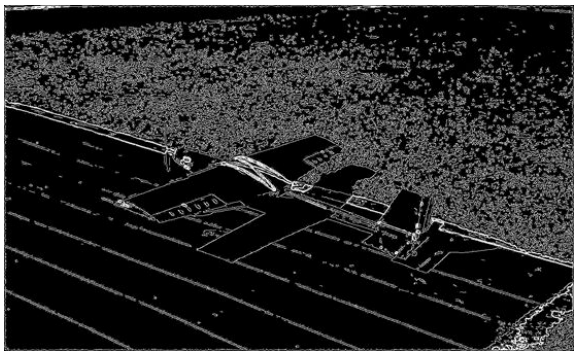
5.2.3 Sobel and 5x5 Sobel





5.3 Comparisons of final edge by based on thresholding philosophy

Thresholding based on median for high is much less strict than the third quartile as can be seen in the images below. 3Q thresholding gives a clearer image.



6. Conclusion

The Canny algorithm is efficient in determining the boundaries of objects in a digital image. The algorithm implementation is however far from being straightforward. In establishing a root structure for edge detection, the technique leaves many of its branches malleable depending on the situation; indeed much improved. Often, various pictures and various goals will require various implementations, kernels or thresholds. The age old dichotomy of speed and accuracy. Is this openness flexibility or ambiguity?

References:

[1] Canny edge detector: https://en.wikipedia.org/wiki/Canny_edge_detector
[2] Mehravar Rafati, Masoud Arabfard and Mehrdad Rafati-Rahimzadeh, “Comparison of Different Edge Detections and Noise Reduction on Ultrasound Images of Carotid and Brachial Arteries Using a Speckle Reducing Anisotropic Diffusion Filter”. In: Iran Red Crescent Med J. 2014 Sep.