Implementation of Canny Edge detection Algorithm

Rajasekhar Madireddy

***Abstract* - *Edge detection refers to the process of identifying and locating sharp discontinuities in an image. The discontinuities are sudden changes in pixel intensity which characterize boundaries of objects in a scene.***

***Keywords— identifying and locating sharp discontinuities, pixel intensity, boundaries of objects***

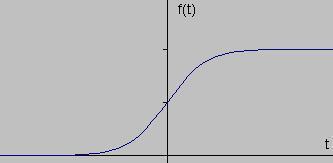
1.Introduction

The Variables involved in the selection of an edge detection operator include edge orientation and noise environment

There are many ways to perform edge detection. However, majority of the different methods can be grouped into two categories:

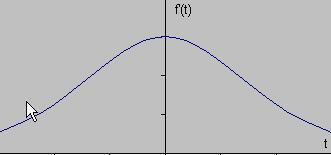
Gradient: The gradient method detects the edges by looking for the maximum and minimum in the first derivative of the image.

Laplacian: The Laplacian method searches for zero crossings in the second derivative of the image to find edges. An edge has the one-dimensional shape of a ramp and calculating the derivative of the image can highlight its location.



*Fig1:* *Edge shown by the jump in intensity of the signal*

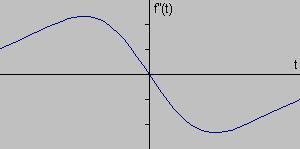
If we take the gradient of this signal (which, in one dimension, is just the first derivative with respect to t) we get the following:



*Fig 2 : Gradient of this signal*

Clearly, the derivative shows a maximum located at the center of the edge in the original signal. This method of locating an edge is characteristic of the “gradient filter” family of edge detection

filters and includes the Sobel method. A pixel location is declared an edge location if the value of the gradient exceeds some threshold. As mentioned before, edges will have higher pixel intensity values than those surrounding it. So once a threshold is set, you can compare the gradient value to the threshold value and detect an edge whenever the threshold is exceeded. Furthermore, when the first derivative is at a maximum, the second derivative is zero. As a result, another alternative to finding the location of an edge is to locate the zeros in the second derivative. This method is known as the Laplacian and the second derivative of the signal is shown below:



*Fig 3: Second derivative of the signal*

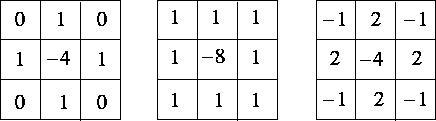
2. Laplacian of Gaussian

The Laplacian is a 2-D isotropic measure of the 2nd spatial derivative of an image. The Laplacian of an image highlights regions of rapid intensity change and is therefore often used for edge detection. The Laplacian is often applied to an image that has first been smoothed with something approximating a Gaussian Smoothing filter to reduce its sensitivity to noise. The operator normally takes a single gray-level image as input and produces another gray-level image as output.

The Laplacian L(x, y) of an image with pixel intensity values I(x,y) is given by:

page5image18672

Since the input image is represented as a set of discrete pixels, we should find a discrete convolution kernel that can approximate the second derivatives in the definition of the Laplacian.



*Figure 4: Three commonly used discrete approximations to the Laplacian filter.*

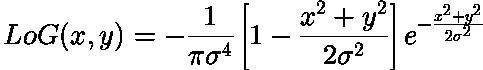
Because these kernels are approximating a second derivative measurement on the image, they are very sensitive to noise. To counter this, the image is often Gaussian Smoothed before applying the Laplacian filter. This pre-processing step reduces the high frequency noise components prior to the differentiation step.

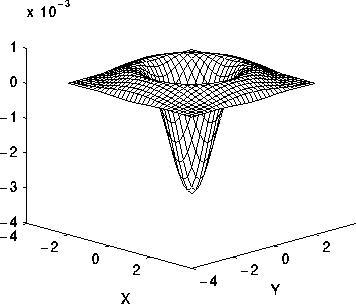
In fact, since the convolution operation is associative, we can convolve the Gaussian smoothing filter with the Laplacian filter first, and then convolve this hybrid filter with the image to achieve the required result. Doing things this way has two advantages:

• Since both the Gaussian and the Laplacian kernels are usually much smaller than the image, this method usually requires far fewer arithmetic operations.

The LoG (`Laplacian of Gaussian') kernel can be pre-calculated in advance so only one convolution needs to be performed at run-time on the image.

The 2-D LoG function centered on zero and with Gaussian standard deviation σ has the form:





*Fig.5 Gaussian Smoothing*

3. Steps in Edge Detection

Algorithms for edge detection contain four steps:

3.1 Filtering: Since gradient computation based on intensity values of only two points are susceptible to noise and other vagaries in discrete computations, filtering is commonly used to improve the performance of an edge detector with respect to noise. However, there is a trade- off between edge strength and noise reduction. More filtering to reduce noise results in a loss of edge strength.

3.2 Enhancement: In order to facilitate the detection of edges, it is essential to determine changes in intensity in the neighborhood of a point. Enhancement emphasizes pixels where there is a significant change in local intensity values and is usually performed by computing the gradient magnitude.

3.3 Detection: We only want points with strong edge content. However, many points in an image have a nonzero value for the gradient, and not all of these points are edges for a particular application. Therefore, some method should be used to determine which points are edge points. Frequently, thresholding provides the criterion used for detection

3.4 Localization: The location of the edge can be estimated with subpixel resolution if required for the application. The edge orientation can also be estimated. It is important to note that detection merely indicates that an edge is present near a pixel in an image but does not necessarily provide an accurate estimate of edge location or orientation. The errors in edge detection are errors of misclassification: false edges and missing edges. The errors in edge estimation are modeled by probability distributions for the location and orientation estimates. We distinguish between edge detection and estimation because these steps are performed by different calculations and have different error models.

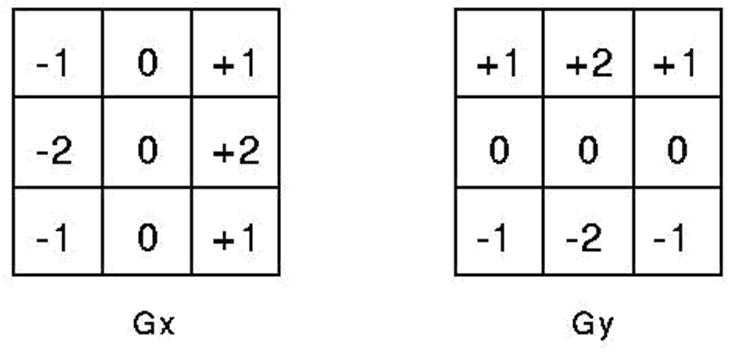
4. Implementation of Canny’s Edge Detection Algorithm

The canny edge detector first smoothest the image to eliminate and noise. It then finds the image gradient to highlight regions with high spatial derivatives. The algorithm then tracks along these regions and suppresses any pixel that is not at the maximum (non-maximum suppression). The gradient array is now further reduced by hysteresis. Hysteresis is used to track along the remaining pixels that have not been suppressed. Hysteresis uses two thresholds and if the magnitude is below the first threshold, it is set to zero (made a non-edge). If the magnitude is above the high threshold, it is made an edge. And if the magnitude is between the 2 thresholds, then it is set to zero unless there is a path from this pixel to a pixel with a gradient above T2.

Step 1

After smoothing the image and eliminating the noise, the next step is to find the edge strength by taking the gradient of the image. The Sobel operator performs a 2-D spatial gradient measurement on an image. Then, the approximate absolute gradient magnitude (edge strength) at each point can be found. The Sobel operator uses a pair of 3x3 convolution masks, one estimating the gradient in the x-direction (columns) and the other estimating the gradient in the y-direction (rows).

page7image33296

page8image1256

The magnitude, or edge strength, of the gradient is then approximated using the formula: |G| = |Gx| + |Gy|

Step 2

The direction of the edge is computed using the gradient in the x and y directions. However, an error will be generated when sum X is equal to zero. So, in the code there has to be a restriction set whenever this takes place. Whenever the gradient in the x direction is equal to zero, the edge direction has to be equal to 90 degrees or 0 degrees, depending on what the value of the gradient in the y-direction is equal to. If GY has a value of zero, the edge direction will equal 0 degrees. Otherwise the edge direction will equal 90 degrees. The formula for finding the edge direction is just

α = Tan-1(Gy / Gx)

Step 3

Once the edge direction is known, the next step is to relate the edge direction to a direction that can be traced in an image. So, if the pixels of a 5x5 image are aligned as follows:

x x x x x

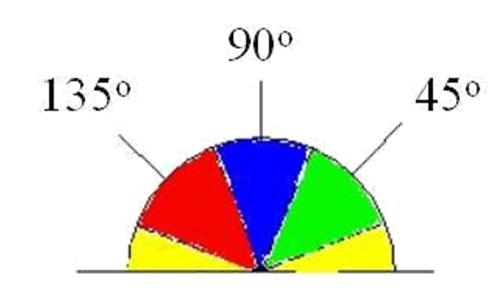
x x x x x

x x **a** x x

x x x x x

x x x x x

Then, it can be seen by looking at pixel "**a**", there are only four possible directions when describing the surrounding pixels - 0 degrees (in the horizontal direction), 45 degrees (along the positive diagonal), 90 degrees (in the vertical direction), or 135 degrees (along the negative diagonal). So now the edge orientation has to be resolved into one of these four directions depending on which direction it is closest to (e.g. if the orientation angle is found to be 3 degrees, make it zero degrees).



*Fig 6: semicircle dividing it into 5 regions.*

Therefore, any edge direction falling within the yellow range (0 to 22.5 & 157.5 to 180 degrees) is set to 0 degrees. Any edge direction falling in the green range (22.5 to 67.5 degrees) is set to 45 degrees. Any edge direction falling in the blue range (67.5 to 112.5 degrees) is set to 90 degrees. And finally, any edge direction falling within the red range (112.5 to 157.5 degrees) is set to 135 degrees.

Step 4

After the edge directions are known, non-maximum suppression now should be applied. Non- maximum suppression is used to trace along the edge in the edge direction and suppress any pixel value (sets it equal to 0) that is not considered to be an edge. This will give a thin line in the output image.

Step 5

Finally, hysteresis is used as a means of eliminating streaking. Streaking is the breaking up of an edge contour caused by the operator output fluctuating above and below the threshold. If a single threshold, T1 is applied to an image, and an edge has an average strength equal to T1, then due to noise, there will be instances where the edge dips below the threshold. Equally it will also extend above the threshold making an edge look like a dashed line. To avoid this, hysteresis uses 2 thresholds, a high and a low. Any pixel in the image that has a value greater than T1 is presumed to be an edge pixel, and is marked as such immediately. Then, any pixels that are connected to this edge pixel and that have a value greater than T2 are also selected as edge pixels. If you think of following an edge, you need a gradient of T2 to start but you don't stop till you hit a gradient below T1.

5. References

[1] R. C. Gonzalez, R. E. Woods, Digital Image Processing, Third Edition, Pearson Prentice Hall,2008.

[2] Digital Image Processing: PIKS Inside, Third Edition. William K. Pratt

[3] Bernd Jähne 123 5th revised and extended edition

**.**