Two Practical Issues in Canny’s Edge Detector Implementation

Gang Liu and Robert M. Haralick
Department of Electrical Engineering
University of Washington
Seattle, WA 98195-2500, U.S.A.
{gliu,haralick}@isl.ee.washington.edu

Abstract

We address two practical issues, namely smoothing factor selection and efficient implementation of thresholding with hysteresis, in implementing Canny’s edge detector. The smoothing factor of the Gaussian kernel should be chosen to maximize the discrete version of Canny’s original criteria. Thresholding with hysteresis should be implemented using an efficient connected component analysis algorithm. Following these suggestions in implementing Canny’s edge detector will in general result in optimal edge detection quality and very significant reduction in running time for large images.

1. Introduction

Canny’s edge detector is a very popular edge operator. It is always among the best performers in various edge operation evaluation experiments [1, 5], and has become part of the standard against which the performance of a newly developed edge detector is compared. It is especially popular among casual users of the edge detection module which only play a small and not-so-vital role in a larger vision algorithm or system. This partly is due to the ease in its implementation and the availability of several “standard” implementations of it. Although many researchers still criticize some aspects of this detector, it is in fact the most widely used operator.

All the implementations of Canny’s edge detector which we have seen so far share the same two weaknesses. The first is the arbitrary (maybe careless too?) choice of the smoothing factor. The second is the inefficient implementation of the thresholding with hysteresis using a recursive procedure.

The optimal choice for the smoothing factor should be the one that maximizes Canny’s combined criteria function. This gives the best combined detection and localization performance. The optimal smoothing factors for a large range of neighborhood sizes are calculated and tabulated in this paper. A linear regression model accounts very well for the relationship between the optimal smoothing factor and the neighborhood size.

In computer programming, the recursive function call is known to have low execution efficiency due to its high requirement for local temporary data storage. However, logically it is a natural way for examining spatial connectivity. Probably that is the reason why all the implementation of the thresholding with hysteresis that we have seen are implemented using the recursive function call. However, this is not the best way to do it. More efficient algorithms exist for examining spatial connectivity. We present a method that makes simple use of a connected component analysis algorithm to produce exactly the same result as produced by the recursive function call based algorithm.

The two suggestions which we make in this paper do not contribute to the theory part of edge detection. However, incorporating them in edge detector implementations will often result in very noticeable improvement both in edge detection quality and in running speed.

2. Smoothing factor selection

Canny [3] uses what has become known as the dG operator for gradient estimation. This operator is the tensor product of a smoothing Gaussian along one direction with the derivative of that Gaussian along the perpendicular direction. The row-derivative kernel is expressed as

$$f_r(r, c) = \frac{r}{2\pi \sigma^4} e^{-\frac{r^2 + c^2}{2\sigma^2}} \quad (1)$$

where $\sigma > 0$ is the standard deviation of the smoothing Gaussian. The bigger $\sigma$ is, the greater the smoothing effect is. It is called the smoothing factor in this paper.

In the 1-D formulation, the derivative of Gaussian enjoys many good properties. It approximates very closely to the numerically obtained optimal digital filter that maximizes Canny’s edge criterion function. It is the exact solution to
the alternative edge location criterion function proposed by Tagare et al [6, 2, 7]. Most studies of the behavior of this filter are done in the continuous case, where the smoothing factor $s$ is assumed to be a chosen constant. There has not been much in-depth study of the effect of discretization on this filter.

In practical applications of edge detection, people tend to use small neighborhood sizes, such as $3 \times 3$ or $5 \times 5$. The effect of discretization is very significant. The choice of the smoothing factor $s$ of the smoothing Gaussian needs to be carefully made in order to get the optimal edge detection output.

In trying to solve the variational optimization problem for the optimal filter, Canny specified some boundary conditions, including one requiring the value of the filter impulse response at the end of the support to be zero. This is reflected in the numerically obtained optimal solution. The smoothing factor $s$ of the approximating derivative of Gaussian seems to be such that most of the mass of the Gaussian is contained by the filter. In practice, standard implementations of the Canny’s edge detector make their local neighborhood cover four to six times the standard deviation of the center part of the Gaussian. (In the continuous case, six times the standard deviation at the center part covers over 99.7% of the mass.)

We propose to use the value for $s$ which maximizes the discrete version of the criterion function which Canny originally proposed. For the $0^{\circ}$ step edge in the $(2R+1) \times (2R+1)$ neighborhood, the square of that criterion function in the discrete 2D case is

$$\left( \sum_{r=R}^{R} \sum_{c=R}^{R} ce^{-\frac{r^2+c^2}{2\sigma^2}} \right)^2 \sum_{r=-R}^{R} \sum_{c=-R}^{R} c^2e^{-\frac{r^2+c^2}{2\sigma^2}} \times s^4 \left( \sum_{r=-R}^{R} \sum_{c=-R}^{R} \left( (r^2 - s^2)^2 + r^2c^2 \right) e^{-\frac{r^2+c^2}{2\sigma^2}} \right)^2$$

Given $R$, the optimal value $s$ which maximizes the above criteria function can be computed exactly.

We compute optimal value $s$ for a range of step edge orientations and neighborhood sizes. The average over the step edge orientation is listed in Table 1 for each neighborhood size. The relationship between the average optimal $s$ and the neighborhood size $N = 2R + 1$ can be closely approximated by a linear model

$$s = 0.1092N + 0.4335 \quad (2)$$

This regression model should be used to find the smoothing factor value when applying the Canny’s edge detector.

### Table 1. Average smoothing factor $s$ maximizing Canny’s criterion function for the dG operator. Regression model $s = 0.1092N + 0.4335$

<table>
<thead>
<tr>
<th>size</th>
<th>5</th>
<th>7</th>
<th>9</th>
<th>11</th>
<th>13</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s$</td>
<td>0.95</td>
<td>1.1</td>
<td>1.4</td>
<td>1.6</td>
<td>1.8</td>
<td>2.1</td>
</tr>
<tr>
<td>size</td>
<td>17</td>
<td>19</td>
<td>21</td>
<td>23</td>
<td>25</td>
<td>27</td>
</tr>
<tr>
<td>$s$</td>
<td>2.3</td>
<td>2.5</td>
<td>2.8</td>
<td>3.0</td>
<td>3.2</td>
<td>3.4</td>
</tr>
<tr>
<td>size</td>
<td>29</td>
<td>31</td>
<td>33</td>
<td>35</td>
<td>37</td>
<td>39</td>
</tr>
<tr>
<td>$s$</td>
<td>3.6</td>
<td>3.8</td>
<td>4.1</td>
<td>4.3</td>
<td>4.5</td>
<td>4.70</td>
</tr>
<tr>
<td>size</td>
<td>41</td>
<td>43</td>
<td>45</td>
<td>47</td>
<td>49</td>
<td>51</td>
</tr>
<tr>
<td>$s$</td>
<td>4.91</td>
<td>5.12</td>
<td>5.33</td>
<td>5.54</td>
<td>5.75</td>
<td>5.96</td>
</tr>
</tbody>
</table>

The computed optimal smoothing factor $s$ and the commonly used smoothing factor $s = 0.2N - 0.2$ (or $N = 5s + 1$) are plotted in Figure 1. For the most commonly used neighborhood sizes, such as $5 \times 5$ and $7 \times 7$, the commonly used smoothing factors is not very much different from the optimal choice. Therefore we might not see much difference in edge detection quality. However, for applications where a large neighborhood size is needed, the difference between the casually chosen value and the optimal value is very large, and the difference in detection quality will be significant.

### 3. Implementation of thresholding with hysteresis

Thresholding with hysteresis [3] is a widely accepted technique for picking up features that are not very strong by themselves but whose significance get boosted by its spatial proximity to strong features. It uses two thresholds called the high and low thresholds. If a feature value is higher than the high threshold, it is declared as a positive. If it is lower than the low threshold, it is declared as a negative. If its value is in between the two thresholds, its classification depends on whether it is spatially close (connected) to some other positive features. It is declared as a negative if it does not have any neighbor that is a positive feature. Otherwise, it is declared as positive. Although originally proposed for the edge detection application, this technique is readily applicable to many other feature detection situations.

In the standard implementations, a recursive procedure is used to handle the spatial connectivity. The procedure starts from pixel locations with values greater than the high threshold. This is declared as a positive pixel. It then looks among the eight neighbors for values are not lower than the low threshold. If there exists such a pixel, the procedure goes down a level of recursion by calling itself again for this
new pixel location. The procedure backs up one level if no new edge pixel is found in the eight neighbors. The result of this procedure is that, all pixels above the high threshold are declared as positive. All pixels between the two thresholds that are connected either directly or through a series of pixels between the two thresholds are declared as positive, too. All other pixels are declared as negative.

As with any recursive procedure, this process needs to use much stack space to store the intermediate data in order to be able to retrace. For large images, this may require a large amount of memory. For small memory systems, it puts unnecessarily tight restrictions on the image size that can be processed. For virtual memory systems, this may incur excessive virtual memory swapping and greatly prolong the running time.

This problem can be avoided if we use the following connected component analysis [4] based procedure. A connected component analysis procedure operates on a binary image of foreground and background pixels. A connected component consists of foreground pixels which connected to each other either directly or through a connected series of foreground pixels. In the edge linking application, 8-connectivity is used, i.e., a pixel is considered directly connected to each of its eight immediate neighbors. There is an efficient algorithm [4] which makes only two passes over the image without using any recursive procedure.

The connected component analysis based thresholding procedure is very simple. Let the pixels below the low threshold be called Class I binary 1 pixels, those above the high thresholds be called Class II binary 1 pixels, and those between the two thresholds be called Class III binary 1 pixels.

- Obtain two binary edge maps by applying the simple threshold with the high and low thresholds respectively. Call the resulting edge maps $B_h$ and $B_l$.
- Apply connected component analysis on the binary 1 pixels of $B_l$. Call the result $S_l$.
- Logically AND $B_h$ with $S_l$ and determine all connected components in $S_l$ that do not contain any foreground pixel in $B_h$, and turn those components into background.

The remaining components contains all the pixels passing the hysteresis thresholding procedure, i.e., the Class III pixels connected to Class II pixels directly or via a series of Class III pixels.

The efficiency of the proposed procedure is observed in one of our applications. An implementation of Canny’s edge detector needed to be run on some large images of more than 7 million pixels. By using the above procedure to replace the recursive procedure, the execution time reduces from around 20 hours to less than 20 minutes on a Sun SPARCstation 5 with a 161 MHz Turbo SPARC CPU running Sun Solaris 2.5.1 with 64 MB memory.

References