Real Time Edge Detection Implementation

Massood Zandi Atashbar *, Ikhlas Abdel-Qader
and Alireza Bab-Hadiashar**
*Department of Electrical and Computer Engineering
Western Michigan University, Kalamazoo, MI 49008, USA
**School of Engineering and Science, Swinburne University of Technology,
Hawthorn, Victoria, 3122, Australia

Abstract:
Edges are important features used in image processing in a wide range of applications. This paper will focus on segmentation operations for edge detection purposes. Java programming language has been employed and the results have been compared with the other available platforms. The results have demonstrated an easier and more efficient way compared to that of the conventional programming languages. Moreover, it is evident from our preliminary results that the performance of presented algorithms for image detection does not degrade when they are implemented in Java.

1. Introduction

Edges are important features widely used in image processing, pattern recognition, and computer vision, and they can be detected by the maxima of gradient or the zero-crossings of the second derivatives including the Laplacian [1]. Many computer vision algorithms rely in one way or the other on the detection of variety of features in the image. These features are most likely to be characterized by a unique combination of edges. In general, to detect edges in noisy images, a smoothing operation should be applied in order to reduce the noise. Because of the simplicity of linear filters, many methods of edge detection based on linear filters have been proposed such as Robert gradient, Sobel operators, Prewitt operators, Facet model, Laplacian operator, Box difference technique, and the derivatives or Laplacian of Gaussians. Java possesses two important features, which they are: online nature and efficient performance. Thus the java programming language has significant potential for the incorporation of various Image Processing applications in a faster and efficient manner, and thus enhancing real time image processing.

2. Background

In the analysis of the objects in images it is essential that we can distinguish between the objects of interest and "the rest". This latter group is also referred to as the background. The techniques that are used to find the objects of interest are usually referred to as segmentation techniques. To segment an image into the object of interest one has to identify boundaries and thus the process of edge detection. In this section we will discuss edge finding and we will present techniques for improving the quality of the segmentation result. It is important to understand that there is no universally applicable
segmentation technique that will work for all images, and no segmentation technique is perfect.

3. Edge Detection

The process of finding segmentation that yields all the pixels that, in principle, belong to the object or objects of interest in an image. An alternative to this is to find those pixels that belong to the borders of the objects. The most common Edge Detection techniques are [2-4]:

Gradient-based procedure - The central challenge to edge finding techniques is to find procedures that produce closed contours around the objects of interest [5]. For objects of particularly high SNR, calculating the gradient and then using a suitable threshold can achieve this.

While this technique works well for the noise free image, it fails to provide an accurate determination of those pixels associated with the object edges of a noisy image. A variety of smoothing techniques can be used to reduce the noise effects before the gradient operator is applied [6].

Zero crossing based procedure - A more modern view to handling the problem of edges in noisy images is to use the zero crossings generated by the Laplacian of an image. The edge location is, according to the model, at that place in the image where the Laplacian changes sign, the zero crossing. As the Laplacian operation involves a second derivative, this means a potential enhancement of noise in the image at high spatial frequencies. To prevent enhanced noise from dominating the search for zero crossings, a smoothing is necessary. The derivative operation is linear and shift-invariant. This means that the order of the operators can be exchanged or combined into one single filter. This approach leads to the Marr-Hildreth formulation [3] of the "Laplacian-of-Gaussians" (LoG) filter [7,8].

PLUS-based procedure - Among the zero crossing procedures for edge detection, perhaps the most accurate is the PLUS filter as developed by Verbeek and Van Vliet [9,10]. Suffice it to say that, for positively curved edges in gray value images, the Laplacian-based zero crossing procedure overestimates the position of the edge and the SDGD-based procedure underestimates the position. This is true in both two-dimensional and three-dimensional images with an error on the order of $(\sigma/R)^2$ where R is the radius of curvature of the edge. The PLUS operator has an error on the order of $(\sigma/R)^4$ if the image is sampled at, at least, three times the usual Nyquist sampling frequency or if we choose $\sigma$ to be greater than or equal to 2.7 and sample at the usual Nyquist frequency.

4. Java Implementation

As binary images frequently result from segmentation processes on gray level images, the morphological processing of the binary result permits the improvement of the segmentation result demonstrate an easier and more efficient way compared to that of the conventional programming languages. Moreover, it is evident from our preliminary results that the performance of presented algorithms does not degrade when they are implemented in Java. Indeed, we have shown that the Java implementation can, in some
cases, even outperform traditional approaches. In implementation several issues have to be taken into consideration such as:

- Physical properties of an image - By saying physical properties, we mean that only those properties of an image are changed which do not need any processing to be done on the pixels. Instead the pixels are remapped to the new coordinates. These operations include zooming, rotation by any given angle, creating mirror images and creating image duplicates. In all of these operations, the pixels are never processed; they are only mapped to the new coordinates determined by the parameters the user enters. The most common parameters accepted from the user are angle of rotation, magnification factor for zooming the image or shrinking the image size etc.

- Morphological operations - The most extensive class of binary image processing operations is sometimes collectively described as morphological operations. These include erosion and dilation and combinations of these operations. Erosion removes pixels from an image or, equivalently, turns pixels that were originally ON to OFF. A classic case would be removal of pixels that are touching another pixel that is part of the background. This removes a layer of pixels from around the periphery of all features and regions, which will cause some shrinking of dimensions. Instead of removing pixels from features, a complimentary operation called dilation can be used to add pixels. The dilation rule, analogous to that for erosion, is to add any background pixel that touches another pixel already part of the region. This will add a layer of pixels around the periphery of all features and regions, which will cause some increase in dimensions and may cause features to merge. It also fills small holes within features.

- The combination of erosion followed by dilation is called an opening, referring to the ability of this combination to open up spaces between just-touching features. It is one of the most commonly used sequences for removing pixel noise from binary images. If the sequence is performed in the other order, that is, a dilation operation followed by erosion, the result is not the same. Instead of removing isolated pixels that are ON, the result is to fill in places where isolated pixels are OFF, missing pixels within the features or narrow gaps between portions of a feature. Because of this ability to close up breaks in features, this combination is called a closing. Opening and closing operators have also been used alongside robust statistics for motion segmentation of image data [11].

These issues when taken into consideration by the programmer can improve and enhance the algorithm results for the real time image processing.

5. Experimental Results
In this section experimental results are presented. The following example illustrates edge detection based on Sobel gradient method combined with Isodata thresholding algorithm. While the technique works well for the noise free image in Figure 1(a), it fails to provide an accurate determination of those pixels associated with the object edges for the noisy image in Figure 1(b). Smoothing techniques can be used to reduce the noise effects before the gradient operator is applied.
Figure 1: Edge finding based on the Sobel gradient combined with the Isodata thresholding algorithm.

Analysis: The designed applet's performance is compared with the performance of the image-processing library designed in C++ and provided by the Intel Corporation. The following are the tabulated results for the performance of these two applications. Each of these implementations is carried out on a personal computer with Intel Pentium II processor with a speed of 450MHz and Windows 98 operating system. The experiment is carried out for two different sizes of a bitmap image.

Table 1. Implementation results for Intel's library programmed in C++ compared to the Java Application.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Time Taken (in seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>256 x 256 pixels</td>
</tr>
<tr>
<td>Intel</td>
<td>Java</td>
</tr>
<tr>
<td>Threshold</td>
<td>0.071</td>
</tr>
<tr>
<td>Edge detection*</td>
<td>-</td>
</tr>
</tbody>
</table>

* Not available with the Intel library

As seen in the above tables, the C++ library performs better for the operations involving physical attributes of the image such as size, rotation, mirror images etc. When considering other operations involving actual processing of images using filters and look up tables, the Java applet shows slightly superior performance. Apart from this the ability of the applet to perform even on the network gives it an edge over the C++ library, which cannot do the same. Unlike the Intel library, which accepts only bitmap files, the application developed in this project accepts images of a wide variety of formats like JPEG, GIF etc.

Implementation in Java.

Java proves to be the ideal choice for most of the image processing operations because, pattern recognition and image processing techniques are extensively used in a variety of fields like remote sensing based on data acquired from satellites, online medical diagnosis, transfer of pictures via the world wide web and many other applications which involve fast transmission and manipulation of the input image over the network. Java with its ability to provide both standalone and portable applications (called applets) has great potential and better prospects in the field of image processing when compared to other programming languages like C and C++. Another important
feature of Java is that the Java2 API introduced a straightforward image-processing model to help developers manipulate the image pixels and perform the basic image manipulation functions in a standard fashion. This model is based on the AWT package of Java 2 [12] and makes programming image processing algorithms more efficient giving the desired output.

6. Conclusions

The preliminary results have shown that Java simply demonstrate an easier and efficient way compared to that of the conventional programming languages. Indeed, we have shown that the Java implementation can in some cases, even outperform traditional approaches. Implementation of other image processing algorithms is underway and will be presented.

In addition to the imaging classes, Java supplies several others tools that provide enhanced control over the imaging process and support advanced imaging operations. If sophisticated graphical output is of a special interest, then Java has a wide range of utilities available, which would enhance the quality and accuracy of image manipulation and make implementations of these kinds of algorithms much more efficient.

References:
(http://www.java.sun.com/products/jdk/1.2/docs/api/index.html)