International Journal of Pure and Applied Mathematics
Volume 67 No. 3 2011, 341-354

MEDICAL IMAGE EDGE DETECTORS

John Schmeelk
Virginia Commonwealth University Qatar
P.O. Box 8095, Doha, QATAR
e-mail: jschmeelk@qatar.vcu.edu

Abstract: Image edge detection is an integral component of image processing to enhance the clarity of edges and the type of edges. Issues regarding edge techniques were introduced in my 2008 paper on transforms, filters, and edge detectors, see [15]. The current paper provides a deeper analysis regarding image edge detection using matrices; partial derivatives; convolutions; and the software, *MATLAB 7.9.0*, and *MATLAB Image Processing Toolbox 6.4*. Edge detection has applications in all areas of research, including medical research, see [6], [13]. For example, a patient can be diagnosed as having prostate cancer by studying the edges of the cells (see Figure 1). One can study a magnetic resonance (MR) brain image to indicate the edge functional, as illustrated in Russ, see [13] and Figure 2. Additionally, a patient can be diagnosed with an aneurysm by studying an angiogram (see Figure 3). The physician can study the angiogram, an image of the view of the problematic blood vessels, and determine the diameter of the increased size. The previous paper (see [15]) studied letters using vertical, horizontal, and Sobel transforms. This paper will study images to include the letter *O* and two images, those of *Cameraman* and *Rice,* included in the library of the *Image Processing Toolbox 6.4*. We then compare the techniques implemented in the previous paper (see [15]) and the images, letter *O*, *Cameraman,* and *Rice,* using vertical, horizontal, Sobel, and Canny transforms implementing the software, *MATLAB 7.9.0*, and *Image Processing Toolbox 6.4*.

1. Introduction

To help motivate this paper, we provide an introduction to the edger problem in

Received: January 11, 2011

Figure 1: Detection and outlining of a prostate cancer cell using segmentation and morphology. Original images courtesy of Alan W. Partin, M.D., Ph.D., John Hopkins University School of Medicine.

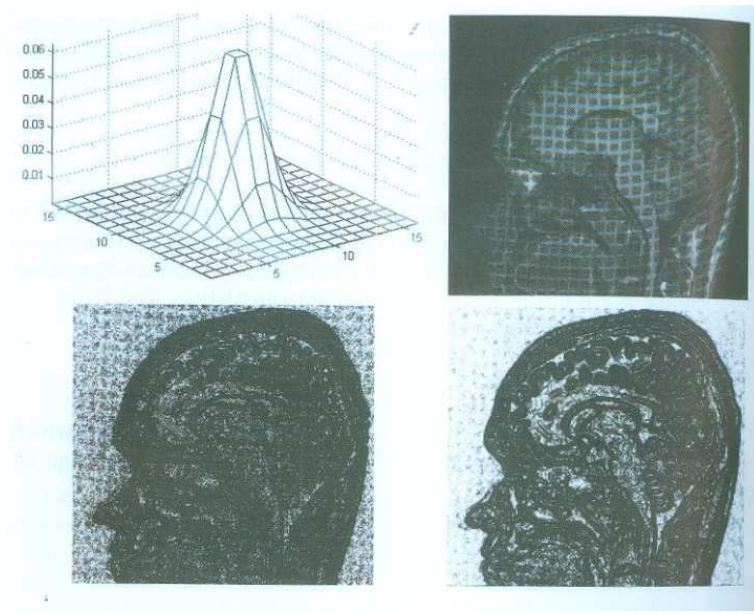

Figure 2: Top Left: Gaussian kernel. Top Right: Noisy MR brain image. Bottom Left: Edge functional. Bottom Right: Edge functional using Gaussian filtering.

image processing by implementing matrix techniques, partial derivatives, and convolutions. Section 2 provides an introduction to matrix and partial derivatives and how they are applied to the pixels to obtain the gray-level value in black and white images. Section 3 introduces the mathematical requirements for a few specific examples, such as the vertical, horizontal, and Sobel edge detectors. Section 4 provides the reader with a series of illustrations that demon-

Figure 3: Angiogram image of an aortic aneurysm.

strate edging techniques in a three-dimensional image, and two black and white images. We compare results by developing mathematical procedures, including convolutions using *MATLAB 7.9.0* versus using the *Image Processing Toolbox 6.4.*

2. Some Notions and Notations

A current laptop in advertisements displays an image using 1680x1050 pixels. Increasing the number of pixels improves the resolution of the image. The number of pixels continues to increase every day as technology progresses. Each pixel location, designated by the coordinates, (x_i, y_j), contains the gray-scale value within the image at that point. The values are on a scale of 0 to 255, whereby 0 corresponds to white and 255 corresponds to black. The gray-scale value at the lattice point, (x_i, y_j), is designated by $f(x_i, y_j)$. Before we continue with the edge detection analysis, to remind the reader, we briefly review a few matrix and calculus techniques. These mathematical techniques are implemented in this paper. We first recall the familiar dot product for two vectors, $\mathbf{x}$, $\mathbf{y}$, which is $\mathbf{x}\bullet\mathbf{y}=\sum_{i=1}^{2} x_i y_i$. From this dot or inner product we define the norm to be $\|x\|^2 = \sum_{i=1}^{2} x_i y_i$. Then we obtain the familiar and very important result

to many applications: the cosine of the angle between the two vectors, $\mathbf{x}$ and $\mathbf{y}$, satisfies the equation that $\cos(\theta)=x\bullet y/(\|x\|\,\|y\|)$. We know the maximum value for the cosine occurs when the two vectors coincide, giving a value, cosine(0)=1. This is an important observation in edge detection and will later be explained. We now evaluate grey-scale values between neighboring pixel locations. This will be determined by introducing the partial derivative formulas,

$$\frac{\partial f(x,y)}{\partial x} = \lim_{\Delta x \to 0} \frac{f(x+\Delta x, y) - f(x,y)}{\Delta x},$$

and

$$\frac{\partial f(x,y)}{\partial y} = \lim_{\Delta y \to 0} \frac{f(x, y+\Delta y) - f(x,y)}{\Delta y}.$$

The distance between pixel locations is normalized to be 1, so all of the increments in the partial derivative formulae will be equal to one. This then gives,

$$\frac{\partial f(x,y)}{\partial x} \approx \frac{f(x+1,y) - f(x,y)}{1},$$

and

$$\frac{\partial f(x,y)}{\partial y} \approx \frac{f(x,y+1) - f(x,y)}{1}.$$

We select the gray-scale values given by the function, $f(x,y)$, between neighboring pixels in the horizontal and vertical directions, respectively, giving us the formulas, $f(x_{i+1},y_j)-f(x_i,y_j)$ and $f(x_i,y_{j+1})-f(x_i,y_j)$. The spatial locations, x_i and y_j, can only take on integer values given by their integer locations.

3. Convolution and Edge Detectors

To compute the adjacent differences between neighboring pixel locations, we introduce the usual calculus definition for convolution given by the formula,

$$h(x,y) * f(x,y) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} h(k_1,k_2) f(x-k_1, y-k_2) dk_1 dk_2,$$

and its discrete version by the formula,

$$h(n_1,n_2) * f(n_1,n_2) = \sum_{k_1=-\infty}^{\infty}\sum_{k_2=-\infty}^{\infty} h(k_1,k_2) f(n_1-k_1, n_2-k_2).$$

Research efforts reported that we can reduce the discrete convolution kernel to be a special three by three matrix, which will play the role of a convolute, and select our function, $h(n_1, n_2)$, to have the matrix values,

$$\mathbf{h} = \begin{pmatrix} h(-1,1) & h(0,1) & h(1,1) \\ h(-1,0) & h(0,0) & h(1,0) \\ h(-1,-1) & h(0,-1) & h(1,-1) \end{pmatrix} = \begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix}.$$

The arguments (n_1, n_2) in $h(n_1, n_2)$ of the first array are easily remembered by noting that they are the needed lattice point coordinates referred to as a Cartesian coordinate system. This is illustrated in Figure 4. Clearly, the reduced array for $h(n_1, n_2)$ is part of the complete array, where $h(n_1, n_2)$ is equal to zero whenever $|n_1|$ or $|n_2| > 2$. Next, we convolve the function, $h(n_1, n_2)$, with the function, f(n_1,n_2), and obtain

$$\begin{aligned} h(n_1, n_2) * f(n_1, n_2) &= \sum_{k_1=-1}^{1} \sum_{k_2=-1}^{1} h(k_1, k_2) f(n_1 - k_1, n_2 - k_2) \\ &= f(n_2 - 1, n_2 + 1) - f(n_1 + 1, n_2 + 1) + f(n_1 - 1, n_2) \\ &\quad - f(n_1 + 1, n_2 + 1) + f(n_1 - 1, n_2 + 1) - f(n_1 + 1, n_2 - 1). \end{aligned}$$

Investigating this last result reveals that it gives the difference of three columns of pixel values in the horizontal direction. If we check the literature (see [8], [9]), we find that this is the approximation used in the horizontal direction in several leading software image-processing packages. The function, $h(n_1, n_2)$, is called the *kernel* of the convolution, and when we change its values, we obtain different edges. The edge is the portion of the image where there is a sudden change in gray-scale values. The edger implemented selects a particular feature in the image, which is beneficial to the particular application. The kernel for vertical edging is given by

$$\mathbf{h} = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{pmatrix}.$$

A more sophisticated edger is the Sobel edger, which uses the gradient to approximate the edges. Since the gradient includes both horizontal and vertical components, two kernels are employed, given by the matrices,

$$\begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}, \quad \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}.$$

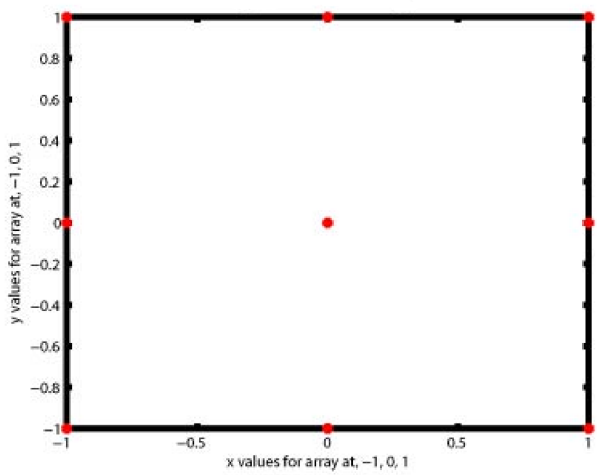

Figure 4: The lattice array.

4. Illustrations Using Edge Detectors

Figures 5-8 show the completed letter, O, using mathematical techniques briefly included in Section 2 and convolutions briefly described in Section 3. Figures 5-8 employ mathematics using *MATLAB 7.9.0* and *not* the *Toolbox Software 6.4.* Figure 5 illustrates the letter O. We then employ a vertical edge detector on the letter O shown in Figure 6. Again, a horizontal edge detector and Sobel Transform are applied on the letter O and illustrated in Figures 7 and 8, respectively.

We now use the *Image Processing Toolbox Version 6.4* to compare the mathematical development using *MATLAB 7.7.0.* The toolkit must have a single matrix whereby one must convert a JPEG format file to a gray-scale format file. The *MATLAB 7.7.0* command for this is as follows:

$$J = \text{rgb}\,2\,\text{gray}\,(I),$$

where I is the image file in JPEG format and J is the file converted to gray-scale format. We select the image, *Letter O.tif* illustrated in Figure 5, convert it to the gray-scale format, and apply the *Sobel Edge Detector* to it as illustrated in Figure 9. We note the difference for the Sobel Transform of the Letter O (illustrated in Figure 8) and the *Matlab Toolbox* using the *Sobel Edge Detector* (see Figure 9). We also include the Canny Edge detector illustrated in Figure 10.

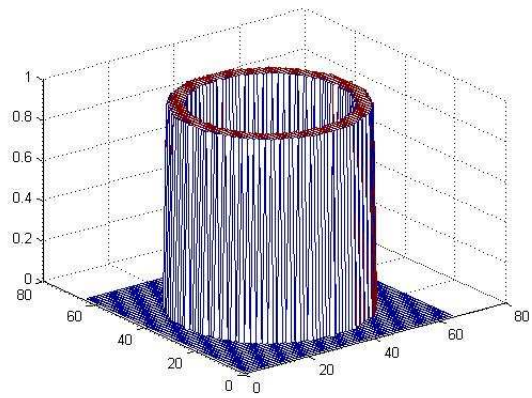

Figure 5: The letter O.

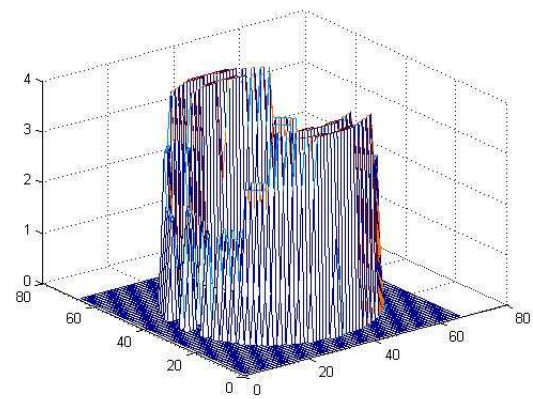

Figure 6: A vertical edge detector on the letter O.

We import the black and white example image, *cameraman.tif*, included in the *MATLAB Imaging Process Toolkit Library* (see Figure 11) to extract the edges of the image. We apply the Sobel Edge Detector contained in the mathematical development using *MATLAB 7.7.0* and illustrate the results in Figure 14. We can see a stronger edge detection in Figure 14 as compared to the *MATLAB Imaging Process Tookit* implementing the *Sobel and Canny Edge Detectors* illustrated in Figures 12 and 13.

Furthermore we import the black and white image example, *Rice.png* (Fig-

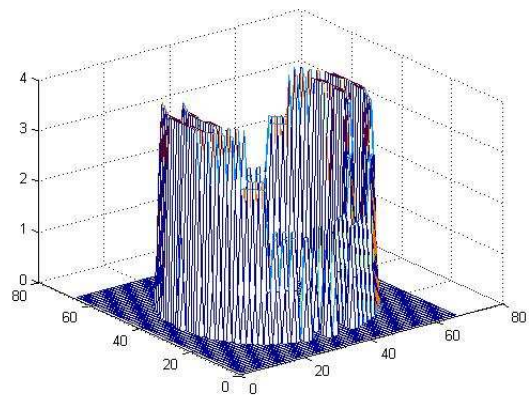

Figure 7: A horizontal edge detector on the letter O.

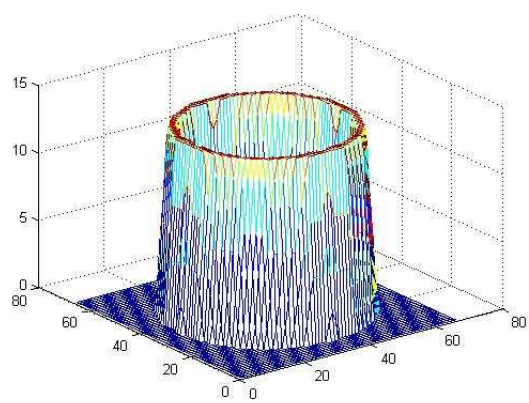

Figure 8: A sobel edge detector on the letter O.

ure 15), contained in the *MATLAB Image Processing Toolkit*. This image shows a large amount of edges, and we can think of comparing this to cells in a blood test. We then consider the image *Rice.png*, apply the Sobel and Canny Edger in the *MATLAB Image Processing Toolkit*, and again compute the Sobel Edger using the mathematical computations for it. They are included in Figures 15-18, respectively. Again, the results are similar to the image *cameraman.tif.*

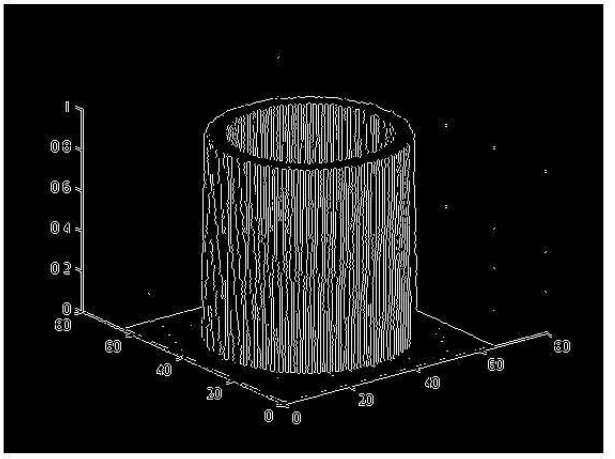

Figure 9: The sobel edge detector using the image processing toolbox version 6.4.

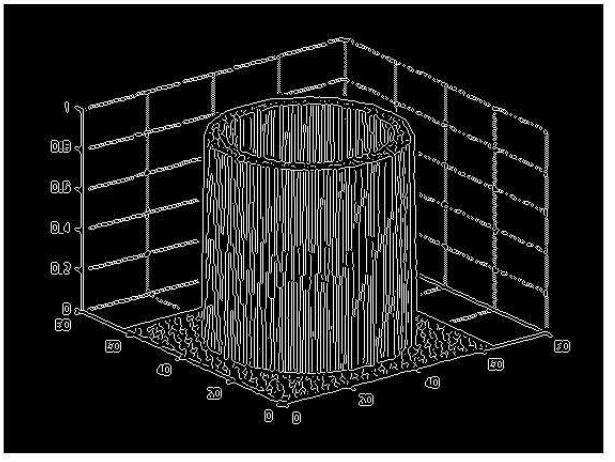

Figure 10: The canny edge detector using the image processing toolbox version 6.4.

Figure 11: The image, Cameraman.

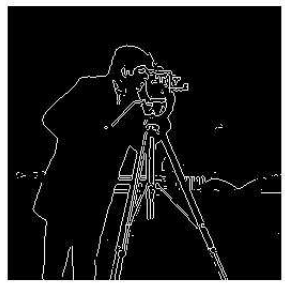

Figure 12: The Sobel Edger on the image, Cameraman.

5. Conclusion

As seen by the previous images, the mathematical development techniques briefly discussed in Sections 3 and 4 illustrate strong edges. The *Image Toolkit*, without any further enhancement techniques included, somewhat submerges the clarity of the edges. However, the particular application being used by the researcher must review both techniques to identify the appropriate desired results for the required goal.

References

[1] H.C. Andrews, B.R. Hunt, *Digital Image Restoration*, Prentice Hall, N.J.

Figure 13: The Canny Edger on the image, Cameraman.

Figure 14: The Sobel Edger on the image, Cameraman using the matrix computations for the edger.

(1977).

[2] D.H. Ballard, Parameter nets, *Artificial Intelligence*, **22** (1984), 235-267.

[3] D.H. Ballard, C.M. Brown, *Computer Vision*, Prentice Hall, N.J. (1982).

[4] B.G. Batchelor, *Pattern Recognition*, Plenum Press, N.Y. (1978).

[5] F.W. Campbell, J.G. Robson, Application of Fourier analysis to the visibility of gratings, *J. Physiol.*, **197** (1968), 551-566.

[6] O. Demirkaya, M.H. Asyali, P.K. Sahoo, *Image Processing with MATLAB-Applications in Medicine and Biology*, CRC Press, Florida (2009).

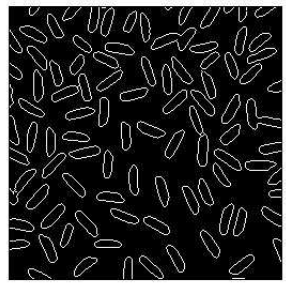

Figure 15: The image, Rice.

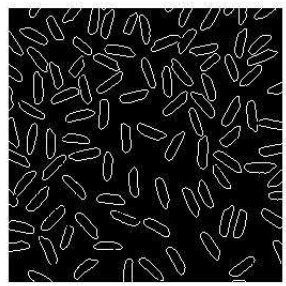

Figure 16: The Sobel Edger on the image, Rice.

[7] R.C. Gonzalez, P. Wintz, *Digital Image Processing*, Addison-Wesley Publ. Co., MA (1987).

[8] A.K. Jain, *Fundamentals of Digital Image Processing*, Prentice Hall, N.J. (1989).

[9] J.S. Lim, *Two-Dimensional Signal and Image Processing*, Prentice Hall, N.J. (1990).

[10] G. Nagy, State of the art in pattern recognition, *Proc. IEEE*, **56** (1968), 836-862.

[11] W. Pedrycz, Fuzzy sets in pattern recognition. Methodology and methods, *Pattern Recognition*, **20**, No-s: 1-2 (1990), 121-146.

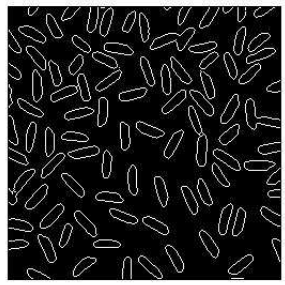

Figure 17: The Canny Edger on the image, Rice.

Figure 18: The Sobel Edger on the image, Rice, using the matrix computations for the edger.

[12] W.K. Pratt, *Digital Image Processing*, John Wiley and Sons, N.Y. (1991).

[13] C.J. Russ, J.C. Russ, *Introduction to Image Processing and Analysis*, CRC Press, Florida (2008).

[14] R.J. Schalkoff, *Digital Image Processing and Computer Vision*, John Wiley and Sons, N.Y. (1989).

[15] J. Schmeelk, Transforms, filters and edge detectors in image processing, *International Journal of Pure and Applied Mathematics*, **46**, No. 2 (2008), 199-208.

[16] I. Zhang, Q.G. Wang, J.P. Qi, Processing technology in microscopic im-

ages of cancer cells in pleural fluid based on fuzzy edge detection method, *Journal of Physics: Conference*, **48** (2006), 329-333.