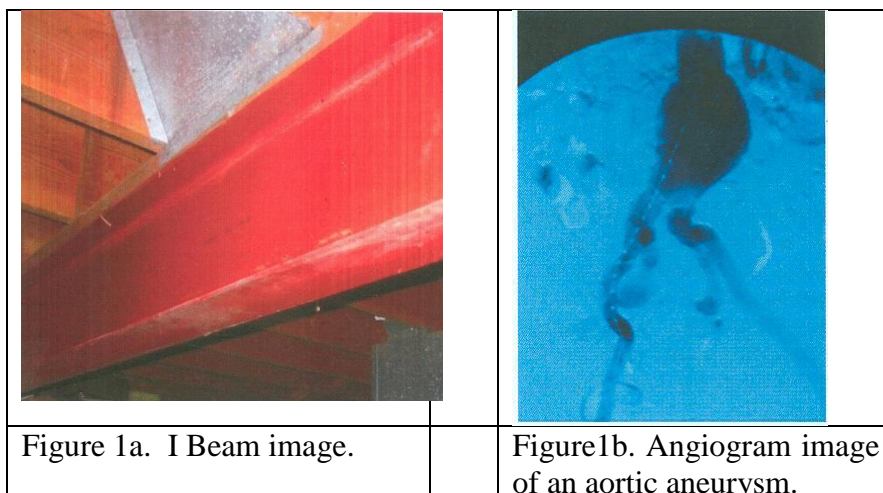

AC 2012-2998: EDGE DETECTORS IN ENGINEERING AND MEDICAL APPLICATIONS

Dr. John Schmeelk, Virginia Commonwealth University, Qatar

Edge Detectors in Engineering and Medical Applications

Abstract

Image edge detection is an integral component of image processing to enhance the clarity of edges and the type of edges. The current paper compares two methods for finding the edges of an image. One method developed by the author is to define special matrices and applying them to the image using approximations for gradients and convolutions. This method then “pulls out” the edges. The second method is using the software, Image Processing Toolbox 6.4. Appendix A compares the relative effectiveness of the two methods. The user’s application will show the variance of the two methods and determine which would work best for the user’s needs. Issues regarding edge techniques were introduced in my 2008 paper on Transforms, Filters and Edge Detectors.¹⁷ showing three dimensional alphabets implementing the author’s matrix results. This paper extends edge detection that can be applied in areas of research, such as engineering and medical applications.^{6,18} An engineering application could be to select an image of an I-beam out of the range of normal eyesight. Then an edge detection technique could be applied to the image to clearly locate any cracking, breaking or delamination found internally or externally to the inspected structure.^{9,11} An I-beam is illustrated in Figure 1a, and the edge technique will be applied to it in Section V to investigate a model of a crack within it. In the field of medicine, a patient can be diagnosed with an aneurysm by studying the shape of the edges in the angiogram. An angiogram is the visual view of the blood vessels (see Figure 1b).The previous paper¹⁵ studied selected letters using vertical, horizontal, and Sobel transforms. This paper will study images to include the letter O and two images, *Cameraman* and *Rice*, that are included in the Image Processing Toolbox 6.4. We compare the techniques implemented by the author using the defined matrix technique on these images, using vertical, horizontal, Sobel, and Canny transforms and compare it to the technique in the Image Processing Toolbox 6.4. We then conclude the paper with a model of the I-beam illustrated in Figure 1a, apply the Sobel transform to it using the MATLAB Image processing Toolbox 6.4, and compare it to the MATLAB 7.9.0 script file developed by the author..



I. Introduction

To help motivate this paper, we provide an introduction to the edge problem in image processing by implementing matrix techniques, partial derivatives and convolutions.

We *define an edger technique* to be a routine that highlights the edges in an image. Section (II) 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 (III) introduces the mathematical requirements for a few specific examples, such as the vertical, horizontal, and Sobel Edge Detectors. Section (IV) provides the reader with a series of illustrations that demonstrate edging techniques on a three-dimensional image (Figure 3) and in images directly taken from a camera. Section (V) shows the reader two images using the gray scale format. Figure 17 is a model image of an I-beam with a crack in it, and Figure 18 uses my script file containing the author's special matrices and implementing the Sobel Edge Detector on the I-beam to locate the crack. We compare results by developing mathematical procedures, including convolutions using the authors technique versus using the Image Processing Toolbox 6.4.

II. Some Notions and Notations

Resolution of an image improves as the number of pixels increase. A current laptop in advertisements displays an image using 1680x1050 pixels. The number of pixels continues to increase everyday as technology progresses. Each pixel location designated by the coordinates, (x_i, y_j) contains a gray level value indicating the shade of gray 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 value of the gray level, at this pixel lattice point, (x_i, y_j) is designated by $f(x_i, y_j)$. Before we continue with the edge detection analysis, we briefly review a few matrix and calculus techniques to familiarize the reader with the mathematical techniques implemented in this paper. We first recall the familiar

dot product for two vectors, \mathbf{x} , \mathbf{y} , to be $\mathbf{x} \bullet \mathbf{y} = \sum_{i=1}^2 x_i y_i$. From this dot or inner product we

define the norm to be $\|\mathbf{x}\|^2 = \sum_{i=1}^2 (x_i^2)$. 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, $\cos(\theta) = \mathbf{x} \cdot \mathbf{y} / (\|\mathbf{x}\| \|\mathbf{y}\|)$. We know the maximum value for the cosine occurs when the two vectors coincide, giving a value, $\cos(0) = 1$. This is an important observation in edge detection and will later be explained. We now evaluate the values of the grey levels between neighboring pixel locations. This will be determined by introducing the partial derivative formulas,

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

and

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

The distance between pixel locations will be 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 now denote the function, $f(x, y)$, to be the gray level values 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.

III. 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,

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

The author defines several special matrices and applies them in a discrete convolution process to obtain the edges to several images in sections IV. The matrix values for these matrices are given as \mathbf{h} below.

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

We note when the author's matrix is selected into MATLAB 7.9.0 the code is introduced as follows:

$$\mathbf{h}=[-1 \ 0 \ 1; -1 \ 0 \ 1; -1 \ 0 \ 1],$$

and the group of numbers between the semi-colons become the rows in the matrix. This observation is essential to be able to locate the edges in an image using the author's special matrix technique.

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 2. 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

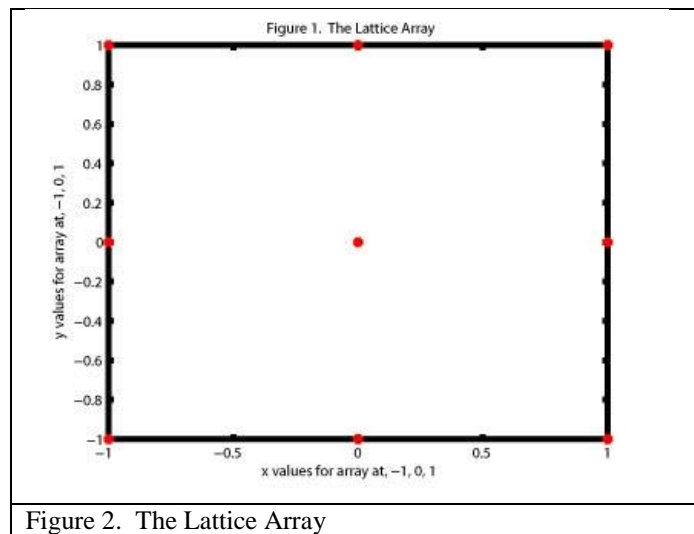
$$\begin{aligned} h(n_1, n_2) * f(n_1, n_2) &= \sum_{k_1=-1}^1 \sum_{k_2=-1}^1 h(k_1, k_2) f(n_1 - k_1, n_2 - k_2) \\ &= \sum_{k_1=-1}^1 (h(k_1, -1)f(n_1 - k_1, n_2 + 1) + h(k_1, 0)f(n_1 - k_1, n_2 - 0) + h(k_1, 1)f(n_1 - k_1, n_2 - 1)) \\ &= \begin{bmatrix} (h(-1, -1)f(n_1 + 1, n_2 + 1) + h(0, -1)f(n_1, n_2 + 1) + h(1, -1)f(n_1 - 1, n_2 + 1) + \\ (h(-1, 0)f(n_1 + 1, n_2) + h(0, 0)f(n_1, n_2) + h(1, 0)f(n_1 - 1, n_2) + \\ (h(1, 1)f(n_1 - 1, n_2 - 1) + h(0, 1)f(n_1, n_2 - 1) + h(1, 1)f(n_1 - 1, n_2 - 1) \end{bmatrix} \\ &= -f(n_1+1, n_2+1) + f(n_1-1, n_2+1) - f(n_1+1, n_2) + f(n_1-1, n_2) - f(n_1+1, n_2-1) - f(n_1-1, n_2-1). \end{aligned}$$

Investigating this last result reveals that it gives the difference of three columns of pixel values in the horizontal direction. The function, $h(n_1, n_2)$, is called the *kernel* of the convolution, and when we change its values, we obtain different edgers. The edge is the portion of the image where there is a sudden change in gray levels. 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 defined as a technique to select the edges in an image 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,

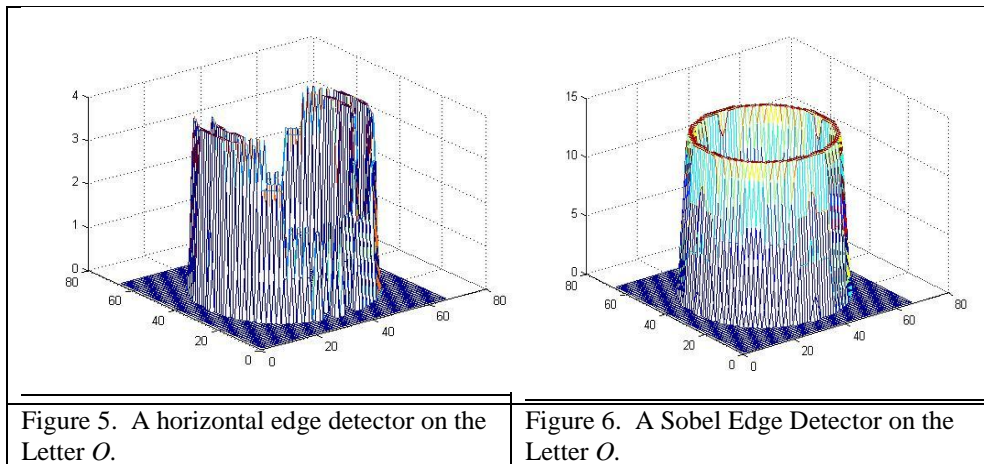
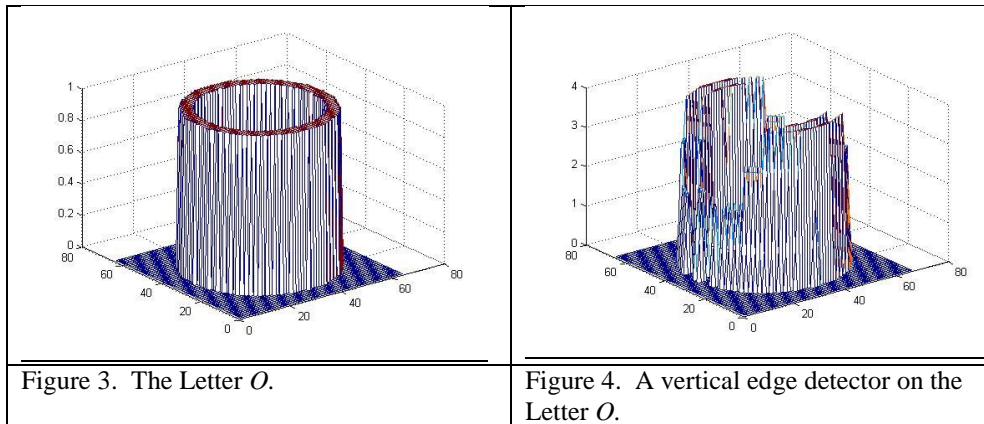
$$\begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}.$$



IV Illustrations using Edge Detectors

Figures 3-6 show the completed letter, *O*, using mathematical techniques briefly included in Section II and convolutions briefly described in Section III. Figures 3-6 employ mathematics using MATLAB 7.9.0 and NOT the Toolbox Software 6.4. Figure 3 illustrates the letter *O*. We then employ a vertical edge detector on the letter *O* shown in

Figure 4. Again a horizontal edge detector and Sobel Transform are applied on the letter *O* and illustrated in Figures 5 and 6, respectively.



We now use the Image Processing Toolbox Version 6.4 to compare the edge detection for the given images. The toolkit requires 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:

$$J = \text{rgb2gray}(I),$$

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 3, convert it to the gray-scale format, and apply the Sobel Edge Detector to it as illustrated in Figure 7. We compare the difference for the Sobel Transform on the Letter *O* (illustrated in Figure 6) and the Matlab Toolbox using the Sobel Edge Detector (see Figure 7). We also compare the Canny Edge Detector illustrated in Figure 8 to that of Figure 6.

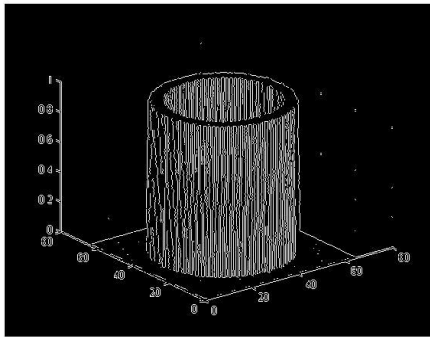


Figure 7. The Sobel Edge Detector using the Image Processing Toolbox Version 6.4.

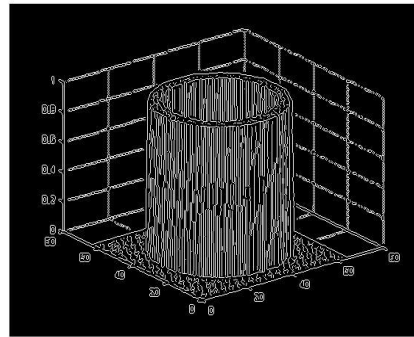
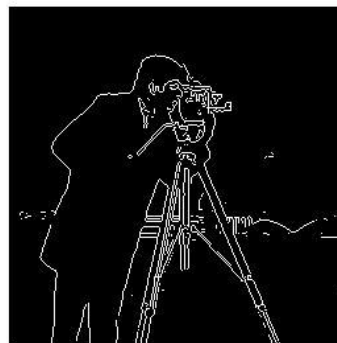


Figure 8. The Canny Edge Detector using the Image Processing Toolbox Version 6.4.

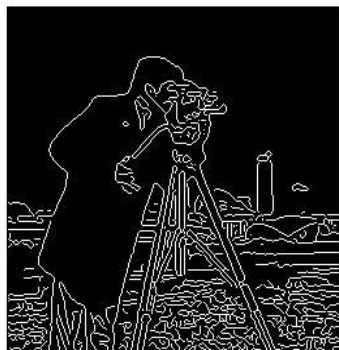
We import the black and white example image, *cameraman.tif* included in the MATLAB Imaging Process Toolkit (see Figure 9) to extract the edges of the image. We apply the mathematical development for the Sobel Edge Detector by implementing the software MATLAB7.7.0 and illustrate the result in Figures 12. We can see a stronger edge detection in Figures 12 as compared to the MATLAB Imaging Process Toolkit implementing the Sobel and Canny Edge Detectors illustrated in Figures 10 and 11.



Figure 9. The image, Cameraman.



10. The Sobel Edger on the image, Cameraman.



11. The Canny Edger on the image, Cameraman.



12. The Sobel Edger on the image, Cameraman using the matrix computations for the edger.

Furthermore we import the black and white image, *Rice.png* (Figure 13) contained in the MATLAB Image Processing Toolkit. We then consider the image, *Rice.png*, and apply the Sobel and *Canny* Edger in the MATLAB Image Processing Toolkit and again compute the Sobel Edger and illustrate the mathematical computations shown for it. They are included in Figures 14-17, respectively. Again, the results are similar to the image *cameraman.tif*.

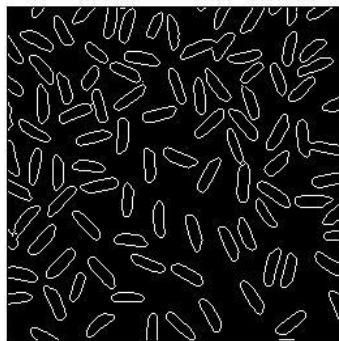


Figure 13. The image, *Rice*.

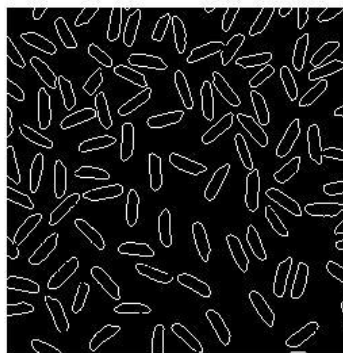
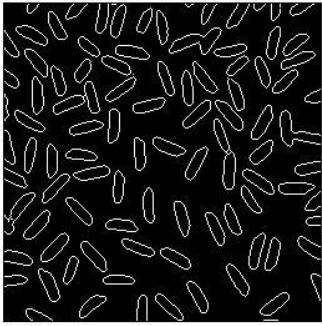

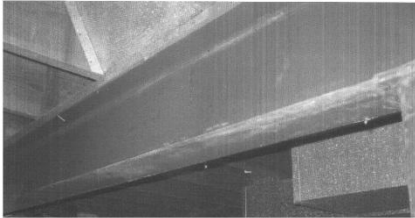
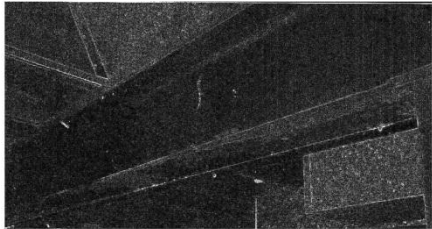


Figure 14. The Sobel Edger on the image, *Rice*

	
<p>Figure 15. The Canny Edger on the image, <i>Rice</i>.</p>	<p>Figure 16. The Sobel Edger on the image, <i>Rice</i>, using the matrix computations for the edger.</p>

V Model of I Beam with a crack

	
<p>Figure 17. Model of an I-beam with a crack</p>	<p>Figure 18. Sobel Edger using my script file</p>

A model of a small crack in the middle of the I-beam is illustrated in Figure 17. Again, with the script file applied to Image 17, one can see a more pronounced crack in the middle of the I-beam. This confirms the premise that an edging technique using my script file written in MATLAB 7.9.0 highlights the edge.

VI Conclusion

As seen by the previous images, the mathematical development techniques briefly discussed in Sections III and IV illustrate strong edges. Appendix A summarizes the relative effectiveness of different edging techniques. 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.

Bibliography

1. Andrews, H.C. & Hunt, B.R., *Digital Image Restoration*, Prentice Hall, N.J., (1977).
2. Ballard, D.H., "Parameter Nets", *Artificial Intelligence*, 22, (1984), 235-267.
3. Ballard, D. H. & Brown, C.M., *Computer Vision*, Prentice Hall, N.J., (1982).
4. Batchelor, B.G., *Pattern Recognition*, Plenum Press, N.Y., (1978).
5. Campbell, F.W., & Robson, J.G., "Application of Fourier Analysis to the Visibility of Gratings", *J. Physiol.* 197, (1968), 551-566.

6. Demirkaya, O., Asyali, M., H., Sahoo, P.K., *Image Processing with MATLAB-Applications in Medicine and Biology*, CRC Press, Florida, (2009).
7. Gonzalez, R.C., &Wintz, P., *Digital Image Processing*, Addison-Wesley Publ. Co., MA. (1987).
8. Jain, A., K., *Fundamentals of Digital Image Processing*, Prentice Hall, NJ, (1989)
9. Kalanad, A. and Rao, B., N., *Detection of Crack location and size in structures using improved damaged finite elements*, IOP Conf. Series: Materials Science and Engineering, IOP Publishing, 10, (2010), 1-10.
10. Lim, J., S., *Two-Dimensional Signal and Image Processing*, Prentice Hall, NJ, (1990).
11. Mannan, M.,A., *Detection and Location of Structural Cracks using FRF Measurements*, International Model Analysis Conference, Orlando, Fl, (1990), 1-6.
12. Nagy, G., "State of the Art in Pattern Recognition", *Proc. IEEE*, 56, (1968), 836-862.
13. Pedrycz, W., "Fuzzy Sets in Pattern Recognition; Methodology and Methods", *Pattern Recognition*, 20 No. 1-2, (1990), 121-146.
14. Pratt, W., K., *Digital Image Processing*, John Wiley & Sons, NY, (1991).
15. Russ, C. J. and Russ, J. C., *Introduction to Image Processing and Analysis*, CRC Press, Florida, (2008).
16. Schalkoff, R. J., *Digital Image Processing and Computer Vision*, John Wiley & Sons, NY, (1989).
17. Schmeelk, J., "Transforms Filters and Edge Detectors in Image Processing", *International Journal of Pure and Applied Mathematics*, 46, No. 2, (2008), 199-208.
18. Zhang, I., Wang, Q.,G., Qi, J., P., "Processing Technology in Microscopic Images of Cancer Cells in Pleural Fluid Based on Fuzzy Edge Detection Method", *Journal of Physics: Conference*, 48, (2006), 329-333.

Appendix A
RELATIVE EFFECTIVENESS OF DIFFERENT
EDGING TECHNIQUES

Figures	Parameters	Techniques	Different views	Comments
Figure 4 vertical edge detector on letter O.	vertical edge matrix, $\begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$	Calculations using my Matlab 7.9.0 program	Different view of the vertical edge for the letter O	This is a three dimensional construction using a Matlab 7.9.0 platform developing the letter O
Figure 5 horizontal edge detector on letter O.	horizontal edge matrix, $\begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}$	Calculations using my Matlab 7.9.0 program	Different view of the horizontal edges for the letter O	This is a three dimensional construction using a Matlab 7.9.0 platform developing the Letter O
Figure 6 Sobel edge detector on letter O.	Uses both vertical and horizontal edge matrices,	Calculations using my Matlab 7.9.0 program	Different view of the Sobel edges for the letter O	This is a three dimensional construction using a Matlab 7.9.0 platform developing the Letter O

	$\begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$ <p style="text-align: center;">and</p> $\begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}.$			
Figure 7 Sobel Edge detector on letter O		Using Image Processing Toolbox Version 6.4	Must change the image to a gray-scale format	A little too dark for my purposes. I just used the software without any enhancements.
Figure 8 Canny Edge detector on letter O		Using Image Processing Toolbox Version 6.4	Must change the image to a gray-scale format	A little too dark for my purposes. I just used the software without any enhancements.
Figure 10 Sobel Edge detector on image Cameraman		Using Image Processing Toolbox Version 6.4	Cameraman image is in the Toolbox library.	A little too dark for my purposes. I just used the software without any enhancements.
Figure 11 Canny Edge detector on image Cameraman		Using Image Processing Toolbox Version 6.4	Cameraman image is in the Toolbox library	A little too dark for my purposes. I just used the software without any enhancements.
Figure 12 Sobel Edge detector on image Cameraman	<p>Uses both vertical and horizontal edge matrices,</p> $\begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$ <p style="text-align: center;">and</p> $\begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}.$	Calculations using my Matlab 7.9.0 program	Use my Matlab 7.9.0 on the image Cameraman.	Sharp and clear for my purpose. I just used my matrix calculations without any enhancements.
Figure 14 Sobel Edge detector on image Rice		Using Image Processing Toolbox Version 6.4	Rice image is in the Toolbox library.	A little too dark for my purposes. I just used the software without any enhancements.
Figure 15 Canny Edge detector on image		Using Image Processing Toolbox Version 6.4	Rice image is in the Toolbox library	A little too dark for my purposes. I just used the software without any enhancements.

Rice				
Figure 16 Sobel Edge detector on image Rice	Same as parameters in Figure 12	Calculations using my Matlab 7.9.0 script file	Use my Matlab 7.9.0 on the image Rice.	Sharp and clear for my purpose. I just used my matrix calculations without any enhancements.
Figure 17			Must change to gray scale format	
Figure 18		Calculations using my Matlab 7.9.0 script file	Must change to gray scale format	