Elysium Journal
Engineering Research & Management

P-ISSN: 2347-4408
E-ISSN: 2347-4734

VIRTUAL IMAGE RENDERING AND STATIONARY RGB COLOUR CORRECTION FOR MIRROR IMAGES

S.Malathy[1] R.Sureshkumar[2] V.Rajasekar[3]
[1]Second Year M.E (Computer Science and Engineering)
[2]Assistant Professor (Computer Science and Engineering)
[3]Assistant Professor (Computer science and Engineering)
[1, 2] Maharaja Prithvi Engineering College, Avinashi - 641654
[3]Muthayammal College of Engineering, Rasipuram
[1]malu.chandru@gmail.com
[2]Sweetsuresh17@gmail.com
[3]brahmaasthra@gmail.com

ABSTRACT
The key idea is to develop an application for digital image processing which concurrently process both stationary images and RGB Colour Corrections. It deals with two types of input, namely system based input and camera based input. The system based input will be the manual input from the user without any hardware devices, the mirror and RGB images will be available in the system user can use these images for image rendering. Another method of input will be Camera based input, for camera method a black box based camera box will be created, the camera will be connected with personal Computer through universal serial port. Whenever the images get placed before the camera, the camera can be operated from the system. So that the image can be capture from the camera and given as the input image. Using image rendering, edge detections and support vector machine method the image will be recognized by the application. Despite rapid progress in mass-storage density, processor speeds, and digital communication system performance, demand for data storage capacity and data-transmission bandwidth continues to outstrip the capabilities of available technologies.

General Terms
Edge Detection, Laplacian of Gaussian, Canny Edge Detection, Sobel Edge Detection Methodologies, SVM Classification.

Keywords -- Mirrors, Image reconstruction, RGB system, Depth image denoising, Support Vector Machine.

1. INTRODUCTION

Digital image processing is an area that has seen great development in recent years. It is an area which is supported by a broad range of disciplines, where signal processing and software engineering are among the most important. Digital image processing applications tend to substitute or complement an increasing range of activities. Applications such as automatic visual inspection, optical character recognition object identification etc are increasingly common.

Digital image processing studies the processing of digital images, i.e., images that have been converted into a digital format and consist of a matrix of points representing the intensities of some function at that point. The main objectives are related to the image improvement for human understanding and 'the processing of scene data for autonomous machine perception'.

This later task normally comprises a number of steps. The initial processing step is the segmentation of the image into meaningful regions, in order to distinguish and separate various components. From this division, objects can then be identified by their shape or from other features. This task usually starts with the detection of the limits of the objects, commonly designated as edges. Effectively a human observer is able to recognize and describe most parts of an object by its contour, if this is properly traced and reflects the shape of the object itself. In a machine vision system this recognition task has been approached using a similar technique, with the difference being that a computer system does not have other information than that which is built into the program. The success of such a recognition method is directly related to the quality of the marked edges.

Under 'engineered' conditions, e.g. backlighting, edges are easily and correctly marked. However, under normal conditions where high contrast and sharp image are not achievable detecting edges become difficult. Effectively as contrast decreases the difficulty of marking borders increases. This is also the case when the amount of noise present in the image increase, which is 'endemic' in some applications such as x-rays. Common images e.g. from interior scenes although containing only small amounts of noise, present uneven illumination conditions. This diminishes contrast which affects the relative intensity of edges and thus complicates their classification. Finally, image blur due to imperfections in focus and lens manufacture smoothes the discontinuities that are typical from edges and thus once again makes the edges difficult to detect.

P-ISSN: 2347-4408
E-ISSN: 2347-4734

These problems prompted the development of edge detection algorithms which, to a certain degree of success, are able to cope with the above adverse conditions. Under suitable conditions most of the edge detection algorithms produce clear and well defined edge maps, from which objects within the image are easily identified. However, the produced edge maps degrade as the image conditions degrade. Not only misplacements of the shape occur, spurious features appear and edge widths differ from algorithm to algorithm. It may be hypothesized that edge maps produced by different algorithms complement each other. Thus it may be possible to override some of the vagueness by comparison between different edge maps.

2. EDGE DETECTION OBJECTIVES

The interest in digital image processing came from two principal application areas: improvement of pictorial information and processing of scene data for autonomous robot classification within autonomous machine perception. In the second area, where the most primordial motivation of this thesis is based and in which edge detection is used. The first processing steps are the identification of meaningful areas within the picture. This process is called segmentation. It represents an important early stage in image analysis and image identification. Segmentation is a grouping process in which the components of a group are similar in regard to some feature or set of features. Given a definition of "uniformity", segmentation is a partitioning of the picture into connected subsets each of which is uniform but such that no union of adjacent subsets is uniform. There are two complementary approaches to the problem of segmenting images and isolating objects - boundary detection and region growing. Edge oriented methods generally lead to incomplete segmentation. The resulting contours may have gaps or there may be additional erroneous edge elements within the area. The results can be converted into complete image segmentation by suitable post processing methods, such as contour following and edge elimination. Region growing is a process that starts from some suitable initial pixels and using iterations neighboring pixels with similar properties are assigned, step by step, to sub regions. A suitable basis for this type of segmentation could be a thresholding process.

The intensity of reflected light is a function of the angle of the surface to the direction of the incident light. This Leeds to a smooth variation of the light intensity reflected across the surface as its orientation to the direction of the light source changes, which cannot be considered as an edge. Also shadows give sharp changes in the brightness within an image of a smooth and flat surface. This does not represent the limit of an object. In the other extreme, such as in the case of technical drawing, where there are thin lines drawn, where no discontinuity on the represented object exists, but which are important for the understanding of the shape of an object. The relation between edges and grey level discontinuities is not clear, and a decision can only be made where an understanding of the image exists (which, in some way, is the ultimate goal of the whole image processing process). As Vicky Bruce states

There is a relationship between the places in an image where light intensity and spectral composition change, and the places in the surroundings where one surface or object ends and another begins, but this relation is by no means a simple one. There are a number of reasons why we cannot assume that every intensity or spectral change in an image specifies the edge of an object or surface in the world

An edge the grey level is relatively consistent in each of two adjacent, extensive regions, and changes abruptly as the order between the regions is crossed. Effectively there are many well known paradoxes in which an edge or contour is clearly seen where none physically exists. This is due to the characteristics of our perception capabilities, and our tendency to group similar information as described in Gestalt's approaches to perception or to infer edges from the context of the image.

The weakness and limitations of the concept can be shown through an image with two areas. Between them grey levels present a linear varying discontinuity from 0 to S. Lets assume that the two areas present a linear varying grey level with the ranges [0.. n] and [0.. n+S] respectively, and such that n/x «S. A schematic three-dimensional graph of such an image is presented in figure 1.

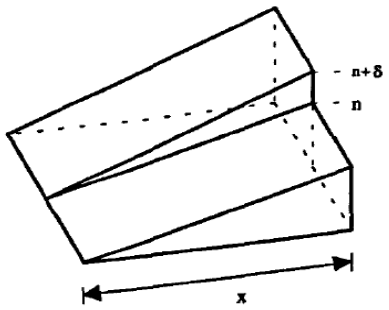

Fig 1 : Grey level 3D plot of an image which consists of two distinct areas

In an image defined like this the edge, as the discontinuity between the two surfaces, will be marked by the same operators with a different extension depending of the value of the parameter n, although the discontinuity itself is independent from it. All the above definitions include some ambiguity in the definition of an edge. Effectively it will be only known at the end of processing, when segmentation has been performed, where the edges are. An edge is a subjective entity, as defined by Blicher. As this author said "The term 'edge' has been fairly abused and we will continue that

P-ISSN: 2347-4408
E-ISSN: 2347-4734

tradition here". Edges appear with several profiles, where the most common are step edges. However ramp edges and roof edges figure 2 are also common.

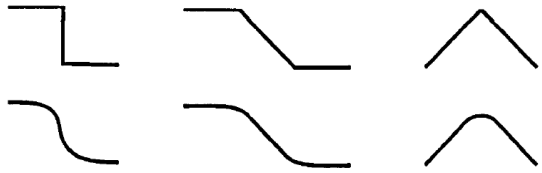

Fig 2 : Common models for edge profiles

Edges also appear with several shapes like curves or straight lines. The most common shape in our environment is vertical or horizontal straight lines. Some particular scenes for instance blood cells do not have straight edges at all. So a particular edge shape cannot be assumed a priori. All these shapes have associated with them some quantity of noise inherent to the acquisition process. These problems can be seen in figure 4 which is a three dimensional plot of the image from the C key of the keyboard image in figure 3.

Fig 3 : Image of a keyboard

In particular the grey level variation in the C print clearly visible in the middle as a re-entrance, which is darker than the key is not abrupt.

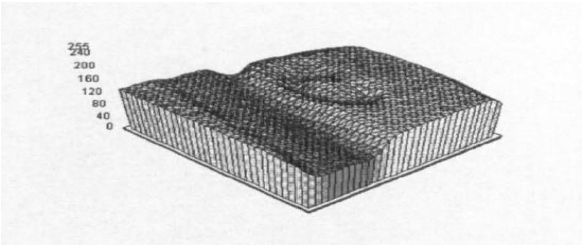

Fig 4: Three dimensional plot of the image from the C key of the computer keyboard

With the known exception of some images used in robotics which are acquired with very high contrast, upon which binary thresholding is carried out, all images from real scenes have a small amount of noise. This can also be originated from external sources to the acquisition system.

3. CANNY EDGE DETECTION

Edges characterize boundaries and are therefore a problem of fundamental importance in image processing. Edges in images are areas with strong intensity contrasts – a jump in intensity from one pixel to the next. Edge detecting an image significantly reduces the amount of data and filters out useless information, while preserving the important structural properties in an image. This was also stated in my Sobel and Laplace edge detection tutorial, but I just wanted reemphasize the point of why you would want to detect edges.

The Canny edge detection algorithm is known to many as the optimal edge detector. Canny's intentions were to enhance the many edge detectors already out at the time he started his work. He was very successful in achieving his goal and his ideas and methods can be found in his paper, "A Computational Approach to Edge Detection". In his paper, he followed a list of criteria to improve current methods of edge detection. The first and most obvious is low error rate. It is important that edges occuring in images should not be missed and that there be NO responses to non-edges. The second criterion is that the edge points be well localized. In other words, the distance between the edge pixels as found by the detector and the actual edge is to be at a minimum. A third criterion is to have only one response to a single edge. This was implemented because the first 2 were not substantial enough to completely eliminate the possibility of multiple responses to an edge.

Based on these criteria, the canny edge detector first smoothes 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. 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. 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.

In order to implement the canny edge detector algorithm, a series of steps must be followed. The first step is to filter out any noise in the original image before trying to locate and detect any edges. And because the Gaussian filter can be computed using a simple mask, it is used exclusively in the Canny algorithm. Once a suitable mask has been calculated, the Gaussian smoothing can be performed using standard convolution methods. A convolution mask is usually much smaller than the actual image. As a result, the mask is slid over the image, manipulating a square of pixels at a time. The larger the width of the Gaussian

P-ISSN: 2347-4408
E-ISSN: 2347-4734

mask, the lower is the detector's sensitivity to noise. The localization error in the detected edges also increases slightly as the Gaussian width is increased. The Gaussian mask used in my implementation is shown below figure 5.

$$\frac{1}{115}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix}$$

Fig 5: Implementation of Gaussian Mask

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). They are shown below figure 6.

$$\begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \quad \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Gx Gy

Fig 6: Estimation of the Gradient in x and y direction

The magnitude, or EDGE STRENGTH, of the gradient is then approximated using the formula: |G| = |Gx| + |Gy|. Finding the edge direction is trivial once the gradient in the x and y directions are known. However, you will generate an error whenever sumX 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 theta = invtan (Gy / Gx)

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). Think of this as taking a semicircle and dividing it into 5 regions as shown in figure7

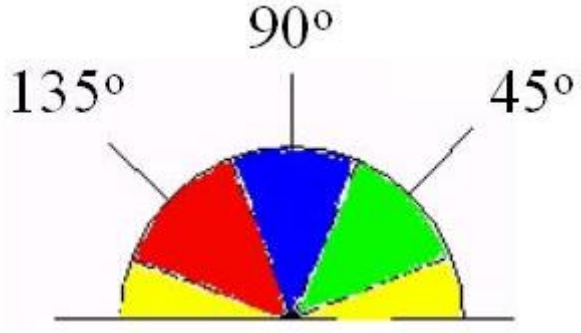

Fig 7: Dividing the semicircle and orientation angle of Edge orientation

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.

After the edge directions are known, non maximum suppression now has to 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.

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.

4. SOBEL EDGE DETECTION

Edges characterize boundaries and are therefore a problem of fundamental importance in image processing. Edges in images are areas with strong intensity contrasts – a jump in intensity from one pixel to the next. Edge detecting an image significantly reduces the amount of data and filters out useless information, while preserving the important structural properties in an image. There are many ways to perform edge detection. However, the majority of different methods may be grouped into two categories, gradient and Laplacian. The gradient method detects the edges by looking for the maximum and minimum in the first derivative of the image. 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 as shown in figure 8

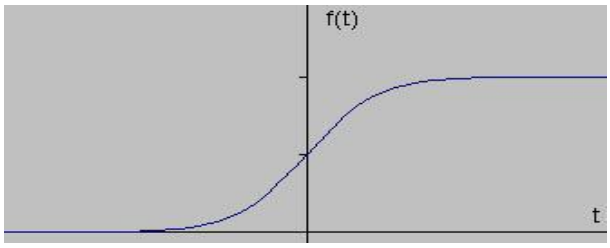

Fig 8: Signal with a an Edge shown by the jump in intensity

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 signal as shown in figure 9

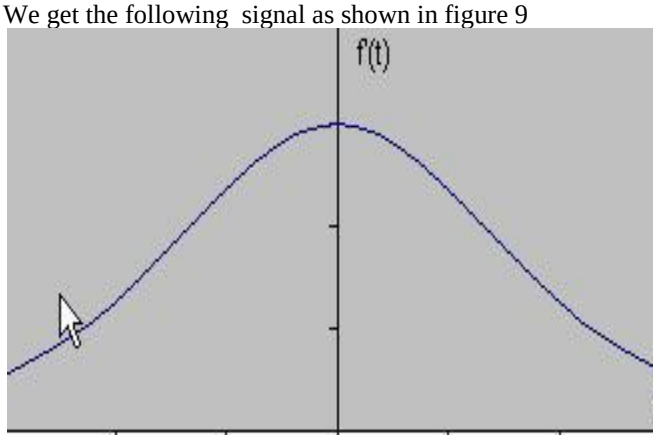

Fig 9: Gradient signal with respect to t

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 figure 10

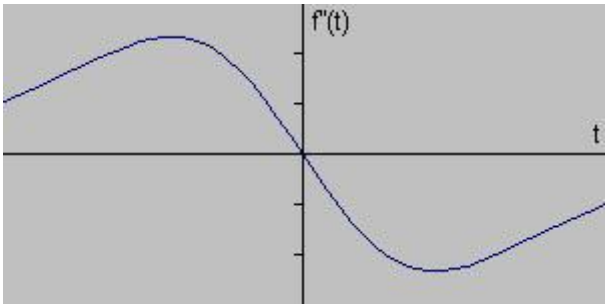

Fig 10: Second Derivative of Laplacian

4.1 Sobel

Based on this one-dimensional analysis, the theory can be carried over to two-dimensions as long as there is an accurate approximation to calculate the derivative of a two-dimensional image. The Sobel operator performs a 2-D spatial gradient measurement on an image. Typically it is used to find the approximate absolute gradient magnitude at each point in an input grayscale image. The Sobel edge detector 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). A convolution mask is usually much smaller than the actual image. As a result, the mask is slid over the image, manipulating a square of pixels at a time. The actual Sobel masks are shown in the figure 11

-1	0	+1
-2	0	+2
-1	0	+1

+1	+2	+1
0	0	0
-1	-2	-1

Fig 11: Actual Sobel Masks

The magnitude of the gradient is then calculated using the formula:

$$|G| = \sqrt{Gx^2 + Gy^2}$$

An approximate magnitude can be calculated using |G| = |Gx| + |Gy|

4.2 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 in order 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:

$$L(x,y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$

Since the input image is represented as a set of discrete pixels, we have to find a discrete convolution kernel that can approximate the second derivatives in the definition of the Laplacian. Three commonly used small kernels are shown in Figure 12

1	1	1
1	-8	1
1	1	1

-1	2	-1
2	-4	2
-1	2	-1

0	1	0
1	-4	1
0	1	0

Fig 12: Small Kernels of Discrete Convolution

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. 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.

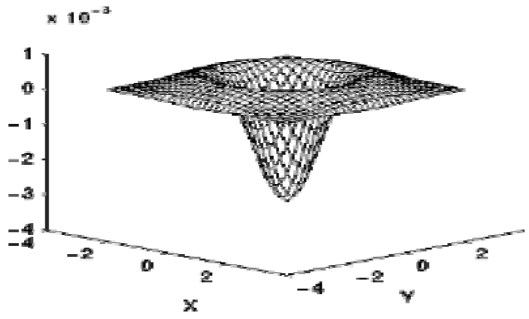

Fig 13 : The 2-D Laplacian of Gaussian (LoG) function

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 *x* and *y* axes are marked in standard deviations. A discrete kernel that approximates this function (for a Gaussian σ = 1.4) is shown in Figure 14

0	1	1	2	2	2	1	1	0
1	2	4	5	5	5	4	2	1
1	4	5	3	0	3	5	4	1
2	5	3	-12	-24	-12	3	5	2
2	5	0	-24	-40	-24	0	5	2
2	5	3	-12	-24	-12	3	5	2
1	4	5	3	0	3	5	4	1
1	2	4	5	5	5	4	2	1
0	1	1	2	2	2	1	1	0

Fig 14: Approximate discrete kernel

Note that as the Gaussian is made increasingly narrow, the LoG kernel becomes the same as the simple Laplacian kernels shown in figure 4. This is because smoothing with a very narrow Gaussian ($\sigma < 0.5$ pixels) on a discrete grid has no effect. Hence on a discrete grid, the simple Laplacian can be seen as a limiting case of the LoG for narrow Gaussians as shown in figure15

Fig 15 : Comparison of Edge Detection Techniques (a)Original Image (b) Sobel (c) Prewitt (d) Robert (e) Laplacian (f)Laplacian of Gaussian

5. SVM CLASSIFICATION

The SVM gives us a simple way to obtain good classification results with a reduced knowledge of the problem. The principles of SVM have been developed by Vapnik and have been presented in several works. In the decision problem we have a number of vectors divided into two sets, and we must find the optimal decision frontier to divide the sets. This optimal election will be the one that maximizes the distance from the frontier to the data. In the two dimensional case, the frontier will be a line, in a multidimensional space the frontier will be an hyperplane. The decision function that we are searching for has the next form.

$$f(\mathbf{x}) = \sum_{i=1}^{l} \alpha_i y_i \langle \mathbf{x}_i \cdot \mathbf{x} \rangle + b \ .$$

The y values that appear into this expression are +1 for positive classification training vectors and –1 for the negative training vectors. Also, the inner product is performed between each training input and the vector that must be classified. Thus, we need a set of training data ($\mathbf{x}$,y) in order to find the classification function. The values are the Lagrange multipliers obtained in the minimization process and the l value will be the number of vectors that in the training process contribute to form the decision frontier. These vectors are those with a value not equal to zero and are known as support vectors.

When the data are not linearly separable this scheme cannot be used directly. To avoid this problem, the SVM can map the input data into a high dimensional feature space. The SVM constructs an optimal hyperplane in the high dimensional space and then returns to the original space transforming this hyperplane in a non-linear decision frontier. The nonlinear expression for the classification function is given below where K is the kernel that performs the non-linear mapping.

$$f(\mathbf{x}) = \sum_{i=1}^{l} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b$$

The choice of this non-linear mapping function or kernel is very important in the performance of the SVM. One kernel used in our previous work is the radial basis function. This function has the expression given in

$$K(x, y) = \exp\left(-\gamma (x - y)^2\right)$$

The γ parameter in the equation must be chosen to reflect the degree of generalization that is applied to the data used. Also, when the input data is not normalized, this parameter performs a normalization task. When some data into the sets cannot be separated, the SVM can include a penalty term (C) in the minimization, that makes more or less important the misclassification. The greater is this parameter the more important is the misclassification error.

5.1 Edge detection using SVM

In this section we present a new way to detect edges by using the SVM classification. The decision needed in this case is between "the pixel is part of an edge" or "the pixel is not part of an edge". In order to obtain this decision we must extract the information from the images since the entire image is not useful as the input to the SVM. The solution is to form a vector with the

pixels in a window around every one into the image. The window size may be changed to improve the detection. In the case of a 3x3 window a nine components vector is calculated at each pixel except for the border of the image. The formed vectors are used as inputs to the SVM in the training process and when it is applied to the images.

5.2 Detection Method

If we apply the trained SVM (1-2) to an image, a value for each pixel into the image is obtained. This value must be a value positive or negative. We can use the sign of these values to say when a pixel is an edge or not but, this way, a lost of information is produced. It is better to use the values obtained and say that there is a gradual change between "no edge" and "edge". Thus, the value obtained indicates a measure of being an edge or not.

6. ARTIFICIAL NEURAL NETWORK

A multi-layer feed forward artificial neural network (ANN) model for edge detection is discussed. It is well known that ANN can learn the input-output mapping of a system through an iterative training and learning process; thus ANN is an ideal candidate for pattern recognition and data analysis.

The ANN model employed in this research has one input layer, one output layer, and one hidden layer. There are 9 neurons in the input layer; in other words, the input of this network is a 9×1 vector which is converted from a 3×3 mask. There are 10 hidden neurons in the hidden layer; and one neuron in the output layer which indicates where an edge is detected.

Initial test results show that, though the neural network is fully trained by the above 17 binary patterns, the performance of the neural network detector is poor when it is applied to test images. The reason is that all the test images are gray-scale (i.e., the intensities of images ranging from 0 to 255), not binary. Thus, we normalize the gray-scale intensities so they are within the range between 0 and 1. Furthermore, to improve the generalization ability of neural network, fuzzy concepts are introduced during the training phase so that more training patterns can be employed by the neural network. The membership functions are shown in Fig. 4. The grade of membership function, $\mu(x)$, can be defined as:

$$\mu(x) = \exp\left(-\frac{(x-\xi)^2}{2\sigma^2}\right)$$

where $\xi = 1$ for "high intensity" and $\xi = 0$ for "low intensity". In this research, we choose $\sigma = 0.25$.

7. CONCLUSION

Since edge detection is the initial step in object recognition, it is important to know the differences between edge detection techniques. In this paper we studied the most commonly used edge detection techniques of Gradient-based and Laplacian based Edge Detection. The software is developed using MATLAB 7.0.

This work presents a new edge detector that use the SVM to perform the pixel classification between edge and no edge. This new detector reduces the execution time compared with our prior implementation by reducing the number of support vectors and by using a linear SVM. Also the window size can be changed with a reduced time increment and the results obtained with larger window sizes may be compared to those from Canny algorithm considered as a standard comparison.

8. ACKNOWLEDGMENTS

I express my sincere and heartfelt thanks to our chairman Thiru.K.Paramasivam B.Sc., and our Correspondent Thiru.P.Sathiyamoorthy B.E., MBA.,MS., for giving this opportunity and providing all the facilities to carry out this paper. I express my sincere and heartfelt thanks to our respected principal Dr.A.S.Ramkumar M.E.,Ph.D.,MIE., for providing me this opportunity to carry out this paper.

I Wish to express my sincere thanks to Mrs.A.Brinda M.E., Assistant Professor and Head of the Department of Computer Science and Engineering for all the blessings and help rendered during tenure of this paper. I am indebted to my project guide Mr.R.Sureshkumar M.Tech., Assistant Professor and Mr.V.Rajasekar M.E., Assistant Professor in Muthayammal College of Engineering for their constant help and creative ideas over the period of project work.

I express my sincere words of thankfulness to members of my family, friends and all staff members of the Department of Computer Science and Engineering for their support and blessings to complete this paper successfully.

9. REFERENCES

1. Jain, A.K., *Fundamentals of Digital Image Processing*, Prentice Hall, Englewood Cliffs, NJ, 1989
2. Canny, J.F., A computational Approach to Edge Detection, *IEEE Trans. On Pattern Analysis andMachine Intelligence*, Vol. 8, 1986, pp. 679-698.
3. Gómez-Moreno, H., Maldonado-Bascón, S., López-Ferreras, F., Edge detection in noisy images by using the support vector machines. IWANN, *Lecture Notes on Computer Science*, Vol. 2084. Springer-Verlag, Heidelberg, 2001, pp. 685-692.
4. Vapnik, V., *The Nature of Statistical Learning Theory*, Springer-Verlag, New York, 1995.
5. Chih-Chung Chang and Chih-Jen Lin, *LIBSVM :a library for support vector machines*, 2001.

6. Gonzalez, R., Woods, R., *Digital Image Processing*, 3rd Edition. Prentice Hall, 2008.
7. Terry, P., Vu, D., "Edge Detection Using Neural Networks", Conference Record of the Twenty-seventh Asilomar Conference on Signals, Systems and Computers. Nov. 1993, pp.391-395.
8. Li, W., Wang, C., Wang, Q., Chen, G., "An Edge Detection Method Based on Optimized BP Neural Network", Proceedings of the International Symposium on Information Science and Engineering, Dec. 2008, pp. 40-44.
9. He, Z., Siyal, M., "Edge Detection with BP Neural Networks", Proceedings of the International Conference on Signal Processing, 1998, pp.1382-1384
10. Mehrara, H., Zahedinejad, M., Pourmohammad, A., "Novel Edge Detection Using BP Neural Network Based on Threshold Binarization", Proceedings of the Second International Conference on Computer and Electrical Engineering, Dec. 2009, pp. 408-412.
11. Haykin, S., *Neural Networks: A Comprehensive Foundation*, Prentice Hall, 1999.
12. *Neural Network Toolbox User's Guide*, The Mathworks.
13. http://en.wikipedia.org/wiki/Bladder_cancer