Fuzzy System Based Edge Extraction Techniques Using Device-Dependent Color Spaces

Er. Vishal Paika[1], Er. Pankaj Bhambri[2]

***Abstract* — The main objective of this paper is to process the image so that resulting image is more suitable than the original image for a specific application like extraction of edges in a digital face image. Fuzzy logic is a field that is being used in various areas and disciplines like digital image processing. Advances in digital image processing technology have opened up a lot of new avenues in various fields like X-ray imaging, Radiography, Electronic Imaging, Linear Tomography, Laminography or radars.**

This paper necessitates the use of fuzzy expert system – a knowledge base system is a system based on methods and techniques of Artificial Intelligence. They uses knowledge acquired and represented using various knowledge representation techniques like rules etc to solve complex problems. In this paper we have used images from different device-dependent color spaces like RGB, HSV, YIQ and YCbCr for the extraction of edges. A self-built fuzzy inference system having smallest possible mask size of 2X2 window is used as a scanning mask. The mask slides over the whole image line-by-line and step-by-step for the detection of edges in a digital image. The system built is represented by the trapezoidal membership function to model the behavior of four input pixels and one output. This membership function uses two fuzzifier's that is black and white and is evaluated with the help of sixteen rules. Our aim is to analyze best possible color space, whose components when enhanced by a particular value, gives us best results. The results obtained are also compared with in-built Matlab edge detection operators like sobel edge operator and canny edge operator.

Keywords: **Artificial Intelligence, Digital image processing, Edge detection, Fuzzy inference system, Fuzzy Logic, Knowledge base systems.**

I. INTRODUCTION

A digital image is a numeric representation; normally in binary form of a two-dimensional image. It has a finite set of digital values, called picture elements or pixels. The digital image contains a fixed number of rows and columns of pixels. Pixels are the smallest individual element in an image, holding quantized values that represent the brightness of a given color at any specific point or intensity in case of gray scale image. Image processing is referred to processing of a 2D picture by a computer. It is a form of signal processing for which the input is an image, such as a photograph or video frame and the output may be either an image or a set of characteristics or parameters related to the image.

Digital image processing is the use of complex computer algorithms to perform image processing on digital images. Before going to processing an image, it is converted into a digital form. Digitization includes sampling of image and quantization of sampled values. After converting the image into bit information, processing is performed. This processing technique may be Image enhancement, restoration, or compression.

- Er. Vishal Paika is currently pursuing Master's Degree Programme in Computer Science from PTU, Jalandhar, Punjab, India.
- Er. Pankaj Bhambri is Assistant Professor in IT Deptt., GNDEC, Ludhiana, Punjab, India.

Digital image processing is the only practical technology used very widely for feature extraction and pattern recognition. The edge detection techniques are generally based on a special type of digital image processing known as morphological image processing. Digital images or better known as binary images may contain numerous imperfections. In particular, the binary regions produced by simple thresholding are distorted by noise and texture. Morphological image processing put into practice the goals of removing these imperfections by accounting for the form and structure of the image. These techniques can be easily extended to grayscale images also. Most of the algorithms work well with grayscale images.

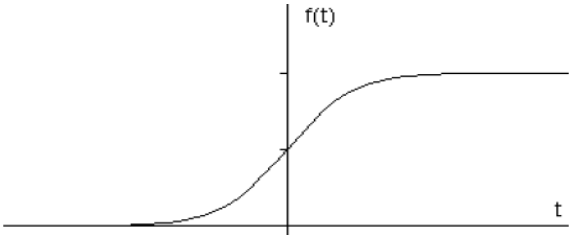
Morphological image processing is a collection of non-linear operations related to the shape or morphology of features in an image. Morphological operations rely only on the relative ordering of pixel values, not on their numerical values, and therefore are especially suited to the processing of binary images. Morphological techniques probe an image with a small shape or template called a structuring element. The structuring element is positioned at all possible locations in the image and it is compared with the corresponding neighborhood of pixels [1].

Edge detection is the name for a set of mathematical methods which aim at identifying points in a digital image at which the image brightness

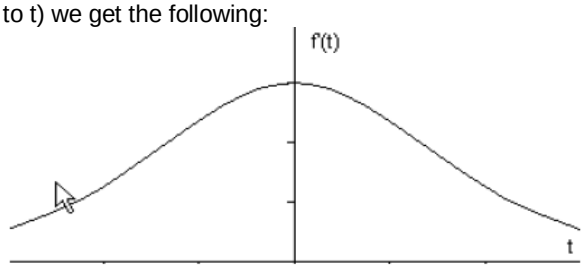
changes sharply or, more formally, has discontinuities. The points at which image brightness changes sharply are typically organized into a set of curved line segments termed edges. Edge detection is a fundamental tool in image processing, machine vision and computer vision, particularly in the areas of feature detection and feature extraction. The purpose of detecting sharp changes in image brightness is to capture important events and changes in properties of the world. It is one of the most commonly used operations in image analysis, and there are probably more algorithms in the literature for enhancing and detecting edges than any other single subject. The reason for this is that edges form the outline of an object. An edge is the boundary between an object and the background, and indicates the boundary between overlapping objects. This means that if the edges in an image can be identified accurately, all of the objects can be located and basic properties such as area, perimeter, and shape can be measured. Since computer vision involves the identification and classification of objects in an image, edge detections is an essential tool [2].

When we take an edge detected image then in practice there may be breaks in boundary between regions because of non-uniform illumination and presence of noise i.e. the edges we get are not always connected. So in that case we have to link the edge points to get some meaningful edges or to extract some meaningful edge information. Edge linking is usually performed by two methods: Local processing approach and Global processing approach. Local processing technique is a similarity based technique. In it we link all other points in edge image which are in the neighborhood of point say (x,y,) and which are similar to the point (x,y). For similarity check, strength of gradient operator and direction of gradient operator is considered. In Global processing approach also know as Hough Transform. It is a mapping from the spatial domain (x,y-plane) to the Parameter domain (m,c-plane).

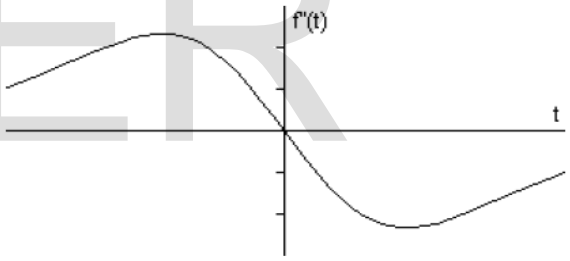
There are many ways to perform edge detection. However, the majority of different methods may be grouped into two categories: Gradient based Edge Detection and Laplacian based Edge Detection. The gradient method detects the edges by looking for the maximum and minimum in the first derivative of the image and 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. Suppose we have the following signal, with an edge shown by the jump in intensity below:



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:



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. 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 [3]. 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:



The two methods used for comparison in this paper are canny edge operator and sobel edge operator. The Sobel operator consists of a pair of 3×3 convolution kernels as shown in Figure 1. One kernel is simply the other rotated by 90°. The 3X3 masks used by Sobel Operator are as shown below:

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

Gx Gy

These kernels are designed to respond maximally to edges running vertically and horizontally relative to the pixel grid, one kernel for each of the two perpendicular orientations. The kernels can be applied separately to the input image, to produce separate measurements of the gradient component in each orientation (call these Gx and Gy). These can then be combined together to find the absolute

magnitude of the gradient at each point and the orientation of that gradient [4]. The gradient magnitude is given by:

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

Typically, an approximate magnitude is computed using: G = |Gx| + |Gy|, which is much faster to compute. The angle of orientation of the edge (relative to the pixel grid) giving rise to the spatial gradient is given by: q = arctan (Gy /Gx).

The canny edge operator on the other hand is based on three criteria's: the first is to have low error rate i.e. edges occurring in images should not be missed and that there be no responses to non-edges, the second 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 and the third is to have only one response to a single edge. The third criteria were implemented because the first two 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 (non-maximum suppression). 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 (made a non edge). 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 [5].

Artificial Intelligence (AI) may be regarded as an attempt to understand the processes of perception and reasoning that underlie successful problem-solving and to incorporate the results of this research in effective computer programs. At present, AI is largely a collection of sophisticated programming techniques. From this perspective, AI seeks to develop systems that attempt to mimic human intelligence without claiming an understanding of the underlying processes involved. Artificial Intelligence (AI) can offer advantages over traditional methods, such as statistical analysis, particularly where the data exhibits some form of non-linearity. Some existing applications of spatial analysis and modeling techniques within the realm GIS include: Rule-based Systems (Fuzzy Logic) and Artificial Neural Networks (ANN) [6].

Fuzzy logic is perhaps the most promising advancement to come along in the Artificial Intelligence community in recent history. Fuzzy logic in its simplest terms expands the dichotomy of true or not true to include a range of answers in between. The usual example is say instead of being black or white, fuzziness allows for shades of gray. Since fuzzy logic allows this extra bandwidth in fuzzy answers, fuzzy rules used in programming can cover a much broader area. A type of logic that recognizes more than simple true and false values. With fuzzy logic, propositions can be represented with degrees of truthfulness and falsehood. For example, the statement, *today is sunny,* might be 100% true if there are no clouds, 80% true if there are a few clouds, 50% true if it's hazy and 0% true if it rains all day. Fuzzy logic has proved to be particularly useful in expert system and other artificial intelligence applications [7].

While variables in mathematics usually take numerical values, in fuzzy logic applications, the non-numeric linguistic variables are often used to facilitate the expression of rules and facts. A linguistic variable such as age may have a value such as young or its antonym old. However, the great utility of linguistic variables is that they can be modified via linguistic hedges applied to primary terms. The linguistic hedges can be associated with certain functions. Fuzzy set theory defines fuzzy operators on fuzzy sets. The problem in applying this is that the appropriate fuzzy operator may not be known. For this reason, fuzzy logic usually uses IF-THEN rules. The AND, OR, and NOT operators of Boolean logic exist in fuzzy logic, usually defined as the minimum, maximum, and complement. There are also other operators, more linguistic in nature, like very, somewhat that can be applied [8].

II. RESEARCH REVIEW

Gao, Y. and Leung, K.H. (2002) proposed "Face Recognition Using Line Edge Map". In this work they generated a compact face feature, Line Edge Map (LEM) together with generic line segment Hausdorff distance measure, for face coding and recognition. A thorough investigation on the proposed concept is conducted which covers all aspects on human face recognition, i.e., face recognition, under 1) controlled/ideal condition and size variation, 2) varying lighting condition, 3) varying facial expression, and 4) varying pose. The results were compared with eigenfaces method and other similar methods where the LEM technique performed superior in most of comparison results [9].

Yinghua, L. et al. (2005) proposed "The application of image edge detection by using fuzzy technique". In this work, they proposed fuzzy enhancement to realize image edge detection. Based on this technology, they firstly set the image fuzzy feature plane of the original image, secondly proceed the fuzzy enhancement, and then detects the edge by Sobel differential arithmetic[10] .

Ryszard, S.C. (2007) proposed "Image Feature Extraction Techniques and Their Applications for CBIR and Biometrics Systems". In this work they use of a number of different color, texture and shape features for image retrieval in CBIR and Biometrics systems. The author described a possible approach for mapping image content onto low-level features. Content-Based Image Retrieval (CBIR), visual

features such as shape, color and texture are extracted to characterize images. Each of the features was represented using one or more feature descriptors. During the retrieval, features and descriptors of the query were compared to those of the images in the database in order to rank each indexed image according to its distance to the query [11].

Sarode, M.V. et al. (2008) proposed "Fuzzy system for color image enhancement". In this work they proposed a fuzzy inference system enhance the image given in HSV color model. As by preserving (H), and changing (S) and (V) we can enhance color image so a Gaussian type membership function is used to model (S) and (V) property of the image. This membership function uses only one fuzzifier and is evaluated by maximizing fuzzy contrast [12].

Mathur, S and Ahlawat, A. (2008) proposed "Application of Fuzzy Logic on Image Edge Detection". In this work they developed the algorithm to find the edges associated with an image by checking the relative pixel values. Exhaustive scanning of an image using the windowing technique takes place which is subjected to a set of fuzzy conditions for the comparison of pixel values with adjacent pixels to check the pixel magnitude gradient in the window. After testing the fuzzy conditions the appropriate values are allocated to the pixels in the window under testing to provide an image highlighted with all the associated edges [13].

Barkhoda, W. et al. (2009) proposed "Fuzzy Edge Detection Based on Pixel's Gradient and Standard Deviation Values". In this work, they had presented a new fuzzy based edge detection algorithm. They had used two different sources of information and a fuzzy system to decide about whether each pixel is edge or not. First both gradient and standard deviation values are computed, form two set of edges, utilized as inputs for our fuzzy system. Then fuzzy system decides on each pixel according to fuzzy rules [14].

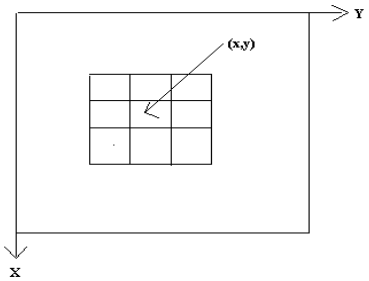
Alshennawy, A.A. and Aly, A.A. (2009) proposed "Edge Detection in Digital Images Using Fuzzy Logic Technique". In their work, they proposed a novel method based on fuzzy logic reasoning strategy for edge detection in digital images without determining the threshold value. The proposed approach begins by segmenting the images into regions using floating 3x3 binary matrixes. The edge pixels are mapped to a range of values distinct from each other. The implemented FIS had greater robustness to contrast and lighting variations, besides avoiding double edges. It also gave a permanent effect in the lines smoothness and straightness for the straight lines and good roundness for the curved lines. In the same time the corners got sharper and could be defined easily [15].

Aborisade, D.O. (2011) proposed "Novel Fuzzy logic Based Edge Detection Technique". In his work he had used three linear spatial filters to generate three edge strength values at each pixel of a digital image through spatial convolution process. These edge strength values were utilized as fuzzy system inputs. Decisions on whether pixels in focus belong to an edge or non-edge were based on the Gaussian membership functions and fuzzy rules. Mamdani defuzzifier method is employed to produce the final output pixel classification of a given image. Experimental results show it less computational expensive and improves the quality of edges as much as compared with Sobel and Kirsch operators [16].

Sandhu, M.S. et al. (2011) presented "Image Edge Detection by Using Rule Based Fuzzy Classifier". In that research work Fuzzy image processing was used for edge detection, which was a collection of fuzzy logic and image processing that understand, represent and process the images as fuzzy sets. The proposed approach was begins by padding the input image and exhaustive scanning of image was done by using the sliding window of 5x5 mask, then pixel values of window were subjected to fuzzy rules designed for horizontal, vertical and diagonal direction, for comparison of pixel values with the adjacent pixels to check, whether the pixel belongs to edge region or non-edge region, an intermediate image was generated as output of fuzzy logic system [17].

III. FUZZY INFERENCE SYSTEM

In this paper we have used fuzzy rule based edge detection system is developed by designing a Fuzzy Inference System (FIS) of Mamdani type using MATLAB version 7.7.0.471 (R2008b) on a personal computer having hardware configuration as: Intel® Core2Duo CPU E7400 having procession speed @ 2.80 GHz, 2GB RAM & 200 GB HDD and the software configuration as: Microsoft Windows Professional XP, version 2002 & service pack 2. In our FIS we have used a smallest possible 2X2 window is used as a scanning mask. The mask has been used for detection of isolated points, lines and edges in a digital image. Given below is an example of the most common 3X3 mask:



$W_{-1,-1}$	$W_{-1,0}$	$W_{-1,+1}$
$W_{0,-1}$	$W_{0,0}$	$W_{0,+1}$
$W_{+1,-1}$	$W_{+1,0}$	$W_{+1,-1}$

$$R = \sum_{i=-1}^{1} \sum_{j=-1}^{1} W_{i,j}\, f(x+i, y+j)$$

In the mask processing operations, we shift the mask, say of size #X#, over the entire image to calculate some weighted sum of pixels at a particular location i.e. we place the mask at location (x,y) of our original image, then we find the weighted sum ® using the equation as shown above. Now depending upon the nature of the image around point (x,y), we will have different values of R. The in-built Matlab edge detection operators like sobel detect isolated points, if the mask has the values as shown below:

-1	-1	-1
-1	8	-1
-1	-1	-1

In a similar way for the detection of horizontal line, vertical line, line inclined at an angle of 45 degree and line inclined at an angle of negative 45 degrees will have the values as shown below. In order to detect an image all the four below mentioned masks are applied simultaneously. (Source: Digital Image Processing by Rafael Ceferino Gonzalez & Richard Eugene Woods, Prentice Hall, 2008, 954 pages).

-1	-1	-1
2	2	2
-1	-1	-1

Horizontal Line

-1	2	-1
-1	2	-1
-1	2	-1

Vertical Line

-1	-1	2
-1	2	-1
2	-1	-1

Diagonal Line at 45°

2	-1	-1
-1	2	-1
-1	-1	2

Diagonal Line at -45°

Mamdani's fuzzy inference method is the most commonly seen fuzzy methodology. Mamdani's method was among the first control systems built using fuzzy set theory. It uses the following graphical tools to build, edit, and view fuzzy inference systems:
1. Fuzzy Inference System (FIS) Editor to handle the high-level issues for the system—How much input and output variables? What are their names? Fuzzy Logic Toolbox software does not limit the number of inputs. However, the number of inputs may be limited by the available memory of your machine. If the number of inputs is too large, or the number of membership functions is too big, then it may also be difficult to analyze the FIS using the other GUI tools.
2. Membership Function Editor to define the shapes of all the membership functions associated with each variable.
3. Rule Editor to edit the list of rules that defines the behavior of the system.
4. Rule Viewer to view the fuzzy inference diagram. Use this viewer as a diagnostic to see, for example, which rules are active, or how individual membership function shapes influence the results.
5. Surface Viewer to view the dependency of one of the outputs on any one or two of the inputs—that is, it generates and plots an output surface map for the system.

These GUIs are dynamically linked, in that changes you make to the FIS using one of them, affect what you see on any of the other open GUIs. For example, if you change the names of the membership functions in the Membership Function Editor, the changes are reflected in the rules shown in the Rule Editor. You can use the GUIs to read and write variables both to the MATLAB® workspace and to a file. You can have any or all of them open for any given system or have multiple editors open for any number of FIS systems.

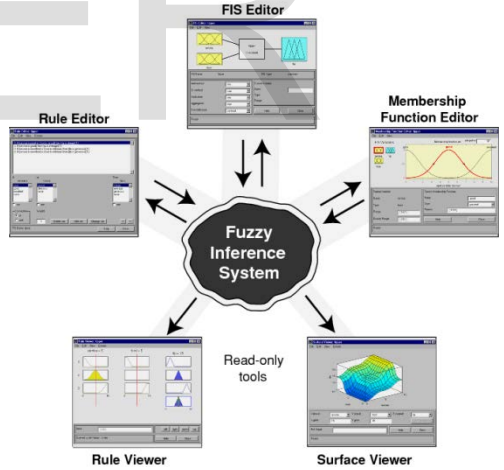


Figure 1: Fuzzy Inference System

The above figure shows how the main components of a FIS and the three editors fit together. The two viewers examine the behavior of the entire system. (Source: http://www.mathworks.in/help/fuzzy/building-systems-with-fuzzy-logic-toolbox-software.html). In the FIS, designed in this system we have used a 2X2 mask for scanning as shown in figure 2 below: -

P_1 Y (i, j)	P_2 Y (i, j+1)
P_3 Y (i+1, j)	P_4 Y (i+1, j+1)

Figure 2: 2×2 mask used for scanning

The above mask slides over the whole image pixel by pixel, first horizontally in topmost horizontal line then after reaching at the end of line, it increments to check the next vertical location and it continues till the whole image is scanned. The FIS built for the system has four inputs (P_1, P_2, P_3 & P_4), where each input representing a pixel for the 2X2 mask, and one output (P_{4_out}) that represents pixel under consideration. The rule editor consists of sixteen fuzzy rules for considering the weights of like P_1, P_2, P_3 and P_4 with P_{4_out} for example if P_1, P_2, P_3 and $P_4$are black then the P_{4_out} is black. The system built is represented by the trapezoidal membership function to model the behavior of four input pixels and one output. This membership function uses two fuzzifier's that is black and white and is evaluated with the help of sixteen rules. The membership functions uses three fuzzy sets i.e. Black, White and Edge for the output.

IV. DESIGN OF FIS

As fuzzy logic is used to detect edges in the image, so we have to design a FIS (Fuzzy Inference System) for this purpose. This work is done according to following steps: 1.

Step 1: First open FIS editor by writing "fuzzy" in the command window of MATLAB, a window will appear, as shown in Figure 3.

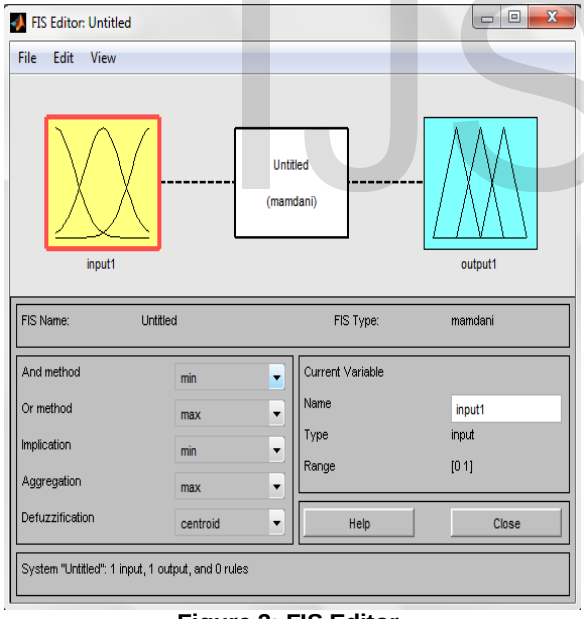


Figure 3: FIS Editor

Step 2: As only one input & one output are shown here, we add another input. In our design, we need four inputs and one output. To add other input or output, click to "edit" and look for "add variable" and in this input/output, as shown in Figure 4. Required input/output variables will appear as shown in Figure 5.

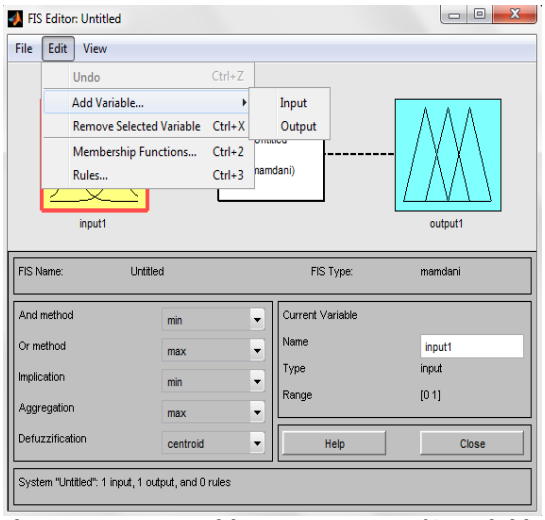


Figure 4: How to add or remove any I/O variable

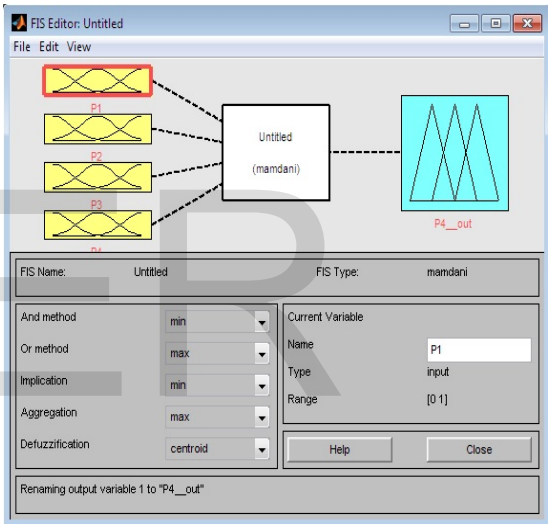


Figure 5: Required input and outputs variables

Step 3: Now by double clicking any one input or output, get all membership plots as shown in Figure 6, Figure 7 and Figure 8.

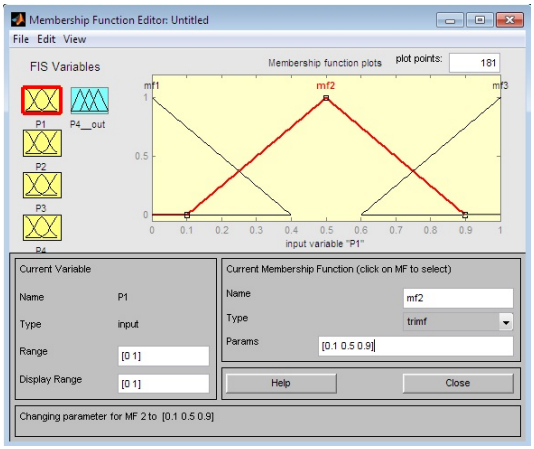


Figure 6: Input variable selection

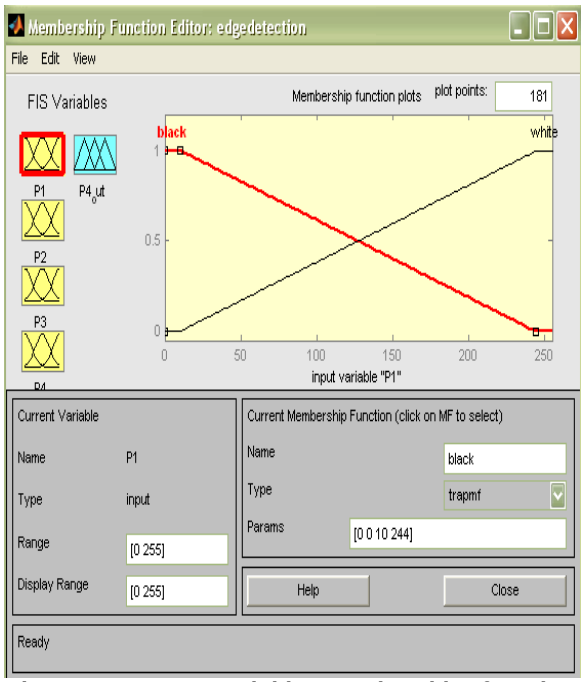


Figure 7: Input Variable Membership function design.

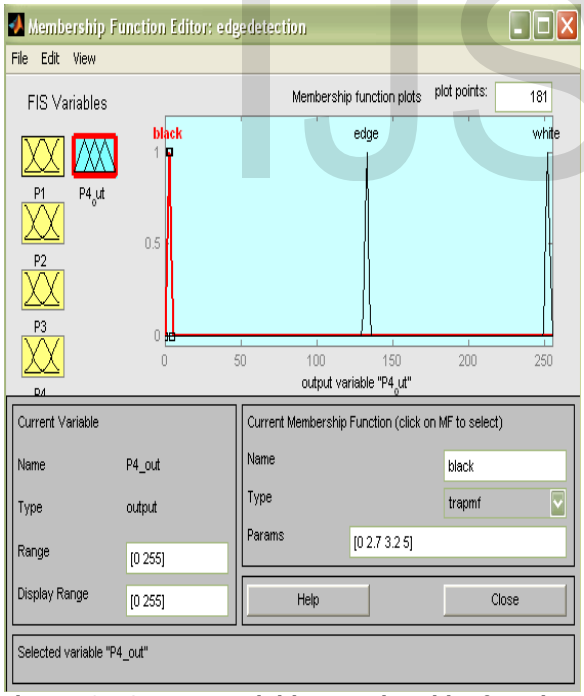


Figure 8: Output Variable Membership function design.

Step 4: Now click centre white box of Figure 9 as shown below to create rules. These rules are defined according to Table 4.2 to control output according to inputs applied. The rules are created by selecting different inputs and output and then clicking "add rule". To delete a rule, first select it and then click "delete rule" and to change a rule, first select it and then select the combination and click change rule.

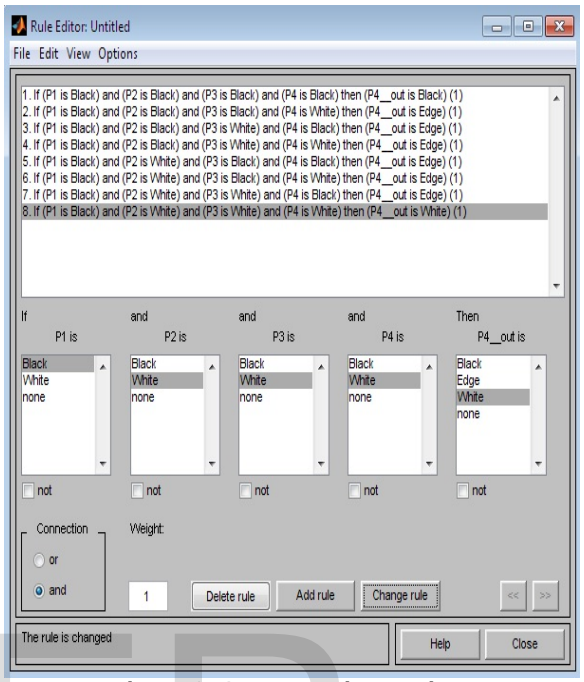


Figure 9: System Rules Design

Step 5: The system output result corresponding to inputs applied can be checked by clicking "view" and then clicking "rules" as shown in Figure 10 and Figure 11.

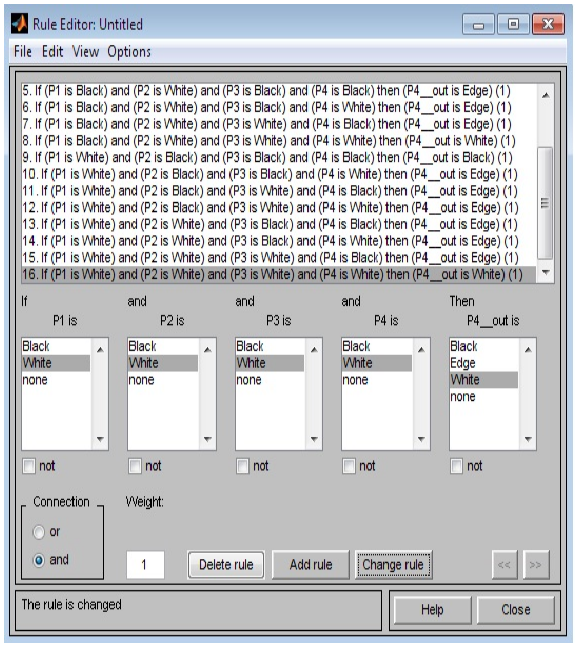


Figure 10: Check the rules

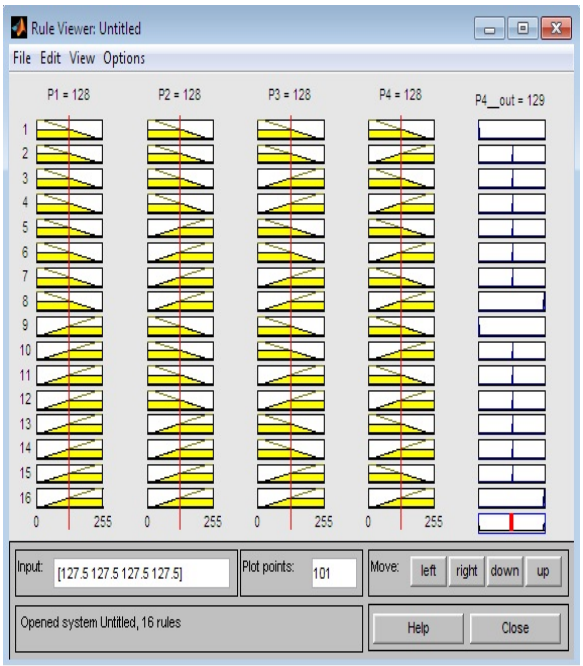


Figure 11: Output check as per rules by different inputs in Rule Viewer.

Step 6: Now save the file by clicking file then export and then by clicking to file of any above fuzzy logic figure as shown in Figure 12.

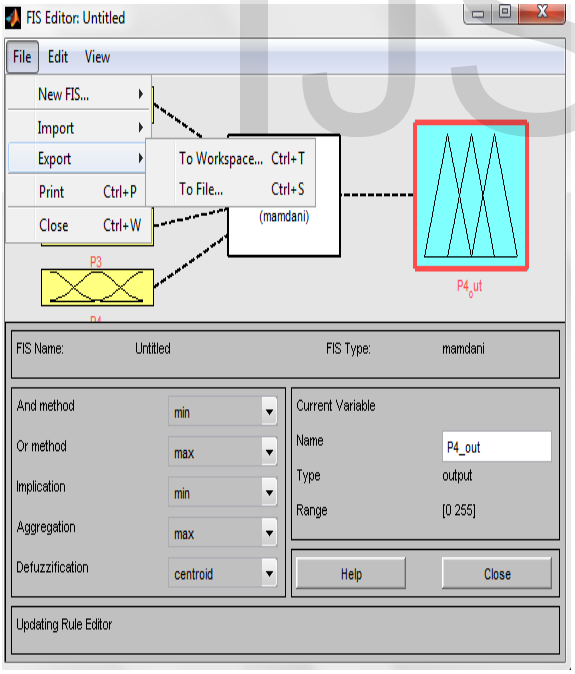


Table I and Table II show the fuzzy sets for I/O variables and fuzzy rule matrix respectively. A rule base of 16 rules is set for the various fuzzy conditions that can occur. Singe output describes whether the output pixel i.e. P_{4_out} belongs to White fuzzy set, Black fuzzy set or Edge fuzzy set. Rules are enlisted in the form of a matrix in Table. II

TABLE I: FUZZY SETS FOR I/O VARIABLES

Name	Range	Membership Function Type
Fuzzy Input No. 1, Pixel P_1		
Black	[0 0 25.5 229.5]	Trapezoidal
White	[25.5 229.5 255 255]	Trapezoidal
Fuzzy Input No. 1, Pixel P_2		
Black	[0 0 25.5 229.5]	Trapezoidal
White	[25.5 229.5 255 255]	Trapezoidal
Fuzzy Input No. 1, Pixel P_3		
Black	[0 0 25.5 229.5]	Trapezoidal
White	[25.5 229.5 255 255]	Trapezoidal
Fuzzy Input No. 1, Pixel P_4		
Black	[0 0 25.5 229.5]	Trapezoidal
White	[25.5 229.5 255 255]	Trapezoidal
Fuzzy Output No. 1, Pixel P_{4_out}		
Black	[0.3 2.7 3.3 5.7]	Trapezoidal
Edge	[130.3 132.7 133.2 134.8]	Trapezoidal
White	[249.3 251.7 252.3 254.7]	Trapezoidal

TABLE II: FUZZY RULE BASE

Rule No.	P_1 Y(i, j)	P_2 Y(i, j+1)	P_3 Y(i+1, j)	P_4 Y(i+1, j+1)	Checked Pixel (P_{4_out})
1	Black	Black	Black	Black	Black
2	Black	Black	Black	White	Edge
3	Black	Black	White	Black	Edge
4	Black	Black	White	White	Edge
5	Black	White	Black	Black	Edge
6	Black	White	Black	White	Edge
7	Black	White	White	Black	Edge
8	Black	White	White	White	White
9	White	Black	Black	Black	Black
10	White	Black	Black	White	Edge
11	White	Black	White	Black	Edge
12	White	Black	White	White	Edge
13	White	White	Black	Black	Edge
14	White	White	Black	White	Edge
15	White	White	White	Black	Edge
16	White	White	White	White	White

V. DEVICE-DEPENDENT COLOR SPACE

We can easily convert RGB data to several common device-dependent color spaces, and vice versa. A device dependent color space is a color space where the color produced depends both the parameters used and on the equipment used for display. These color spaces are: YIQ, YCbCr and HSV.

YIQ Color Space: The National Television Systems Committee (NTSC) defines a color space known as YIQ. This color space is used in televisions. In this format we can separate grayscale information from color data, so the same signal can be used for both color and black and white sets. In the NTSC color space, image data consists of three components: luminance (Y), hue (I), and saturation (Q). The first component, *luminance*, represents grayscale information, while the last two components make up *chrominance (color information)*.

YCbCr Color Space: The YCbCr color space is widely used for digital video. In this format, luminance information is stored as a single component (Y), and chrominance information is stored as two color-difference components (Cb and Cr). Cb represents the difference between the blue component and a reference value. Cr represents the difference between the red component and a reference value.

HSV Color Space: Hue, Saturation, Value or HSV is a color model that describes colors(hue or tint) in terms of their shade (saturation or amount of gray) and their brightness (value or luminance). Hue is expressed as a number from 0 to 360 degrees representing hues of red (starts at 0), yellow (starts at 60), green (starts at 120), cyan (starts at 180), blue (starts at 240), and magenta (starts at 300). Saturation is the amount of gray (0% to 100%) in the color. It can be thought of as the purity of a color. Value (or Brightness) works in conjunction with saturation and describes the brightness or intensity of the color from 0% to 100% [18]. The following figure 13 illustrates the HSV color space. (Source: http://www.mathworks.in/help/images/converting-color-data-between-color-spaces.html).

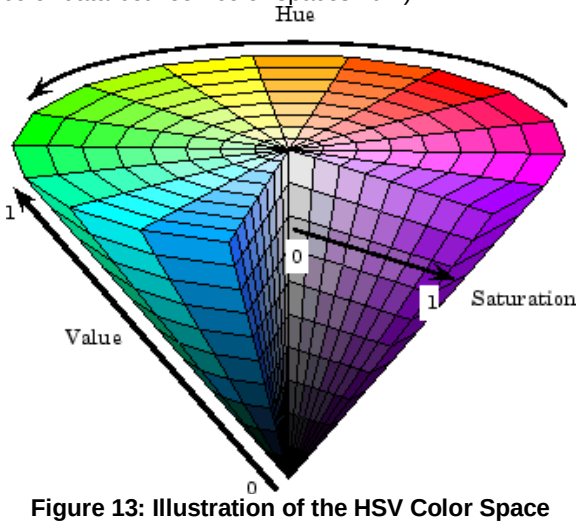


Figure 13: Illustration of the HSV Color Space

VI. EXPERIMENTS

The implementation was carried out consisdering that the input image and the output image are both 8-bit quantized so that their gray levels are always between 0 and 255. The condition of each pixel is decided by using a floating mask window of 2X2 size which will scanning all the grays. In rule number 1 and 9, the checked pixel is black, in rule number 8 and 16 the checked pixel is white and in rule numbers from 2 to 7 and 10 to 15, the checked pixel corresponds to edge. In fuzzification the input grays are ranged from 0 to 255 gray intensity and in the defuzziaction again the vlaues range from 0 to 255 gray intensity levels. From the tested images in this we found that best results are achived at the range black from 0 to 25 gray valus and from 230 to 255 meaning that the weight is white.

The fuzzy conditions help us to test the relative values of pixels which can be present in case of presence on an edge. So the relative pixel values are instrumental in extracting all the edges associated to an image. The image is said to have an edge if the intensity variation in between the adjacent pixels is large. Edge Detection – Locating areas with strong intensity contrasts. The mask is slid over an area of the input image, changes that P_4 pixel's value and then shifts one pixel to the right and continues to the right until it reaches the end of a row. It then starts at the beginning of the next row and process continues till the whole image is scanned.

When this mask is made to slid over the image, the output is generated by the fuzzy inference system based upon the sixteen rules and the value of the pixels P_1, P_2, P_3 and P_4 say for example rule number 4 states that when the values of pixels P_1 and P_2 are black and that of pixels P_3 and P_4 are white then the output P_{4_out} is an edge. The FIS system was tested with different type of face images and its performance was compared with the standard sobel edge detection algorithm and canny edge detection algorithm.

VII. CODES

1. Conversion from RGB color space to YIG color space and vice-versa:
   X = imread ('D:\88.jpg', 'jpg');
   % Address of the image along-with its format.
   Y = rgb2ntsc (X);
   % Convert the image from RGB format into YIQ format.
   Z = ntsc2rgb (X);
   % Convert the image from YIQ format back into RGB format.

2. Conversion from RGB color space to YcbCr color space, then changing the values of Y, Cb & Cr components of the image and finally converting it back to RGB color space:
   X = imread ('D:\88.jpg', 'jpg');
   % Address of the image along-with its format.
   Y = rgb2ycbcr (X);

```
% Convert the image from RGB format into YCbCr format.
imageY = Y (:,:,1);
% Separating the Y Component.
imageCb = Y (:,:,2);
% Separating the Cb Component.
imageCr = Y (:,:,3);
% Separating the Cr Component.
imageY = imageY*2;
% Changing the Y Component.
imageCb = imageCb*2;
% Changing the Cb Component.
imageCr = imageCr*2;
% Changing the Cr Component.
YCbCr = cat (3, imageY, imageCb, imageCr);
% Obtaining image after combining new Y, Cb & Cr Components.
Z = ycbcr2rgb (YCbCr)
% Convert the new changed image from YCbCr format back into RGB format.
```

3. Conversion from RGB color space to HSV color space, then changing the values of H, S & V components of the image and finally converting it back to RGB color space:

```
X = imread ('D:\88.jpg', 'jpg');
% Address of the image along-with its format.
Y = rgb2hsv (X);
% Convert the image from RGB format into HSV format.
hChannel = Y (:,:,1);
% Separating the H Component.
sChannel = Y (:,:,2);
% Separating the S Component.
vChannel = Y (:,:,3);
% Separating the V Component.
hChannel = hChannel *2;
% Changing the H Component.
sChannel = sChannel *2;
% Changing the S Component.
vChannel = vChannel *2;
% Changing the V Component.
HSV = cat (3, hChannel, sChannel, vChannel);
% Obtaining image after combining new H, S & V Components.
Z = hsv2rgb (HSV);
% Convert the new changed image from HSV format back into RGB format.
```

VIII. RESULTS

The proposed system was tested with a standard image of "Lena" compressed to size 256X256 is used in mostly functions demonstrating Matlab results. The image in RGB format is converted to YIQ, YCbCr format and HSV format. The results are obtained from YIQ format directly. In case of YCbCr and HSV,its individual components like H, S & V in HSV and Y, Cb & Cr in YCbCr are changed firstly by generating each individual component separately, then changing it say by multiplying each individual component by a factor of two and then at last converting it back to RGB format . Finally the performance of all the device-dependent color space including that of RGB is being compared to that of the Sobel edge detection algorithm and Canny edge detection algorithm in MATLAB environment. The firing order associated with each fuzzy rule were tuned to obtain good results while extracting edges of the image where we used this image as comparative model for the classical Sobel operator, Canny operator and the FIS method. Also while changing the various components of the HSV and YCbCr color space, they are simply multiplied by a factor of two. Maximum care has been taken to enhance the quality of image. Results of images of different color space are shown below.

Figure 14.a shows the original image in RGB format, figure 14.b shows the edges using the classical Sobel operator, figure 14.c shows the edges using the Canny edge detection operator and finally figure 14.d shows the edges of our designed Fuzzy Inference System.

Figure 15.a shows the image in YIQ, converted from RGB format, figure 15.b shows the edges using the classical Sobel operator, figure 15.c shows the edges using the Canny edge detection operator and finally figure 15.d shows the edges of our designed Fuzzy Inference System.

Figure 16.a shows the image in RGB format, this image is obtained first by converting RGB color space to YCbCr and then from YCbCr format, we obtain each component i.e. Y, Cb & Cr separately, then we multiply the Y component with 1.5 and keep the Cb & Cr components unchanged. Next we combine all the components to obtain the changed YCbCr color space image. Finally this image is then converted back to RGB color space for the detection of edges. Figure 16.b shows the edges using the classical Sobel operator, figure 16.c shows the edges using the Canny edge detection operator and finally figure 16.d shows the edges of our designed Fuzzy Inference System.

Figure 17.a shows the image in RGB format, this image is obtained first by converting RGB color space to HSV and then from HSV format, we obtain each component i.e. H, S & V separately, then we multiply the S & V components with 1.5 and keep the H component unchanged. Next we combine all the components to obtain the changed HSV color space image. Finally this image is then converted back to RGB color space for the detection of edges. Figure 17.b shows the edges using the classical Sobel operator, figure 17.c shows the edges using the Canny edge detection operator and finally figure 17.d shows the edges of our designed Fuzzy Inference System.

(ALL RESULTS SHOWN ON NEXT PAGE)


Figure 14.a (Original RGB Image)


Figure 14.b (Sobel Edge Operated)


Figure 14.c (Canny Edge Operated)


Figure 14.d (Edges using FIS)

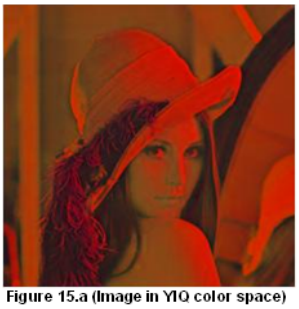


Figure 15.a (Image in YIQ color space)


Figure 15.b (Sobel Edge Operated)

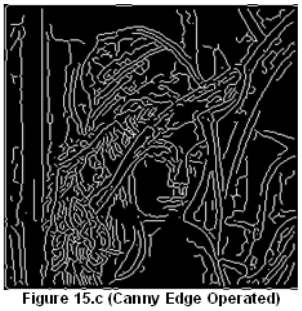


Figure 15.c (Canny Edge Operated)


Figure 15.d (Edges using FIS)


Figure 16.a (YCbCr image – Changed)


Figure 16.b (Sobel Edge Operated)


Figure 16.c (Canny Edge Operated)

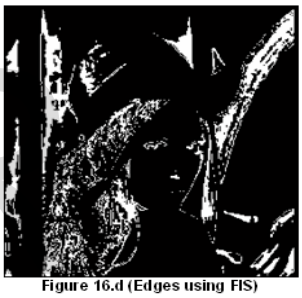


Figure 16.d (Edges using FIS)


Figure 17.a (HSV image – Changed)


Figure 17.b (Sobel Edge Operated)


Figure 17.c (Canny Edge Operated)


Figure 17.d (Edges using FIS)

IX. CONCLUSION

Fuzzy processing is desirable in computer vision because of the uncertainties that exist in many aspects of image processing. These uncertainties include additive and non-additive noise in low-level image processing, imprecision in the assumptions underlying the algorithms, and ambiguities in interpretation during high-level image processing. For example, for computational convenience, the common process of edge detection usually models edges as intensity ridges. Nevertheless, in practice this assumption only holds approximately, leading to some of the deficiencies of these algorithms.

In this paper, emphasis has been to develop a very simple and small but a very efficient, fuzzy rule based edge detection algorithm to abridge the concepts of artificial intelligence and digital image processing. We have used the smallest possible mask of size 2×2. The algorithm has been developed in MATLAB environment. Comparisons were made with the various other edge detection algorithms that have already been developed like sobel edge detection algorithm and canny edge detection algorithm using digital images taken from different device-dependent color spaces. Displayed results have shown the accuracy of the edge detection using the fuzzy rule based algorithm over the other

algorithms specially in RGB color space and HSV color space.

The fuzzy rule based algorithm has been successful in obtaining the edges that are present in an image after its implementation and execution. Four sample outputs, all human faces, each representing one color space, have been shown to make the readers understand the accuracy of the algorithm. Thus developed algorithm exhibits tremendous scope of application in various areas of digital image processing.

X. FUTURE SCOPE

a) In our technique, the image is first to be converted into gray image. This limitation can be eliminated and algorithm can be applied directly to color images but the detection would then become significantly more complex.

b) Also the size of mask that we had used is 2x2. We can also use mask of size 3x3, 4x4 or even of 5X5. Also accordingly to the increase in the size of mask more input membership functions may be added. The rules will also have to be changed as per the mask size.

c) In our paper, in order to enhance the quality of image in YCbCr and HSV color space, their individual components have been changed with some specific numeric number. Reader can try changing their components in order to achieve the best possible results.

REFERENCES

[1] M. Kowalczyk, P. Koza, P. Kupidura and J. Marciniak, “Applications of mathematical Morphology Operations for Simplification and Improvement of Corelation of Images”, The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. Vol. XXXVII. Part B5. Beijing 2008, pp. 153-158.

[2] Ehsan Nadernejad, Sara Sharifzadeh and Hamid Hassanpour, “Edge Detection Techniques: Evaluations and Comparisons”, Applied Mathematical Sciences, Vol. 2, 2008, no. 31, pp. 1507 – 1520.

[3] F. Bergholm. “Edge focusing,” in Proc. 8th Int. Conf. Pattern Recognition, Paris, France, 1986, pp. 597- 600.

[4] J. Matthews. “An introduction to edge detection: The sobel edge detector,” Available at http://www.generation5. org /content/2002/im01.asp, 2002.

[5] Canny, J.,” A Computational Approach to Edge Detection”, IEEE Transactions on pattern Analysis and Machine Intelligence, Vol. PAMI-8, No. 6, November 1986, pp. 679-698.

[6] Nelishia Pillay, “Artificial Intelligence in Computer Science Teaching and Research”, Conference’04, South Africa, Month 1–2, 2004, pp. 1-5.

[7] Bih, Joseph, “Paradigm Shift – An Introduction to Fuzzy Logic”, IEEE Potentials, January/February 2006, pp. 6-10, 21.

[8] Zadeh, L. A. et al. 1996 Fuzzy Sets, Fuzzy Logic, Fuzzy Systems, World Scientific Press, ISBN 981-02-2421-4.

[9] Gao, Y. and Leung, K.H. “Face Recognition Using Line Edge Map (June 2002),” IEEE Transactions On Pattern Analysis & Machine Intelligence, Vol. 24, Issue No. 6, pp. 764-779.

[10] Li, Y., Lin, B., and Zhou, B. (November 2009), “The Application of Image Edge Detection by Using Fuzzy Technique”, Electronic Imaging and *Multimedia Technologies*, pp. 398-405.

[11] Ryszard, S.C. (2007), “Image Feature Extraction Techniques and Their Applications for CBIR and Biometrics Systems”, International Journal of Biology & Biomedical Engineering, Issue No. 1, Vol. 1, pp. 6-16.

[12] Sarode, M.V., Ladhake, S.A. and Deshmukh, P.R. (2008), “Fuzzy system for color image enhancement”, World Academy of Science, Engineering and Technology, pp. 311-316.

[13] Mathur, S., and Ahlawat, A. (June-July 2008), “Application of Fuzzy Logic on Image Detection”, Intelligent Information and Engineering Systems, INFOS 2008, Bulgaria, pp. 24-28.

[14] Barkhoda, W., Akhlaqian, F. and Shahryari, O.K. (2009), “Fuzzy Edge Detection Based on Pixel's Gradient and Standard Deviation Values”, International Multi-conference on Computer Science and Information Technology, Vol. 4, pp. 7–10.

[15] Abdallah, A., Alshennawy and Ayman, A. (July 2009), “Edge Detection in Digital Images Using Fuzzy Logic Technique”, International Journal of Electrical and Computer Engineering, Vol. 4, pp. 178 - 186.

[16] Aborisade, D.O. (April 2011), “Novel Fuzzy logic Based Edge Detection Technique”, International Journal of Advanced Science and Technology, Vol. 29, pp. 75-82.

[17] Sandhu, M.S., Mutneja, V. and Nishi, “Image Edge Detection by Using Rule Based Fuzzy Classifier”, International Journal of Computer Science and Information Technology, Vol. 2(5), 2011, pp. 2434-2439.

[18] Nishad PM*1 and Dr. R.Manicka Chezian2, “VARIOUS COLOUR SPACES AND COLOUR SPACE CONVERSION ALGORITHMS”, Journal of Global Research in Computer Science, Volume 4, No. 1, January 2013, pp. 44-48.