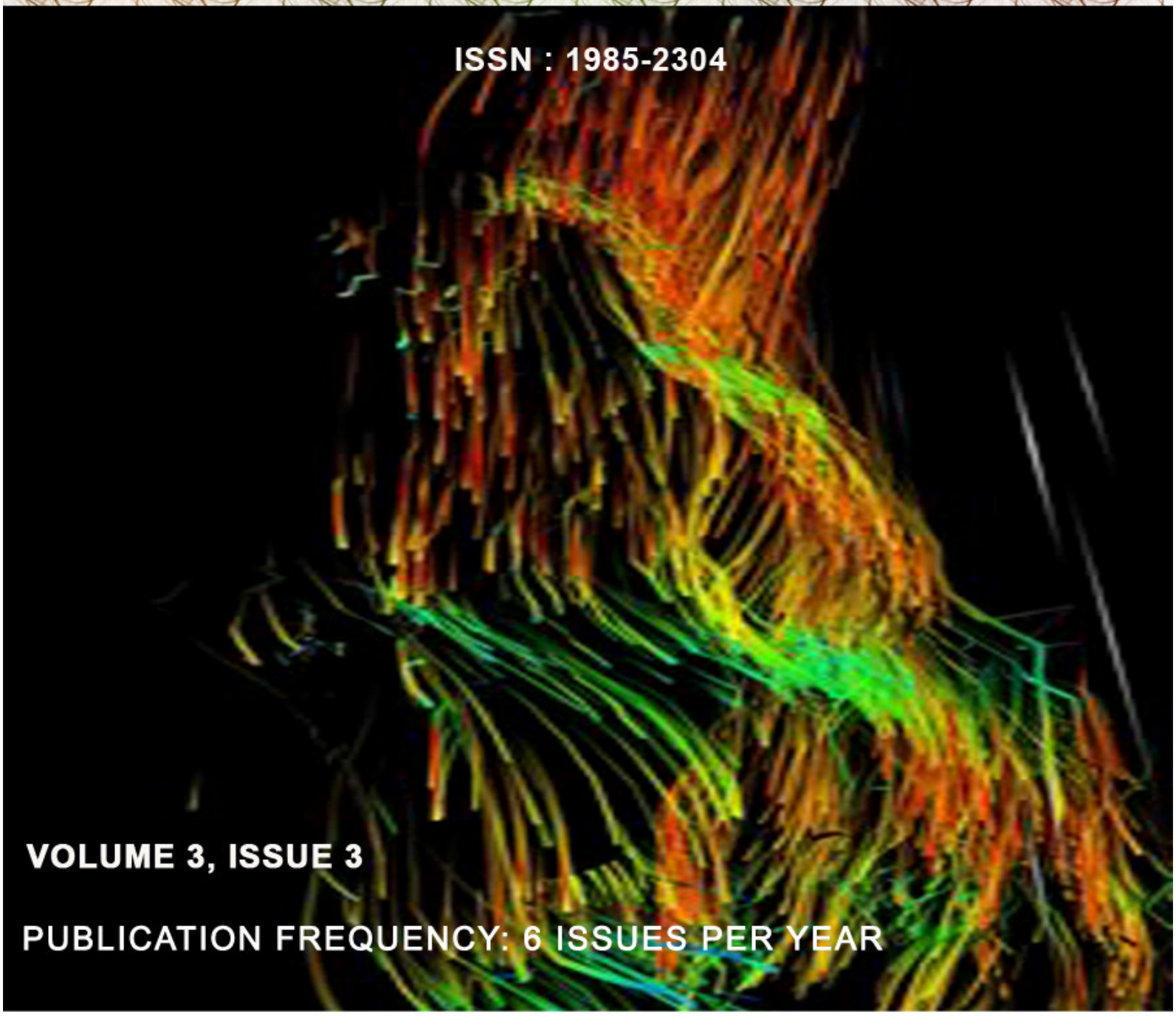
International Journal of Image Processing (IJIP)

ISSN : 1985-2304



VOLUME 3, ISSUE 3

PUBLICATION FREQUENCY: 6 ISSUES PER YEAR

Editor in Chief Professor Hu, Yu-Chen

International Journal of Image Processing (IJIP)

Book: 2009 Volume 3, Issue 3
Publishing Date: 31 - 06 - 2009
Proceedings
ISSN (Online): 1985 -2304

IJIP Journal is a part of CSC Publishers
http://www.cscjournals.org

Published in Malaysia

Typesetting: Camera-ready by author, data conversation by CSC Publishing Services – CSC Journals, Malaysia

CSC Publishers

Table of Contents

Volume 3, Issue 3, June 2009.

Pages

92 – 104	Optical Character Recognition System for Urdu (Naskh Font)Using Pattern Matching Technique **Tabassam Nawaz, Syed Ammar Hassan Shah Naqvi, Habib ur Rehman , Anoshia Faiz**.
105 - 119	Filtering Corrupted Image and Edge Detection in Restored Grayscale Image Using Derivative Filters **Chandra Sekhar Panda, Srikanta Patnaik.**
120 - 130	Independent Component Analysis of Edge Information for Face Recognition **Kailash Jagannath Karande, Sanjay N Talbar.**

Optical Character Recognition System for Urdu (Naskh Font) Using Pattern Matching Technique

Tabassam Nawaz drtnawaz@uettaxila.edu.pk
Faculty of Software Engineering,
University of Engineering & Technology
Taxila, Pakistan

Syed Ammar Hassan Shah Naqvi ammar_hassan13@yahoo.com
Department of Software Engineering
University of Engineering & Technology
Taxila, Pakistan

Habib ur Rehman habieb_rehman@yahoo.com
Department of Software Engineering
University of Engineering & Technology
Taxila, Pakistan

Anoshia Faiz an_oshia@yahoo.com
Department of Software Engineering
University of Engineering & Technology
Taxila, Pakistan

Abstract

The offline optical character recognition (OCR) for different languages has been developed over the recent years. Since 1965, the US postal service has been using this system for automating their services. The range of the applications under this area is increasing day by day, due to its utility in almost major areas of government as well as private sector. This technique has been very useful in making paper free environment in many major organizations as far as the backup of their previous file record is concerned. Our this system has been proposed for the Offline Character Recognition for Isolated Characters of Urdu language, as Urdu language forms words by combining Isolated Characters. Urdu is a cursive language, having connected characters making words. The major area of utility for Urdu OCR will be digitizing of a lot of literature related material already stocked in libraries. Urdu language is famous and spoken in more than 3 big countries including Pakistan, India and Bangladesh. A lot of work has been done in Urdu poetry and literature up to the recent century. Creation of OCR for Urdu language will make an important role in converting all those work from physical libraries to electronic libraries. Most of the stuff already placed on internet is in the form of images having text, which took a lot of space to transfer and even read online. So the need of an Urdu OCR is a must. The system is of training system type. It consists of the image preprocessing, line and character segmentation, creation of xml file for training purpose. While Recognition system includes taking xml file, the image to be recognized, segment it and creation of chain codes for character images and matching with already stored in xml file.

The system has been implemented and it has 89% recognition accuracy with a 15 char/sec recognition rate.

Keywords: Pattern matching, chain code creation, morphology, segmentation, training system, recognition system, digital image processing.

1. INTRODUCTION

An Optical Character Recognition System is software engineered to convert hand-written or typewritten text (usually scanned) documents into machine editable text formats.

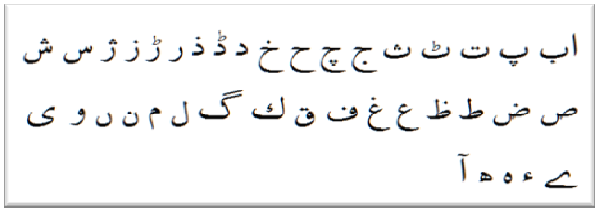


FIGURE 1: Urdu Characters set

This paper describes a training based offline [11] optical character recognition system for a Naskh font of Urdu language. The main idea behind this recognition is matching the pixel values of the samples already stored with pixel values of those character-images to be recognized. The major difficulty while the recognition of Urdu language is its cursive nature, i.e. the characters joined to create new words. The research in this has been greatly increased during last decade, as the applications of this area are increasing. It has improved Human Computer Interaction; other examples include paperless environment, online newspapers, old literature online availability, paper checking, automating official tasks, reading bank receipts, postal addresses and data entry forms. Many people are now a day working on Urdu OCR research. As Standard Urdu has approximately the twentieth largest population of native speakers, among all languages. Due to technical issues induced by the cursive nature of Urdu language, its OCR has not been developed completely. If Urdu OCR system is available, it will be very useful and will have great commercial value. The overall flow of our OCR is shown in Fig.2. Section II describes the Urdu language specifications. Some related work done for Urdu OCR is described very briefly in Section III. Section IV and V describes the Training and Recognition Systems respectively. Section VI describes the process of creating Unicode file [3] from already recognized characters. All the algorithms used in this paper are described in Section VII. Finally, this paper is concluded in Section VIII.

2. URDU LANGUAGE SPECIFICATIONS

Urdu language is the old language of Indo-Pak Sub-continent, now it is the national language of Pakistan. This language is a combination of characteristics of Arabic, Farsi and Sanskrit languages, as it is the language of troops. The Urdu language has the characteristics of all these languages mentioned above, all the characters in this language are picked from these languages. Urdu language is a more cursive and complex language than Arabic and Farsi language as it contains the connected characters to make words. This cursive nature [9] makes it very difficult to be recognized through usual Character Recognition Methods.

Urdu character set consists of 40 characters. The characters contain single loop, double loops and incomplete loops. Dots and diacritics (i.e., Telda (~) and Nuktay (◌)) are also included in the character set. Dots include single, double and triple dots. The recognition of Urdu language is

very difficult due to the different multiple shapes of a single character. In Urdu language, every letter is of minimum 2 shapes and maximum 4 shapes. These shapes are based on their occurrence in the given word. The locations in the Urdu word are isolated, initial, medium and last. Here all isolated shapes of the characters are mentioned in Fig.1. So the Urdu language character recognition is very difficult still. Another characteristic of this language is; if we read or write something is from right to left.

3. RELATED WORK

One of the oldest techniques for pattern recognition is used for character recognition, but through all days, more focus was on Latin, Chinese [24] and Japanese [25] languages, though connected. First, we applied Hilditch's method, which consists of removing the pixels that lie on the edge of the binary image until only one-pixel-wide line remains. This is followed by some conditions suggested by Al-Emami to reduce the junction points to one junction point [23]. The matching procedure is executed based on an image based matching algorithm. From a practical viewpoint however, the matching time must be reduced as much as possible through the classification techniques [3].

In our studies we analyzed the shape and visual properties of Urdu characters and define a set of features which can distinguish one character to another. In Urdu Qaeda system, the online character recognition technique is used for isolated characters which is some how based on the strokes by the user runtime [17].

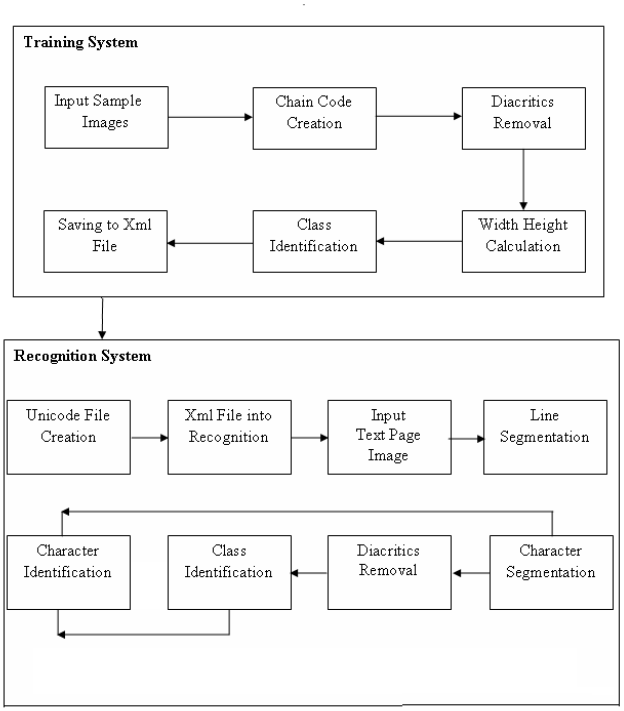


FIGURE 2: Block Diagram of System

4. SYSTEM

4.1 Training System

- *Requirements*

As we will first train our system for a specific font and then matching algorithm will match both the sampled image information and the actual image (input image) information. So the system needs to know the font and size of the text to be recognized. Thus we have to select a specific font and specific size in our system. In our case, the following demonstration of training and matching is only for NASKH font and it can be extended to any font of any size. Sample font size to demonstrate is taken to be 36. The images used for training should be noise-free and gray scale. We will first convert the gray scaled image into binary image as shown in Fig.3, and remove extra noise from the image. Now we can apply morphological transforms [12] easily on these images to get required results.


Original Image


Binary Image

FIGURE 3: Binarization of input image

- *How to Take Character's Sample*

A gray scaled image comprising of the whole character set of Urdu language (Naskh font) is taken as sample and individual characters are separated out. All these individually separated characters are created in such a way that from each side one totally white column or row is left for the ease of chain code [2] creation. As stated earlier, only isolated characters will be recognized so the samples are taken in such a way. This technique can be applied to any font having any size.

- *How to Create Chain code of characters*

The individual character images are then passed into the training software. Each character image is then scanned from top to bottom and then to the next column, a chain code [18] of each column of the image is generated. The chain code based on the sum of consecutive one's or zeros, as the image is only a binary image. According to our assumption, Zero is assumed to be the first entry of the column. If this Zero appears zero times, then we will place 00 in the string, then for example 1's appear for 23 times, we will add 23 in the string. If we continue scanning, now its turn of zero, if zero appears for 12 times, we will add 12 to the string. Now, if the column is finished, we will add @ to the string to make sure that column is ended (as shown in Fig.4) the same process will be repeated for next column and so on. It should be noticed that when the number is less than 10 e.g., 3 zeros then we write "03"; we add this because maximum no. of pixels can't exceed 99. As these columns are of only individual characters. Thus by calculating all columns we get a chain code (from the final string) of set of the character image. This chain code is generated by calculating alternating on and off pixels as shown in the Fig.4.

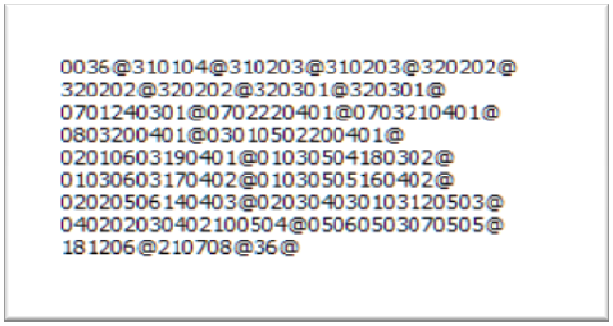


FIGURE 4: Calculated String

Now the chain code is available, the task is to store this chain code, we can use different options for storing this string including database, collection objects of C# and xml file. We have chosen the xml file for storing this file, as it will take least space and reading/writing into the file will be easy. The Fig.5 explains the syntax of our xml file. This xml file is described in next section.

FIGURE 5: Classxml.xml

- *How to create classes*
- *Xml file*

Second step during training phase is creation of xml file. Xml file contains all the 21 classes [3] of Urdu alphabets as parent nodes or elements. Each class contains character set belonging to that particular class, where every character makes a single child node of parent class node. Each child node has three attributes. One, the name of the character and the other, chain code of that character, calculated from its image earlier. Unicode of the character is saved in xml as third attribute of the child node, which will be assigned to the identified character at the end of the matching procedure.

- *Classification Criteria*

Classification of characters is the key technique in our pattern matching method of optical character recognition. Urdu script has a large character set, consisting of 40 characters. Pattern matching technique is of no use if system has to traverse all the 40 characters and match their chain codes. This special matching technique [14] is made very efficient by optimization at every level, including logical methods and programming techniques. This classification is done on the bases of height and width of the character without diacritics and dots. Urdu script consists of total 21 shapes, occupying specific dimensions. These 21 shapes are declared as the classes in which our system divides the characters for recognition. All the recognize-able characters are input to the training module to train our OCR system for a specific font and size. Training system can train

the OCR for any font and size character set, as mentioned earlier. System will store their dimensions as individual classes of characters. Fig.6 describes all the 21 classes.

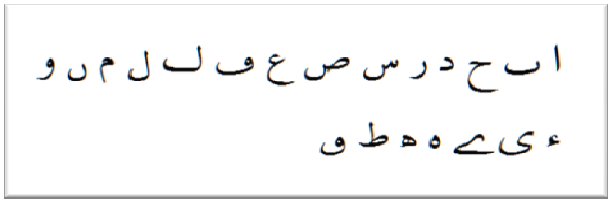


FIGURE 6: Character's Classification

- *How to remove diacritics*

The character's dots and diacritics are removed using morphology techniques [12] of binary images. The dots are removed using pepper noise [6] removal process and diacritics like ~ are removed using thinning process [5]. Special filters are designed for removing diacritics. In Urdu script characters contain a variation of diacritics, like single Nukta, double Nukta or triple Nukta and "Chota Tuay" as in Fig.7.

FIGURE 7: Shows Chota Tuay, Double, triple and Double Dots respectively from left.

- *Filters*

Different special filters are designed particularly to remove diacritics from the isolated character images. When the shape of the character is obtained after removing diacritics, we calculated the dimensions (height and width) of each character and place it into particular class. Fig.8, Fig.9a, Fig.9b and Fig.9c describe the filter for removing ttuay' and single dot, double dot, triple dot respectively.

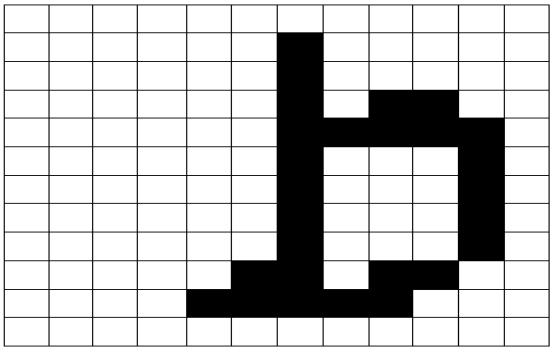


FIGURE 8: Ttuay Filter

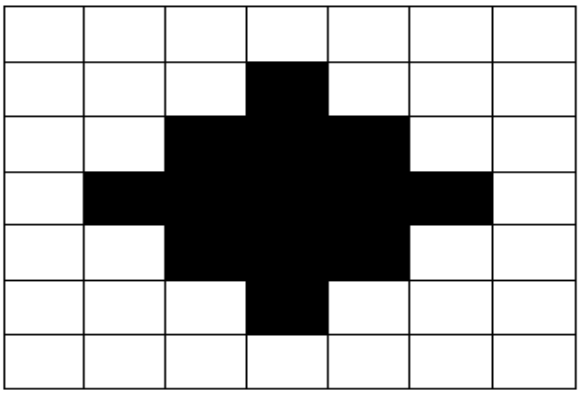


FIGURE 9:. Single dot Filter

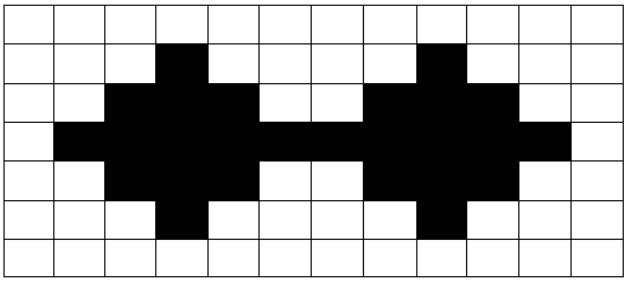


FIGURE 9a: Double dots Filter

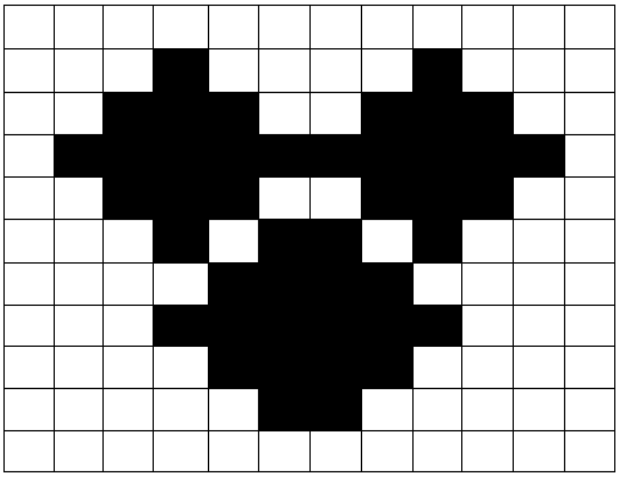


FIGURE 9b: Triple dot Filter

- *Chain code*

After creating the xml for 21 classes, measured the chain code for each class characters. All the characters of a class are added to the xml file as children to the parent classes. All the character set classes are shown in the Fig.1. This xml file will act as an input for the recognition system.

4.2 Recognition System

- *Segmentation*

When an image containing text is given as the input to the character recognition system, system performs some preprocessing steps on it; including converting grayscale image into binary image and enhancing the image by removing pepper noise. Binary image is need for applying filters and

calculating chain code. Pepper noise must be removed because it will lead the system to erroneous classification and character recognition. Error margin (described later in this section) methods are very sensitive to extra noise in image as noise can be located at random locations disturbing the chain code and specially dimensions. Remember that "character image variation" is always symmetric, not random. System then divides the image into segments. As at this stage this paper is describing the recognition of isolated characters, thus segmentation is required at two levels instead of three levels as required for connected characters in recursive script i.e. Urdu script.

Two levels of segmentation are:

- *Line segmentation*

The image is first divided into different lines of text by checking for whole row of white pixels consecutively from right to left. Each separated line will be saved separately in the recognition system. Fig.10 shows an image after the lines have been segmented from an image.

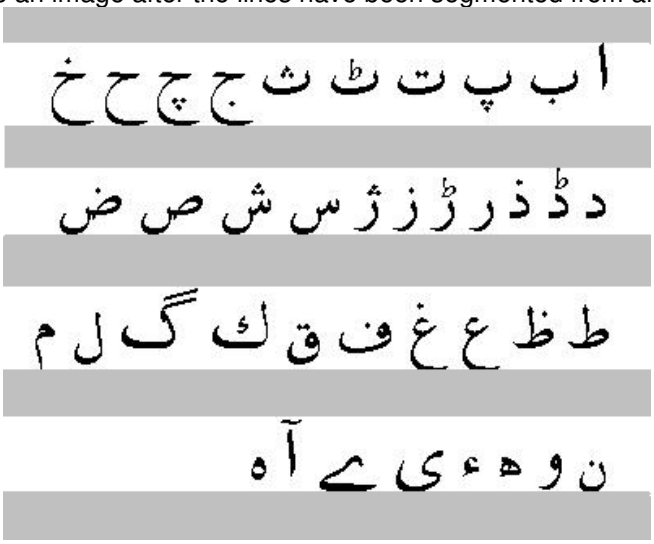


FIGURE 10: Line Segmentation Results

- *Isolated character segmentation*

The already separated text line images are to be processed now, for isolated character recognition, by converting them into individual characters. During processing, we check for full white pixels column at the starting and ending point of the isolated character. Start of the character is identified when any black pixel is scanned in the column. Scan continues until another white pixel column is identified. Image in-between the white columns, is saved, and starting and ending white column of pixels is preserved in the image. These results into different small images of individual characters are shown in Fig.11 which will be processed separately. Every separate image of isolated character is now completely void on four sides. Leaving one pixel margin on each side of the image is because of the assumption that in chain code of each column of the image will start with white pixels count.

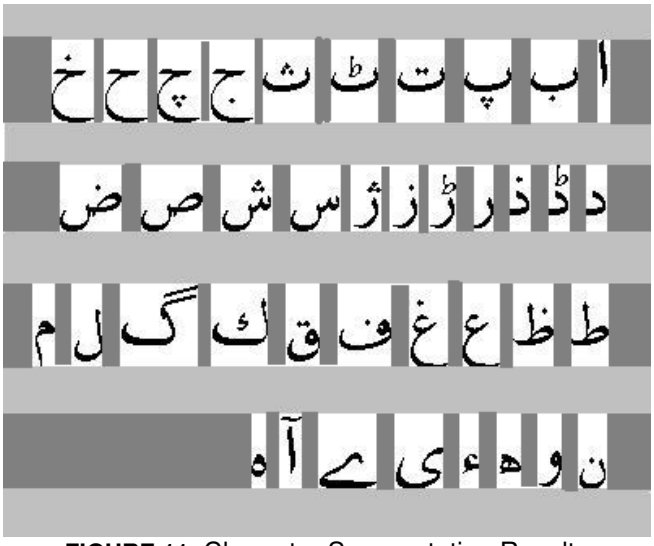


FIGURE 11: Character Segmentation Results

- *How to Identify Class of a Character*

After segmentation of whole document image into small images of isolated characters, next step is to identify class of each character. Following Steps will describe whole recognition steps;

- *Diacritics Removal*

To obtain an image of recognizable class, dots (Nuka's) and diacritics (ttuay) are removed from the above, below or inside the character image using especially designed filters (as shown earlier in Figs[8,9a,9b,9c]) specific to them. This dot-less image is stored as another copy of the character image. Now, the height and width of this new character image are measured with only considering black pixel, as was done before while taking samples and matched with the height and width of different classes in the classxml.xml file.
When a class gets matched, then the image's string will be calculated again and matched with the already calculated strings of the matched class characters each. This classification helps in providing more efficient approach in pattern matching.

- *Chain code calculation*

At this stage, we knew the class to which the character belongs; now the original image will be used for recognition. And this original image is scanned from top to bottom for each column to obtain its' chain code. We will call this chain code, the "calculated chain code" and the one that is saved in characterset.xml file called "sampled chain code". The calculated chain code is calculated using the scanning the input image and generating the calculated string of on and off pixels as described earlier in section 3.1.

- *How to Recognize Character*

After determining the class of the character, we have to just match the calculated chain code with only some character's sampled chain codes. This makes the process of identification more reliable and efficient. The calculated chain code is matched column by column with every character's sampled chain code in the identified class. This process is repeated until all the columns are matched with some extent of error margin. As we have already set the error margin, character is identified up to the margin because different images can have little different properties, so exact pattern matching cannot be so efficient, to identify the exact character.

  - *How to check Error Margin*

As any image can be error prone, we must satisfy the "chain code matching method" about the correct and exact character to recognize. In any case error count (mismatches) for each column is calculated, while matching total number of columns that is the width of the image. If width varies, equalize sample image width with the calculated image width. To equalize the width just add some characters like '.' in the shorter image as new columns. Now both images contain equal number of columns to match with each other. Once we are done with width it's time to check whether both columns are of equal height or not. To equalize the heights of sample column and calculated column again add characters like '.' in shorter image as a new row in affected column. Now both images contain equal number of columns and in each column equal number of rows. Match column by column the chain code and calculate the mismatches as error count. If error count increases the already set value, get to the next image in the identified class only. Character is identified when error is less than error margin.

5. ALGORITHMS USED

Chain Code Calculation

- Start at right top of the image.

- Scan from top to bottom of the first column of segmented image.

    - Assume that first pixels scanned are "off", so count number of "off" pixels and add to chain code.
    - Continue scan, if pixel value changes from "off" to "on", start counting "on" pixels and add to chain code.
    - Continue scan, counting "on" and "off" pixels till bottom of the image is reached and add to chain code.
    - At the end of the first column code add "@" as the end of the column.
    - Continue scanning the next columns to the width of the segmented image. Chain codes of all columns are calculated and saved as a single chain code in the xml file, in the corresponding class.

Segmentation

Line Segmentation:

- Load input image.
- Start scan from right top of the original image.
- Scan up to the image width, on the same Y component.
- Scan the first row to check any "on" pixels. If no "on" pixel is found go to next row.
- Continue scan until a row containing "on" pixels come across. Save the row before this row as the top edge of the image text line.
- Continue scan until another row with all "off" pixels is found. Set this row as the bottom edge of the image text line.
- Save image as first text line and continue scan to find next line.
- Continue till the bottom of the image is reached. Save all text line images to be input to character segmentation

Character Segmentation:

- Input text line image.
- Start scan from right top of the text line image.
- Scan up to the image height, on the same X component.

- Scan the first column to check any “on” pixels. If no “on” pixel is found go to next column.
- Continue scan until a column containing “on” pixels come across. Save the column before this column as the right hand edge of the isolated character image.
- Continue scan until another column with all “off” pixels is found. Set this column as the left hand edge of the isolated character image.
- Save image as first character and continue scan to find next character.
- Continue till the left hand edge of the image text line is reached. Save all character images.

Classification

- Load the segmented character image.
- Apply filters to remove diacritics, and get the shape of the corresponding class.
- Measure the dimensions of the shape i.e. the width and height of the character image without diacritics, by scanning the image from top right to left bottom pixel by pixel.
- Open xml file and traverse all the class nodes. Match “width” and “height” attribute values with the calculated values.
- When a dimension match occurs according to some already set error margin (in this case error margin is 1, i.e. if width or height is one less or on greater than the sample values, match occurs.) set that class as the key class for that character.

Character Matching

- Restore the “original segmented character image” with diacritics and dots. Calculate its chain code according to the algorithm described earlier in this section.
- Open xml file. Traverse all child nodes in the identified class and match “code” attribute values with the calculated chain code.
- When a chain code match occurs according to Error Margin, described in next algorithm (Testing), set the “name” attribute of that node as the name of the character. And set the “Unicode” attribute of that node as the Unicode of the input character.

Testing

- Get the sampled chain codes in the identified class from xml file and the calculated chain code of the classified character image.
- Start matching the calculated chain code with each of the sampled chain codes.
- Chain code of each column is compared.
- In each column every single code is compared with actual value (sampled value). In each column every single mismatch is counted as “error count”. If error count exceeds the Error Margin, set according to font and size, disqualify the node to be the exact match.
- Continue with the next node chain code and check the Error Margin. Continue until a character is confirmed. Confirmation occurs when “total error count” of the character is less than the “Error Margin Limit”.

Unicode File Creation

- Characters are recognized through the above process.
- Xml file includes all the information about the recognized character.

- This recognized character is written into a Unicode file, according to the information written in the xml file.

- The Unicode file is written from right to left.

6. RESULTS

Unicode is simply a character encoding system. It has nothing to do with how these characters finally get displayed on computer screen. Thus it cannot be Naskh based only, its only type style issues relating to rendering which is handled by operating system and application not by encoding scheme [4]. After getting the identification of exact character, we create a new Unicode file that will include all the characters identified. All the Unicode characters are according to the Unicode Consortium [3]. We will place the Unicode of those characters in the same position from right to left as in the Urdu text image file. The output Unicode file is now editable and searchable. This file is written from right to left as Urdu language is right to left.

7. CONCLUSION

This training system has recognized different printed Urdu text image files, which include isolated characters. The training system has been tested for different sizes of fonts, the results are quite impressive. Our system has achieved an accuracy of 89% for the isolated characters with accuracy of a 15 char/sec recognition rate. Different sizes of filters are used for different sizes, and this system has proven good for multi font sized characters. The system has been tested for different images and efficient results are found. Fig12.a shows the input image file, after being processed by our OCR, the screen shot of our output text file is shown in Fig.12b.

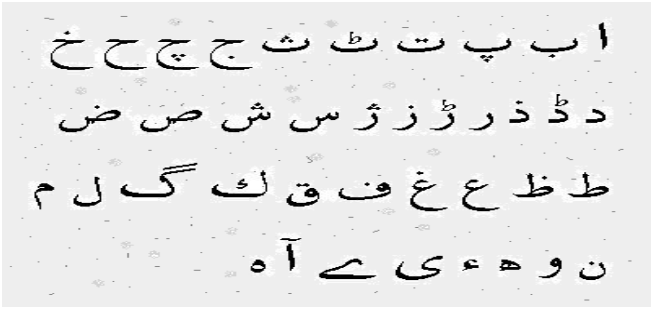


FIGURE 12a: Input image

ا پ ت ٹ ث ج چ ح خ د ڈ ذ ر ڑ ز ژ س ش ص ض ط
ظ ع غ ف ق ك گ ل م ن و ھ ء ی ے آ ہ

FIGURE 13: Output text file

8. REFERENCES

[1]. Afzal, M. and Hussain, S., "Urdu Computing Standards: Urdu Zabta Takhti (UZT) 1.01", in the Proceedings of International IEEE Multi topic Conference (INMIC), Lahore University of Management Sciences (LUMS), Lahore, Pakistan, 2001.

[2]. Ethnologue, Languages of Pakistan, http://www.ethnologue.com/show_country.asp?name=Pakistan

[3]. See the Unicode Consortium website at http://unicode.org
[4]. Bhurgari, A. M. 2007. Enabling Pakistani Languages through Unicode, published at http://download.microsoft.com/download/1/4/2/142aef9f-1a74-4a24-b1f4-782d48d41a6d/PakLang.pdf
[5]. Thresholding, Image Segmentation, Digital Image Processing 2/e Rafael C. Gonzalez, Richard E. Woods.
[6]. Fast, Bruce B., Allen, Dana R. OCR image preprocessing method for image enhancement of scanned documents.
[7]. Zaheer Ahmad, Jehanzeb Khan Orakzai, Inam Shamsher, and Awais Adnan. "Urdu Nastaleeq Optical Character Recognition", "Proceedings of world academy of science, engineering and technology volume 26 december 2007".
[8]. U. Pal and Anirban Sarkar, "Recognition of Printed Urdu Script", "Proceedings of the Seventh International Conference on Document Analysis and Recognition (ICDAR 2003)".
[9]. Khalid Saeed, "New Approaches for Cursive Languages Recognition: Machine and Hand Written Script and Texts".
[10]. T. Sari and M. Sellami, "Cursive Arabic Script Segmentation and Recognition System".
[11]. Bozinovic, R.M.; Srihari, S.N, "Off-line cursive script word recognition".
[12]. Soille, P. [2003]. "Morphological Image Analysis: Principles and Applications", 2nd ed., Springer-Verlag, NY.
[13]. Dougherty. E. R. and Lotufo, R. A. [2003]. "Hands-on Morphological Image Processing", SPIE--The International Society for Optical Engineering, Bellingham, WA.
[14]. International Journal of Pattern Recognition and Artificial Intelligence.
[15]. Alasdari McAndrew, Anne Venables, "A 'Secondary' Look at Digital Image Processing".
[16]. Ganapathy, V., Lean, C.C.H., "Optical Character Recognition Program for Images of Printed Text using a Neural Network".
[17]. Nabeel Shahzad, Brandon Paulson, Tracy Hammond, "Urdu Qaeda: Recognition System for Isolated Urdu Characters".
[18]. Hermilo, Ernesto, Ramon M. "Efficiency of chain codes to represent binary objects".
[19]. Yong Kui Liua and Borut Žalik, "An efficient chain code with Huffman coding".
[20]. Shah, Z.A., "Ligature based optical character recognition of Urdu- Nastaleeq font".
[21]. Inam Shamsher, Zaheer Ahmad, Jahenzeb Khan Orakzai and Awais Adnan, "OCR For Printed Urdu Script Using Feed Forward Neural Network".
[22]. "The Origin of Urdu Language" http://www.essortment.com/all/urdulanguage_rguo.htm
[23]. T.S El-Sheikh and R.M Guindi, "computer Recognition of Arabic Cursive Script," Pattern Recognition, Vol.21, No, 4, 1988, pp.293-302.
[24]. G. Nagy Rensselaer Polytechnic Institute Troy, New York, "Chinese Character Recognition A Twenty Five Year Retrospective". Tsuyoshi Kitani t, riguchi and Masami Ilara Yoshio, "Pattern Matching in the Textract Information Extraction System".

Filtering Corrupted Image and Edge Detection in Restored Grayscale Image Using Derivative Filters

Chandra Sekhar Panda ur_chandra2002@yahoo.co.in
Lecturer, Dept. of Comp. Sci & Applications
Sambalpur University, Jyoti Vihar
Sambalpur-768019(Orissa), India

Prof. (Dr.) Srikanta Patnaik patnaik_srikanta@yahoo.co.in
Professor & Director
Interscience Institute of Management & Technology
Bhubaneswar-751015(Orissa), India

Abstract

In this paper, different first and second derivative filters are investigated to find edge map after denoising a corrupted gray scale image. We have proposed a new derivative filter of first order and described a novel approach of edge finding with an aim to find better edge map in a restored gray scale image. Subjective method has been used by visually comparing the performance of the proposed derivative filter with other existing first and second order derivative filters. The root mean square error and root mean square of signal to noise ratio have been used for objective evaluation of the derivative filters. Finally, to validate the efficiency of the filtering schemes different algorithms are proposed and the simulation study has been carried out using MATLAB 5.0.

Keywords: Derivative filter, Denoising, Image processing, Root-mean-square error, Signal-to-noise ratio.

1. INTRODUCTION

Edge detection plays a vital and forefront role in image processing for object detection. The edge of an image describes the boundary between an object and its background. Edge can be identified as a sudden change in the value of the image intensity function. So an edge separates two regions of different intensities. The objective of this paper is to find the relationship between a given pixel's intensity value and its neighborhood for determining the edge pixels on the image. The edge finding is very much helpful in solving several problems in the field of Artificial Vision and Image Processing [1]. However all the edges in an image are not due to the change in intensity values, where parameters like poor focus or refraction can result in edge in an image [2].The shape of edges in an image depends on different attributes like, lighting conditions, the noise level, type of material and the geometrical and optical properties of the object [3]. Generally, noise occurs in the image due to the result of errors in the image acquisition process, by which the intensities acquired by the pixels are not same as the pixels value in the original image [4]. The degradation models like Gaussian and Salt & Pepper are used to contaminate noise in the original image [5, 6]. For denoising a corrupted image for Gaussian noise, the Wiener filtering and for Salt & Pepper noise the Median filtering are used as reported by Tukey [7, 8]. The functionalities of Wiener filtering have been reported [9-13]. Fast median filtering algorithms are proposed by Huang et al. [14] and Astola and Campbell [15]. Different derivative filters of first and second order like Sobel, Prewitt, Laplacian, and Robert are used to find edge map in the image

[16- 23]. The different subjective and objective methods for determining the performances of edge detection operator are described [24 -27].

Section (2) describes the Gaussian and Salt & Pepper noise models to contaminate the image. It also describes the Wiener and Median filtering schemes for image restoration and the methods for evaluating the performances of edge detection operators. Section (3) classifies the first and second derivative gradient operator along with the proposed operator. Section (4) describes the different algorithms for corrupting an image, filtering of corrupted image, convolving an image with a spatial mask, edge detection filter, normalizing and thresholding an image. Section (5) presents the experimental results of different edge detection images, the subjective and objective results and finally conclusion is presented in section (6).

2. IMAGE DEGRADATION MODELS AND FILTERS

The degradation function X (m, n) for an original image Y (m, n) with noise η (m, n) can be expressed as [5, 6]:

$$X(m,n) = Y(m,n) + \eta(m,n) \tag{1}$$

Gaussian Noise Model

Gaussian noise is a type of white noise which is normally distributed over the image. Generally, noise in digital image arises during the process of digitization and transmission. Image corrupted by Gaussian noise is caused by random fluctuations in the signal during transmission. The Gaussian noise can be modeled with a probability density function as:

$$p(a) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(a-\mu)^2/2\sigma^2} \tag{2}$$

where, 'a' is the gray level, μ is the mean of 'a', and σ is the standard deviation.

Salt & Pepper Noise Model

Salt & Pepper noise is an idealized form of impulse noise model. The pixels values in grayscale image corrupted by various impulse noise models are generally replaced by values equal to or near the maximum or minimum of the allowable range. The strength of impulse noise is very high as compared to the strength of image signal. Noise impulses can be of negative or positive type. For an 8-bit grayscale image, the minimum value is 0 and maximum is 255. If the corrupted pixel is replaced according to some probability density function to either 0 or 255, then that particular impulse noise model is known as Salt & Pepper noise. The negative impulses appear as black (pepper) points and positive impulses appear as white (salt) points in the image. An image contaminated by Salt & Pepper noise degrades by sharp and sudden disturbances in the image signal and it appears as randomly scattered white and black pixels over the image. The probability density function for Salt & Pepper noise is:

$$p(a) = \begin{cases} P_x & for\, a = x \\ P_y & for\, a = y \\ 0 & otherwise \end{cases} \tag{3}$$

where, x and y are positive integers. So for an 8-bit gray scale image, x = 0 appears as black point and y = 255 appears as white point.

Wiener Filter

Wiener filter is a standard image restoration approach proposed by N. Wiener [7] that incorporates both the degradation function and statistical characteristics of noise into the restoration process. This method assumes image and noise as random processes and the objective of this filter to find an estimate of the original image such that the mean square error between them is minimized. Wiener filter estimate a prior statistical knowledge of the noise field [9-13] and the impulse response of the restoration filter is chosen such that the mean-square restoration error is minimized.

Median Filter

Median filtering is a standard nonlinear signal processing technique developed by Tukey [8] for suppressing the Salt & Pepper noise in image by removing the outliers that are the extreme pixel values. Huang et al. [14] and Astola and Campbell [15] have developed fast median filtering algorithms. It uses sliding neighborhood to process an image and determine the value of each output pixel by examining an m-by-n neighborhood around the corresponding input pixel. Median filtering arranges the pixel values in an order around the neighborhood and takes the median value as the result.

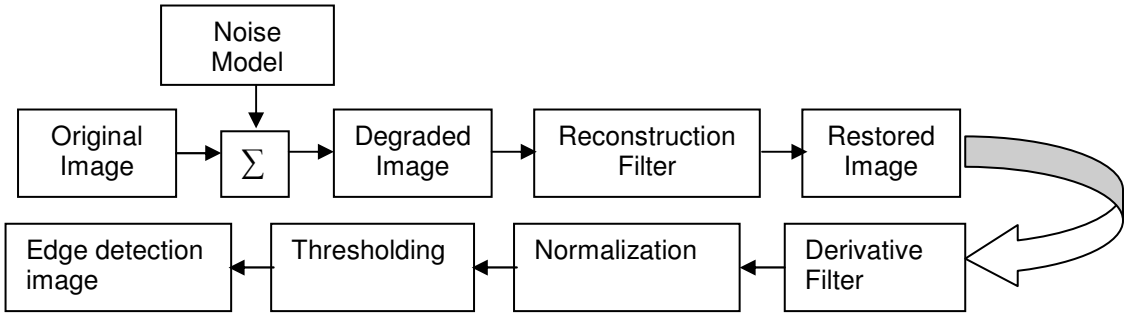


FIGURE 1: Block Diagram for Image Degradation, Restoration and Edge Detection

Root mean Square Error

The root-mean-square error e_{rms} between the original image $f(x, y)$ and the restored image $\hat{f}(x, y)$ of size M X N is defined as [5]:

$$e_{rms} = \left[\frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [\hat{f}(x, y) - f(x, y)]^2 \right]^{1/2} \quad (4)$$

Root mean Square of Signal to Noise ratio

The root-mean-square of signal to noise ratio SNR_{rms} between the original image $f(x, y)$ and the restored image $\hat{f}(x, y)$ of size M X N is defined as:

$$SNR_{rms} = \left[\frac{\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \hat{f}(x, y)^2}{\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \left[\hat{f}(x, y) - f(x, y) \right]^2} \right]^{1/2} \quad (5)$$

3. FIRST & SECOND DERIVATIVE GRADIENT FILTER

The first derivative operator follows some basic properties like; the first derivative of the gray level is negative at the leading edge of the transition, positive at the trailing edge, and zero in the areas of constant gray levels. The gradient of an image f(x, y) at the location (x, y) is given by the two dimensional column vector [19, 28].

$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \quad (6)$$

The magnitude of the first derivative is used to detect the presence of an edge in the image. The gradient vector points in the direction of the maximum rate of change of the image f at (x, y). The magnitude of this vector is given by [29]:

$$mag(\nabla f) = [G^2{}_x + G_y{}^2]^{1/2} \quad (7)$$

Here $\partial f / \partial x ..and... \partial f / \partial y$ are the rates of change of two dimensional function f(x, y) along x and y axis respectively. A pixel position is declared as an edge position if the value of the gradient exceeds some threshold value, because edge points will have higher pixel intensity values than those surrounding it. So a simple way is to compare the gradient value of a point to a threshold value and the point is said to be on edge if the threshold value is more than the gradient value of that point [30].

We have used a 3X3 region to denote image points of an input image as follows:

W_1	W_2	W_3
W_4	W_5	W_6
W_7	W_8	W_9

FIGURE 2: A 3X3 Region of an Image

$$W_1 = f(x-1, y-1), W_2 = f(x-1, y), W_3 = f(x-1, y+1)$$
$$W_4 = f(x, y-1), W_5 = f(x, y), W_6 = f(x, y+1)$$
$$W_7 = f(x+1, y-1), W_8 = f(x+1, y), W_9 = f(x+1, y+1)$$

Sobel Operator

The Sobel operator is given by the equations [5, 19, 31]:

$$G_x = (W_7 + 2W_8 + W_9) - (W_1 + 2W_2 + W_3)$$
$$G_y = (W_3 + 2W_6 + W_9) - (W_1 + 2W_4 + W_7) \quad (8)$$

Where, W_1 to W_9 are pixels values in a sub image as shown in Fig.2.

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

(a)

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

(b)

FIGURE 3: (a) Sobel Mask for Horizontal Direction
(b) Sobel Mask for Vertical Direction

Roberts Operator

The Roberts operator is given by the equations [32]:

$$G_x = W_9 - W_5$$
$$G_y = W_8 - W_6 \quad (9)$$

-1	0
0	1

(a)

0	-1
1	0

(b)

FIGURE 4: (a) Roberts Mask for Horizontal Direction
(b) Roberts Mask for Vertical Direction

Prewitt Operator

The Prewitt's operator is given by the equations [33]:

$$G_x = (W_7 + W_8 + W_9) - (W_1 + W_2 + W_3)$$
$$G_y = (W_3 + W_6 + W_9) - (W_1 + W_4 + W_7) \quad (10)$$

-1	-1	-1
0	0	0
1	1	1

(a)

-1	0	1
-1	0	1
-1	0	1

(b)

FIGURE 5: (a) Prewitt Mask for Horizontal Direction
(b) Prewitt Mask for Vertical Direction

Proposed Operator

Our proposed operator is given by equations:

$$G_x = (W_7 + 3W_8 + W_9) - (W_1 + 3W_2 + W_3)$$
$$G_y = (W_3 + 3W_6 + W_9) - (W_1 + 3W_4 + W_7) \quad (11)$$

The new mask is given by:

-1	-3	-1
0	0	0
1	3	1

(a)

-1	0	1
-3	0	3
-1	0	1

(b)

FIGURE 6: (a) Proposed Mask for Horizontal Direction
(b) Proposed Mask for Vertical Direction

Second Derivative Gradient Filter

The second derivative operator satisfies the basic properties like; the second derivative is negative for the light side of the edge, positive for the dark side of the edge, and zero for pixels lying exactly on edges [18, 29]. The sign of the second derivative is used to decide whether the edge pixel lies on the dark side or light side of an edge [34]. The second derivative at any point in an image is obtained by using the Laplacian operator [19]. The Laplacian for an image function f(x, y) of two variables is defined as [35]:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \quad (12)$$

The Laplacian operator is given by the equation:

$$\nabla^2 f = (W_2 + W_4 + W_6 + W_8) - 4W_5 \quad (13)$$

0	-1	0
-1	4	-1
0	-1	0

(a)

0	1	0
1	-4	1
0	1	0

(b)

FIGURE 7: (a) Laplacian Mask for Horizontal Direction
(b) Laplacian Mask for Vertical Direction

4. METHODS

We propose the following algorithms to find the edge map from a gray scale noisy image. The first algorithm used is to corrupt a gray scale image. Denoising a corrupted image by using

appropriate filtering technique is described in second algorithm. The third algorithm describes how to convolve an image with a given mask. Finding the edge map by different derivative operators is described in the fourth algorithm. The fifth algorithm describes the steps to normalize an image and finally thresholding an image is described in the sixth algorithm. For simulation, all the algorithms are written and executed using MATLAB.

ALGORITHM 4.1. Corrupting a gray scale image
Begin
1. Select a gray scale image for making it noisy.
2. If the noise to be added is of type = 'Additive'
   Then contaminate the image with Gaussian noise.
3. Else if the noise to be added is of type = 'Impulse'
   Then contaminate the image with Salt & Pepper noise.

End

ALGORITHM 4.2. Filtering of corrupted image for noise removal
Begin
1. Select the corrupted gray scale image created from algorithm 4.1.
2. If the type of noise in the image = 'Gaussian'
3. Then filter the corrupted image with Wiener filter.
4. Else if the type of noise in the image = 'Salt & Pepper'
5. Then filter the image with Median filter.

End

ALGORITHM 4.3. Convolving an image with odd mask
Begin
1. Select the image restored by algorithm 4.2.
2. Read all the pixel vales of restored image with M rows and N columns where, f(x, y) represents the pixel value at x and y co-ordinate.
3. Store all the pixel vales in an integer matrix of dimension M X N.
4. Select a mask 'w' of type horizontal or vertical, which is an array with dimension m X n, indexed from 0 to m -1 horizontally and 0 to n -1 vertically for m rows and n columns.
5. Fill the mask 'w' with mask coefficients.
6. The sum of all the coefficients of the mask must be zero.
7. Compute half-width of mask, a = (m – 1)/2
8. Compute half-height of mask, b= (n – 1)/2
9. Create an M X N output image, 'g' with M rows and N columns
10. for all pixel coordinates, x and y, do
11. g(x, y) = 0
12. end for
13. for y = b to N – b - 1 do
14. for x = a to M – a – 1 do
15. sum = 0
16. for k = -b to b do
17. for j = -a to a do
18. sum = sum + w(k, j) f(x + k, y + j)
19. end for
20. end for
21. end for
22. end for
23. g(x, y) = sum
24. Find the convolved image for both horizontal and vertical directions.

End

ALGORITHM 4.4. Edge detection by derivative filter
Begin

1. Select the convolved image both horizontal and vertical created by algorithm 4.3.
2. Find the magnitude of the gradient vector.
3. Magnitude = square root ((horizontal component)^2 + (vertical component)^2)
4. Finally, normalize and threshold the gradient magnitude to display the edge map.

End

ALGORITHM 4.5. Normalization of an image.
Begin

1. Select the gradient magnitude of size M X N obtained from algorithm 4.4.
2. Calculate the minimum value for each column in gradient magnitude matrix.
3. Calculate the smallest value among all the minimum column values.
4. Calculate the maximum value for each column in the gradient magnitude matrix.
5. Calculate the largest value among all the maximum column values.
6. Calculate range = largest value – smallest value
7. for x = 1 to N do
8. for y = 1 to M do
9. g (y, x) = (f(y, x) – smallest pixel value) * 255 / range.
10. end for
11. end for

End

ALGORITHM 4.6. Thresholding an image
Begin

1. Select the normalized image of size M X N obtained from algorithm 4.5.
2. Choose a value for the label.
3. for x = 1 to N do
4. for y = 1 to M do
5. if f(y, x) is greater than level then
6. if(y, x) = 255, it sets the point to white
7. else
8. f(y, x) = 0, it sets the point to black
9. end
10. end for
11. end for

End

The image is processed using the different gradient first and second derivative operators like Sobel, Robert, Prewitt, Laplacian and the proposed one to find edge map [25, 17, 20, 36]. The mask for horizontal and vertical direction is convolved to the image and the magnitude of the gradient is obtained [28, 37]. Finally the gradient magnitude is normalized and threshold to find the edge in the image [23, 4]. For writing and executing the algorithms used in this paper, we have used MATLAB 5.0 [38, 39].

5. RESULTS

To validate the efficiency of the image restoration filtering schemes and edge detection derivative filters, simulation study has been carried out using MATLAB image processing Toolbox. Two standard gray scale images 'Lena' and 'Girl' of size 256 X 256 are selected for simulation study. At first the images are distorted with Gaussian noise with mean = 0 and standard deviation = 0.01 and in the second case, the images are distorted with Salt & Pepper noise with noise density of 0.1. The threshold value is selected to be 0.2 for both the images. The performance of the

proposed derivative filter for edge detection is evaluated and compared with those of the existing derivative filters. The Wiener filter is used for cleaning Gaussian noise from the images and the Median filter is used for denoising the Salt & Pepper noise from the gray scale images. The performance of the derivative filters are evaluated by both subjective and objective method. The root mean square error and root mean square of signal to noise ratio are used to evaluate the performance of the filters. The edge detection by different derivative filters in restored gray scale image is shown in Figs. 9 to Figs. 18. The subjective fidelity scoring scales is shown in Table 1. The subjective evaluation of different gradient operators with Gaussian noise and Salt & Pepper noise are shown separately in Table 2 and Table 3 respectively. The root mean square error with Gaussian noise and Salt & Pepper noise are shown separately in Table 4 and Table 5 respectively. Finally the root mean square of signal to noise ratio with Gaussian noise and Salt & Pepper noise are shown separately in Table 6 and Table 7 respectively.

In general, it is observed that among all the derivative gradient operators, the proposed derivative operator, Sobel operator and the Prewitt operator show best performances in terms of all indices. Thus from the simulation study, it is evident that the proposed operator should be preferred for extracting good quality of edge map from the gray scale image.

Original images

FIGURE 8: Original Images without Noise
('Lena' Image Source by MathWorks Inc., USA (MATHLab))
('Girl' Image Source by USC-SIPI Image Database, California)

Edge detection by Sobel Operator

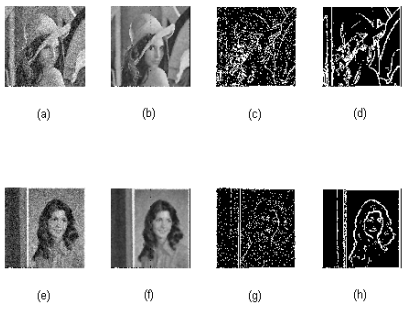


FIGURE 9: Results of Sobel Operator
(a) and (e) Images with Gaussian Noise With Mean= 0 and Standard Deviation=0.01
(b) and (f) Restored Images by Wiener Filtering(c) and (g) Edge Without Filtering
(d) and (h) Edge With Filtering

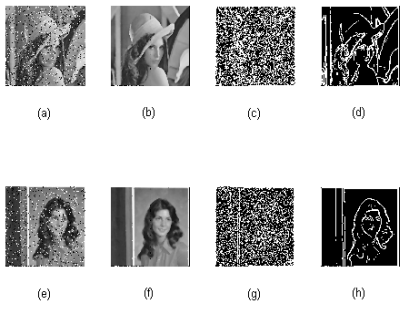


FIGURE 10: Results of Sobel Operator
(a) and (e) Images with Salt & Pepper Noise With Noise Density =0.01
(b) and (f) Restored Images by Median Filtering
(c) and (g) Edge Without Filtering
(d) and (h) Edge With Filtering

Ec

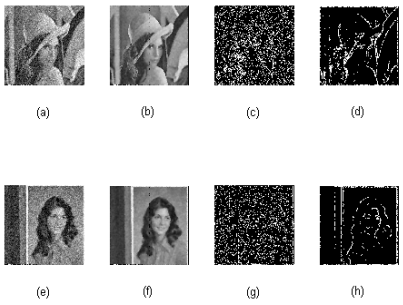


FIGURE 11: Results of Robert Operator
(a) and (e) Images With Gaussian Noise with Mean=0 and Standard Deviation=0.01
(b) and (f) Restored Images by Wiener Filtering
(c) and (g) Edge Without Filtering
(d) and (h) Edge With Filtering

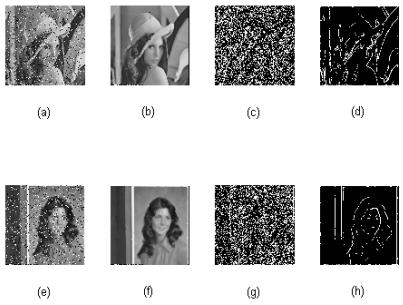


FIGURE 12: .Results of Robert Operator
(a) and (e) Images With Salt & Pepper Noise With Noise Density =0.01
(b) and (f) Restored Images by Median Filtering
(c) and (g) Edge Without Filtering
(d) and (h) Edge With Filtering

Edge detection by Prewitt Operator

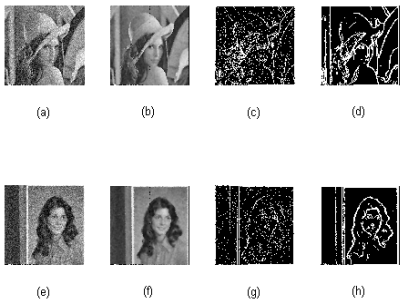


FIGURE 13: Results of Prewitt Operator
(a) and (e) Images With Gaussian Noise With Mean=0 and Standard Deviation=0.01
(b) and (f) Restored Images by Wiener Filtering
(c) and (g) Edge Without Filtering
(d) and (h) Edge With Filtering

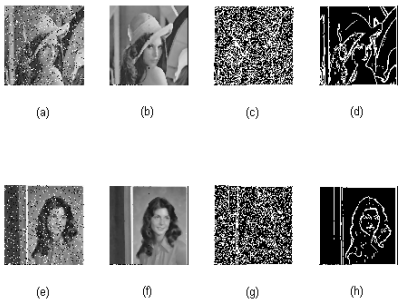


FIGURE 14: Results of Prewitt Operator
(a) and (e) Images With Salt & Pepper Noise With Noise Density =0.01
(b) and (f) Restored Images by Median Filtering
(c) and (g) Edge Without Filtering
(d) and (h) Edge With Filtering

Edge detection by Laplacian Operator

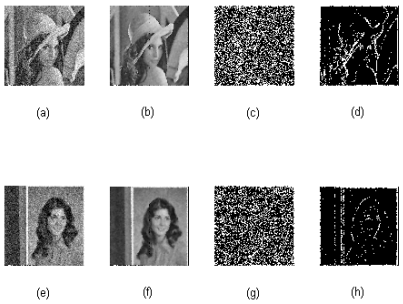


FIGURE 15: Results of Laplacian Operator
(a) and (e) Images With Gaussian Noise With Mean=0 and Standard Deviation=0.01
(b) and (f) Restored Images by Wiener Filtering
(c) and (g) Edge Without Filtering
(d) and (h) Edge With Filtering

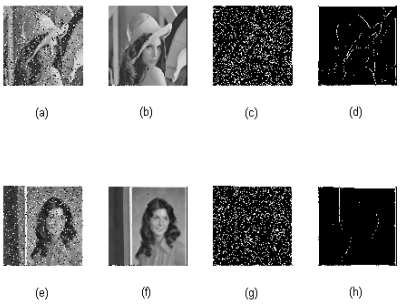


FIGURE16: Results of Laplacian Operator
(a) and (e) Images With Salt & Pepper Noise With Noise Density =0.01
(b) and (f) Restored Images by Median Filtering
(c) and (g) Edge Without Filtering
(d) and (h) Edge With Filtering

Edge detection by Proposed Operator

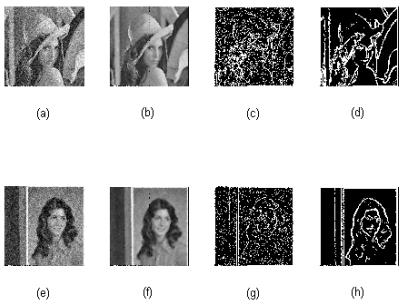


FIGURE 17: Results of Proposed Operator
(a) and (e) Images With Gaussian Noise With Mean=0 and Standard Deviation=0.01
(b) and (f) Restored Images by Wiener Filtering
(c) and (g) Edge Without Filtering
(d) and (h) Edge With Filtering

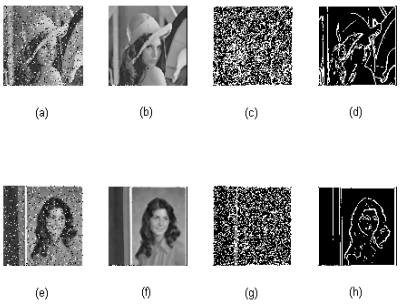


FIGURE 18: Results of Proposed Operator
(a) and (e) Images With Salt & Pepper Noise With Noise Density =0.01
(b) and (f) Restored Images by Median Filtering
(c) and (g) Edge Without Filtering
(d) and (h) Edge With Filtering

Quality	Comparison

A – Very good	+ 3 Very high
B – Good	+2 High
C – Fair	+1 Medium
D – Poor	- 1 Less
E – Bad	- 2 Much less

TABLE 1: Subjective Fidelity Scoring Scales

Operators\ Factors for Comparison	**Contrast**		**Edge Map**		**Noise Content**	
	Lena image	Girl image	Lena image	Girl image	Lena image	Girl image
Sobel	B	B	B	B	-1	-1
Roberts	C	C	C	D	+2	+1
Prewitt	B	B	B	B	-1	-1
Laplacian	D	D	D	E	+2	+2
Proposed	B	B	B	B	-1	-1

TABLE 2: Performance of Different Gradient Operators in Gaussian Noise

Operators\ Factors for Comparison	**Contrast**		**Edge Map**		**Noise Content**	
	Lena image	Girl image	Lena image	Girl image	Lena image	Girl image
Sobel	B	B	B	B	-1	-1
Roberts	C	C	C	C	+2	+1
Prewitt	B	B	B	B	-1	-1
Laplacian	D	E	C	E	+1	+1
Proposed	B	B	B	B	-1	-1

TABLE 3: Performance of Different Gradient Operators in Salt & Pepper Noise

Operators\ Factors for Comparison	**e_{rms} without filtering**	**e_{rms} with filtering**	**e_{rms} without filtering**	**e_{rms} with filtering**	**Difference in e_{rms} in Girl image**
	Lena image	Lena image	Girl image	Girl image	
Sobel	141.505	142.463	153.197	143.063	10.133
Roberts	141.183	138.744	150.991	121.379	29.611
Prewitt	141.556	143.182	152.926	137.327	15.598
Laplacian	141.131	140.407	151.179	123.261	27.917
Proposed	141.649	142.065	153.677	146.402	7.275

TABLE 4: Root-mean-square Error with Gaussian Noise

Operators\ Factors for	**e_{rms} without**	**e_{rms} with filtering**	**e_{rms} without**	**e_{rms} with filtering**	**Difference in e_{rms} in**

Comparison	filtering		filtering		Girl image
	Lena image	Lena image	Girl image	Girl image	
Sobel	146.815	144.835	149.579	136.727	12.851
Roberts	144.562	140.743	137.555	127.549	10.006
Prewitt	146.448	144.525	147.757	134.435	13.322
Laplacian	147.008	141.560	143.701	126.947	16.753
Proposed	146.354	145.308	151.063	138.912	12.150

TABLE 5: Root-mean-square Error with Salt & Pepper Noise

Operators\ Factors for Comparison	SNR_{rms} without filtering	SNR_{rms} with filtering	SNR_{rms} without filtering	SNR_{rms} with filtering
	Lena image	Lena image	Girl image	Girl image
Sobel	1.780	1.555	1.644	1.513
Roberts	1.688	0.753	1.570	0.718
Prewitt	1.764	1.417	1.633	1.408
Laplacian	1.714	0.889	1.599	0.874
Proposed	1.792	1.641	1.648	1.561

TABLE 6: Root-mean-square of Signal-to-noise Ratio with Gaussian Noise

Operators\ Factors for Comparison	SNR_{rms} without filtering	SNR_{rms} with filtering	SNR_{rms} without filtering	SNR_{rms} with filtering
	Lena image	Lena image	Girl image	Girl image
Sobel	1.602	1.226	1.510	1.142
Roberts	1.232	0.624	1.194	0.594
Prewitt	1.553	1.123	1.476	1.060
Laplacian	1.402	0.663	1.281	0.657
Proposed	1.644	1.305	1.530	1.208

TABLE 7: Root-mean-square of Signal-to-noise Ratio with Salt & Pepper Noise

6. CONCLUSION

In this paper, the proposed operator's performance for edge detection in a noisy image is evaluated both subjectively and objectively against the first and second order derivative filters and the results are shown in Fig. 9 to Fig. 18 and from Table 2 to Table 7 respectively.

The subjective evaluation of edge detected images show that proposed operator, Sobel and Prewitt operator exhibit better performances respectively for image contaminated with Gaussian noise with mean = 0 and standard deviation of 0.01 and filtered with Wiener filter in Table 2 and with Salt & Pepper noise with noise density of 0.1 and filtered with Median filter in Table 3. Table 2 and Table 3 also show that Robert and Laplacian have poor performance in terms of contrast, edge map strength and noise content. Prewitt, Sobel and proposed operator have good contrast, edge map strength and low noise content. Prewitt is more acceptable than Roberts and Laplacian while Sobel and proposed are more acceptable than Prewitt. It also shows that Laplacian is very much sensitive to noise. The objective evaluation of edge detection results as in Table 4 and Table 5 agree the subjective evaluation as in Table 2 and Table 3 that proposed, Sobel and

Prewitt operators are better than Laplacian and Robert in case of a noisy image. The root mean square error difference of Laplacian and Robert is more than Prewitt, which is more than Sobel and proposed operator. The root mean square error with less value gives better result. Table 6 and Table 7 shows that the evaluation of root mean square of signal to noise ratio of proposed and Sobel operator are higher than Prewitt, whose value is higher than Laplacian and Robert. Filtering the noisy image with a suitable filter is an initial process in the edge detection for noisy images. This paper concludes that the subjective and objective evaluation of noisy image shows that proposed, Sobel, Prewitt, Roberts and Laplacian exhibit better performance for edge detection respectively and the results of the subjective evaluation matches with the results of the objective evaluation.

7. REFERENCES

[1] Q.Ji, R.M.Haralick, “Quantitative Evaluation of Edge Detectors using the Minimum Kernel Variance Criterion”, In Proceedings of the IEEE International Conference on Image Processing ICIP 99., volume: 2, pp.705-709, 1999.

[2] E. Argyle., “Techniques for edge detection”, In Proceedings of the IEEE, vol. 59, pp. 285-286, 1971.

[3] H.Chidiac, D.Ziou, “Classification of Image Edges”, In Proceedings of the Vision Interface’99, Troise-Rivieres, Canada, pp. 17-24, 1999.

[4] Rital, S.; Bretto, A., Cherifi, H., Aboutajdine, D.; "A combinatorial edge detection algorithm on noisy images", In Proceedings of the Video/Image Processing and Multimedia Communications 4th EURASIP-IEEE Region 8 International Symposium on VIPromCom, pp.351 – 355, June, 2002.

[5] R. C. Gonzalez, R. E. Woods, “Digital Image Processing”, 2nd ed., Upper Saddle River, New Jersey, Prentice-Hall Inc.(2002).

[6] Pratt, W.K., “Digital Image Processing”, 4 th ed., Hoboken, New Jersey, John Wiley & Sons, Inc (2007).

[7] Wiener, N. “Extrapolation, Interpolation, and Smoothing of Stationary Time Series”, MIT Press, Cambridge, Mass (1942).

[8] J.W.Tukey, “Exploratory Data Analysis”, Addison-Wesley, Reading, MA (1971).

[9] C. W. Helstrom, “Image Restoration by the Method of Least Squares,” Journal of Optical Society of America, 57(3): 297–303, March 1967.

[10] J. L. Harris, Sr., “Potential and Limitations of Techniques for Processing Linear Motion-Degraded Imagery,” in Evaluation of Motion Degraded Images, US Government Printing Office, Washington DC, pp.131–138, 1968.

[11] J. L. Homer, “Optical Spatial Filtering with the Least-Mean-Square-Error Filter,” Journal of Optical Society of America, 51(5): 553–558, May 1969.

[12] J. L. Homer, “Optical Restoration of Images Blurred by Atmospheric Turbulence Using Optimum Filter Theory,” Applied Optics, 9(1): 167–171, January 1970.

[13] B. L. Lewis and D. J. Sakrison, “Computer Enhancement of Scanning Electron Micrographs,” IEEE Trans. Circuits and Systems, CAS-22(3): 267–278, March 1975.

[14] T. S. Huang, G. J. Yang and G. Y. Tang, “A Fast Two-Dimensional Median Filtering Algorithm,” IEEE Trans. Acoustics, Speech and Signal Processing, ASSP-27(1): 13–18, February 1979.

[15] J. T. Astola and T. G. Campbell, “On Computation of the Running Median,” IEEE Trans. Acoustics, Speech and Signal Processing, 37(4): 572–574, April 1989.

[16] Hueckel.,M., “ A local visual operator which recognizes edges and line”. Journal of ACM, 20(4): 634-647, Oct. 1973.

[17] T. Peli and D. Malah, “A Study of Edge Detection Algorithms” Computer Graphics and Image Processing, vol. 20: 1-21, 1982.

[18] Chanda, B., Chaudhuri, B.B. and Dutta Majumder, D., “A differentiation/ enhancement edge detector and its properties”, IEEE Trans. on System, Man and Cybern. SMC- 15: 162-168, 1985.

[19] Cyganek, C., and Siebert, J.P., “An Introduction to 3D Computer Vision Techniques and Algorithms”, New York, John Wiley & Sons, Ltd (2009).

[20] Ziou, D. and S. Tabbone, “Edge detection techniques an overview”. International Journal of Pattern Recognition Image Analysis, vol. 8: 537-559, 1998.

[21] Davis, L. S., "Edge detection techniques", Computer Graphics Image Process., vol. 4: 248-270, 1995.

[22] V.Torre and T. A. Poggio., “On edge detection”, IEEE Trans. Pattern Anal. Machine Intell., vol. PAMI-8, no.2: 187-163, Mar. 1986.

[23] Bovik, A. C., Huaung, T. S. and JR. D. C. M., "Non-parametric tests for edge detection noise", Pattern Recognition, vol.19: 209-219, 1986.

[24] M.Heath, S. Sarkar, T. Sanocki, and K.W. Bowyer. “A Robust Visual Method for Assessing the Relative Performance of Edge Detection Algorithms”, IEEE Trans. Pattern Analysis and Machine Intelligence, 19(12): 1338-1359, 1997.

[25] M. Heath, S. Sarkar, T. Sanocki, and K.W. Bowyer. “Comparison of Edge Detectors: A Methodology and Initial Study “Computer Vision and Image Understanding, 69(1): 38-54, Jan. 1998.

[26] M.C. Shin, D. Goldgof, and K.W. Bowyer. “Comparison of Edge Detector Performance through Use in an Object Recognition Task”. Computer Vision and Image Understanding, 84(1): 160-178, Oct. 2001.

[27] Umbaugh, S., "Computer Imaging: digital image analysis and processing", CRC press book (2005).

[28] Chanda, B., and Dutta Majumder, D. “Digital Image Processing and Analysis”, India, Prentice Hall of India (2008).

[29] Forsyth, D.A., and Ponce, J., “A Modern Approach”, India, Prentice Hall of India (2003).

[30] F. Bergholm. “Edge focusing,” In Proceedings of the 8th Int. Conf. Pattern Recognition, Paris, France, pp. 597- 600, 1986.

[31] Sobel, I.E., “Camera Models and Machine Perception,” Ph.D. dissertation, Stanford University, Palo Alto, California, 1970.

[32] Roberts, L.G., Tippet, J.T., “Machine Perception of Three-Dimensional Solids”, Cambridge, Mass, MIT Press (1965).

[33] Prewitt, J.M.S., Lipkin, B.S., and Rosenfeld, “A. Object Enhancement and Extraction”, New York, Academic Press (1970).

[34] Duda, R.O, Hart, P.E., “Pattern Classification and Scene Analysis”, New York, Wiley Interscience (2001).

[35] Marr, D.C. and Hildreth, E., “Theory of edge detection”, Proc. Royal Soc. London, vol. B, pp. 187-217, 1980.

[36] Yakimovsky Y., "Boundary and object detection in real world image", Journal ACM, vol. 23: 599-618, 1976.

[37] Hueckel, M., “An operator which locates edges in digitized pictures”, J. Assoc. Comput., vol. 18: 113-125, 1971.

[38] Gilat, A., “Matlab An Introduction with Applications”, New York, John Wiley & Sons, Inc (2004).

[39] “Image Processing Toolbox, User guide for use with MATLAB”, the Math Works Inc., USA (2001).

Independent Component Analysis of Edge Information for Face Recognition

Kailash J. Karande kailashkarande@yahoo.co.in
Department of Information Technology,
Sinhgad Institute of Technology, Lonavala,
Dist-Pune, Maharashtra State. (India) 410401.

Sanjay N. Talbar sntalbar@yahoo.com
Department of Electronics & Telecommunication
SGGS Institute of Engineering &Technology, Nanded,
Maharashtra State, India.

Abstract

In this paper we address the problem of face recognition using edge information as independent components. The edge information is obtained by using Laplacian of Gaussian (LoG) and Canny edge detection methods then preprocessing is done by using Principle Component analysis (PCA) before applying the Independent Component Analysis (ICA) algorithm for training of images. The independent components obtained by ICA algorithm are used as feature vectors for classification. The Euclidean distance and Mahalanobis distance classifiers are used for testing of images. The algorithm is tested on two different databases of face images for variation in illumination and facial poses up to 1800rotation angle.

Keywords: Principle Component analysis (PCA), Independent Component Analysis (ICA), Laplacian of Gaussian (LoG) and Canny edge detection Euclidean distance classifier, Mahalanobis distance classifier.

1. INTRODUCTION

Face recognition is a task that humans perform routinely and effortlessly in their daily lives. Wide availability of powerful and low-cost desktop and embedded computing systems has created an enormous interest in automatic processing of digital images and videos in a number of applications, including biometric authentication, surveillance, human-computer interaction, and multimedia management. Research and development in automatic face recognition follows naturally.

Research in face recognition is motivated not only by the fundamental challenges this recognition problem poses but also by numerous practical applications where human identification is needed. Face recognition, as one of the primary biometric technologies, became more and more important owing to rapid advances in technologies such as digital cameras, the Internet and mobile devices, and increased demands on security. Face recognition has several advantages over other biometric technologies: It is natural, non intrusive, and easy to use. Among the six biometric attributes considered by Hietmeyer [1], facial features scored the highest compatibility in a Machine Readable Travel Documents (MRTD) [2] system based on a number of evaluation factors, such as enrollment, renewal, machine requirements, and public perception.

A face recognition system is expected to identify faces present in images and videos automatically. It can operate in either or both of two modes: (1) face verification (or authentication), and (2) face identification (or recognition). Face verification involves a one-to-one match that compares a query face image against a template face image whose identity is being claimed. Face identification involves one-to-many matches that compare a query face image against all the template images in the database to determine the identity of the query face.

There are two predominant approaches to the face recognition problem: geometric (feature based) and photometric (view based). As a researcher interest in face recognition continued and many different methods are proposed, like well studied methods using Principle Component Analysis (PCA), Linear Discriminant Analysis (LDA) and Elastic Bunch Graph Matching (EBGM). In order to organize the vast field of face recognition, several approaches are conceivable. For instance, algorithms treating the face and its environment as uncontrolled systems could be distinguished from systems that control the lighting or background of the scene, or the orientation of the face. Or systems that use one or more still images for the recognition task could be distinguished from others that base their efforts on video sequences.

The comparison of face recognition using PCA and ICA on FERET database with different classifiers [3] [4] are discussed and found that the ICA has better recognition rate as compared with PCA with statistically independent basis images and also with statistically independent coefficients. Their findings are based on the frontal face image datasets are encouraging with few face expressions. Marian S Bartlett used version of ICA [5] derived from the principle of optimal information transfer through sigmoidal neurons on face images from FERET database has proved that ICA representation gave the best performance on the frontal face images. Feature selection in the independent component subspace [6] which gives the benefits for face recognition with changes in illumination and facial expressions. Fusion of ICA features like Spatial, Temporal and Localized features [7] [8] for Face Recognition are considered as optimization method. An independent Gabor features (IGFs) method and its application to face recognition using ICA [9] is discussed by Chengjun Liu; Wechsler, H. The novelty of the IGF method comes from 1) the derivation of independent Gabor features in the feature extraction stage and 2) the development of an IGF features-based probabilistic reasoning model (PRM) classification method in the pattern recognition stage. In particular, the IGF method first derives a Gabor feature vector from a set of down sampled Gabor wavelet representations of face images, then reduces the dimensionality of the vector by means of principal component analysis, and finally defines the independent Gabor features based on the independent component analysis (ICA).

The illumination has great influence on how a face image looks. Researchers have proved that for a face image, the difference caused by illumination changes has even exceeded the difference caused by identity changes [10]. The big challenge of face recognition lies in dealing with variations of pose, illumination, and expression. Also there is need to address the problem of identity changes using structural components.

In this paper we propose the face recognition using Edge information and ICA with large rotation angles with poses and variation in illumination conditions. Here we used the edge information of face images using different standard edge detection function operators like canny, Laplacian of Gaussian (log). The edge information is being used to calculate independent components for face recognition. We used the database which has the large rotation angles up to 180^0 change between the images of person while looking right and or left. We considered the face images having various orientations of the face i.e.: looking front, looking left, looking right, looking up, looking up towards left, looking up towards right, looking down. In this study we considered the samples of individual person which

consist of sufficient number of images having expressions, changes in illumination and large rotation angles. For illumination variation the effects of light on face image from left, right, top and bottom sides are considered. The paper is organized as follows. In Section 2 we introduce the ICA and information about edge detection in section 3. The section 4 specifies the need of the preprocessing before applying the ICA algorithm. The modified Fast ICA algorithm is presented in the Section 5 and classifiers are discussed in section 6. Experimental results are discussed in Section 7 and accordingly the conclusions are drawn in Section 8.

2. Introduction to ICA

ICA will be rigorously defined as a statistical 'latent variable' model. Assume that we observe n linear mixtures x_1, ..., x_n of n independent components

$$x_j = a_{j1}s_1 + a_{j2}s_2 + ... + a_{jn}s_n \quad \text{For all j.} \qquad (1)$$

we have now dropped the time index t; in the ICA model, we assume that each mixture x_j as well as each independent component s_k is a random variable, instead of a proper time signal. The observed values $x_j(t)$, e.g. the microphone signals in the cocktail party problem, are then a sample of this random variable. Without loss of generality, we can assume that both the mixture variables and the independent components have zero mean. If this is not true, then the observable variables x_i can always be centered by subtracting the sample mean, which makes the model zero-mean.

It is convenient to use vector-matrix notation instead of the sums like in the previous equation. Let us denote by **x** the random vector whose elements are the mixtures x_1, ..., x_n. And likewise by **s** the random vector with elements s_1, ..., s_n. Let us denote by **A** the matrix with elements a_{ij}. Generally, bold lower case letters indicate vectors and bold upper-case letters denote matrices. All vectors are understood as column vectors; thus $\mathbf{x}^T$ or the transpose of **x**, is a row vector. Using this vector-matrix notation, the above mixing model is written as

$$\mathbf{x} = \mathbf{A}\mathbf{s} \qquad (2)$$

Sometimes we need the columns of matrix **A**; denoting them by $\mathbf{a}_j$ the model can also be written as,

$$\mathbf{x} = \sum_{i=1}^{n} \mathbf{a}_i s_i \qquad (3)$$

The statistical model in equation.(2) is called independent component analysis or ICA model[11]. The ICA model is a generative model which means that it describes how the observed data are generated by a process of mixing the components s_i. The independent components are latent variables, meaning that they cannot be directly observed. Also the mixing matrix is assumed to be unknown. All we observe is the random vector **x**, and we must estimate both **A** and **s** using it. This must be done under as general assumptions as possible.

The starting point for ICA is the very simple assumption that the components s_i are statistically independent. It will be seen that we also assume that the independent component must have nongaussian distributions. However, in the basic model we do not assume these distributions known (if they are known, the problem is considerably simplified). For simplicity, we are also assuming that the unknown mixing matrix is square, but this assumption can be sometimes relaxed. Then, after estimating the matrix **A**, we can compute its inverse, say **W**, and obtain the independent component simply by:

$$\mathbf{s} = \mathbf{W}\mathbf{x}. \qquad (4)$$

ICA is very closely related to the method called blind source separation (BSS) or blind signal separation. A "source" means here an original signal, i.e. independent component, like the speaker in a cocktail party problem. "Blind" means that we know very little, if anything, on the mixing matrix, and make little assumptions on the source signals. ICA is one method perhaps the most widely used, for performing blind source separation.

3. Edge Detection

Edges are boundaries between different textures. Edge also can be defined as discontinuities in image intensity from one pixel to another. The edges for an image are always the important characteristics that offer an indication for a higher frequency. Detection of edges for an image may help for image segmentation, data compression, and also help for well matching, such as image reconstruction and so on.

Laplacian of a Gaussian (LoG) Detector: Consider the Gaussian function

$$h(r) = -e^{-\frac{r^2}{2\sigma^2}} \qquad (5)$$

where $r^2=x^2+y^2$ and σ is the standard deviation. This is smoothing function which, if convolved with an image, will blur it. The degree of blurring is determined by the value of σ. The Laplacian function [13] is

$$\nabla^2 h(r) = -\left[\frac{r^2-\sigma^2}{\sigma^4}\right] e^{-\frac{r^2}{2\sigma^2}} \qquad (6)$$

For obvious reason, this function is called the Laplacian of a Gaussian (LoG). Because of the second derivative is a linear operation, convolving an image with $\nabla^2 h(r)$ is the same as convolving the image with the smoothing function first and then computing the Laplacian of the result. This is the key concept underlying the LoG detector.

Canny Edge Detection:

Finds edge by looking for local maxima of the gradient of f(x, y). The gradient is calculated using the derivative of a Gaussian filter. The method uses two thresholds to detect strong and weak edges and includes the weak edges at the output only if they are connected to strong edges. Therefore this method is more likely to detect true weak edges. The Canny edge detector[13] is the most powerful edge detector provided by function edge. In this method the image is smoothed using a Gaussian filter with a specified standard deviation, σ, to reduce noise. The local gradient,

$$g(x, y) = \left[G^2 x + G^2 y\right]^{1/2} \qquad (7)$$

and edge direction,

$$\alpha(x, y) = \tan^{-1}(Gy / Gx) \qquad (8)$$

are computed at each point. Gx and Gy computed using sobel, prewitt or Roberts method of edge detection. An edge point is defined to be a point whose strength is locally maximum in the direction of the gradient.

4. Preprocessing by PCA

There are several approaches for the estimation of the independent component analysis (ICA) model. In particular, several algorithms were proposed for the estimation of the basic version of the ICA model, which has a square mixing matrix and no noise. Practically when applying the ICA algorithms to real data, some practical considerations arise and need to be taken into account. To overcome these practical considerations we implemented a preprocessing technique in this algorithm that is dimension reduction by principal

component analysis. That may be useful and even necessary before the application of the ICA algorithms in practice. Overall face recognition benefits from feature selection of PCA and ICA combination [6].

A common preprocessing technique for multidimensional data is to reduce its dimension by principal component analysis (PCA) [3]. Basically, the data is projected linearly onto a subspace

$$\tilde{X} = E_n x \qquad (9)$$

so that the maximum amount of information (in the least-squares sense) is preserved. Reducing dimension in this way has several benefits. First, let us consider the case where the number of independent components (ICs) *n* is smaller than the number of mixtures; say m. Performing ICA on the mixtures directly can cause big problems in such a case, since the basic ICA model does not hold anymore. Using PCA we can reduce the dimension of the data to *n.* After such a reduction, the number of mixtures and ICs are equal, the mixing matrix is square, and the basic ICA model holds.

The question is whether PCA is able to find the subspace correctly, so that the *n ICs* can be estimated from the reduced mixtures. This is not true in general, but in a special case it turns out to be the case. If the data consists of n ICs only, with *no* noise added, the whole data is contained in an n-dimensional subspace. Using PCA for dimension reduction clearly finds this n-dimensional subspace, since the eigenvalues corresponding to that subspace, and only those eigenvalues, are nonzero. Thus reducing dimension with PCA works correctly. In practice, the data is usually not exactly contained in the subspace, due to noise and other factors, but if the noise level is low, PCA still finds approximately the right subspace. In the general case, some weak ICs may be lost in the dimension reduction process, but PCA may still be a good idea for optimal estimation of the strong ICs.

5. ICA Algorithm

Here we used the modified FastICA algorithm [11] for computing of independent components. Before computing the independent components we use to calculate the principle components and then whitened those components to reduce the size of matrix. In the previous section we have seen the need to use PCA before applying ICA. After finding principle components we whiten the eigenvector matrix to reduce the size of matrix and make it square. To estimate several independent components, we need to run the ICA algorithm several times with weight vectors $\mathbf{w}1$, ...,$\mathbf{w}n$. To prevent different vectors from converging to the same maxima we must decorrelates the outputs $\mathbf{w}^T{}_1\mathbf{x}$, ...,$\mathbf{w}^T{}_n\ \mathbf{x}$ after every iteration.

A simple way of achieving decorrelation is a deflation scheme based on a Gram-Schmidt-like decorrelation[13]. This means that we estimate the independent components one by one. When we have estimated *p* independent components, or *p* vectors $\mathbf{w}_1$, ...,$\mathbf{w}_p$, we run the one-unit fixed-point algorithm for $\mathbf{w}_{p+1}$, and after every iteration step subtract from $\mathbf{w}_{p+1}$ the projections $\mathbf{w}^T{}_{p+1}\ \mathbf{w}_j\ \mathbf{w}_j$, j = 1, ..., p of the previously estimated *p* vectors, and then renormalize $\mathbf{w}_{p+1}$:

$$W_{p+1} = W_{p+1} - \sum_{j=1}^{p} W^T{}_{p+1} W_j W_j \qquad (10)$$

$$W_{p+1} = W_{p+1} / \sqrt{W^T{}_{p+1} W_{p+1}} \qquad (11)$$

In certain applications, however, it may be desired to use a symmetric decorrelation, in which no vectors are privileged over others. This can be accomplished, e.g., by the classical method involving matrix square roots,

$$W = (WW^T)^{-1/2}W \qquad (12)$$

where **W** is the matrix $(\mathbf{w}_1, ..., \mathbf{w}_n)^T$ of the vectors, and the inverse square root $(\mathbf{WW}^T)^{-1/2}$ is obtained from the eigenvalues decomposition of $\mathbf{WW}^T = \mathbf{FDF}^T$ as $(\mathbf{WW}^T)^{-1/2} = \mathbf{FD}^{-1/2}\mathbf{F}^T$. A simpler alternative is the following iterative algorithm,

$$W = W / \sqrt{\| WW^T \|} \qquad (13)$$

As we are using the ICA algorithm the training time required is more as compared to other methods. For the algorithm developed in this paper required around few seconds time as training time.

6. Similarity Metrics using Classifiers

As we are using the ICA algorithm the feature extraction time required is more as compared to other methods. For the algorithm developed in this paper required around few seconds time as extraction time. The query image will be more similar to the database images if the distance is smaller. For similarity measurement we used the Euclidean distance classifier and Mahalanobis distance classifier [12][14], for calculating the minimum distance between the query image and images to be matched from the database. If x and y are the two d-dimensional feature vectors of the database image and query image respectively, then these distance metrics are defined as follows. Euclidean distance to determine closeness reduces the problem to computing the distance measures:

$$d_E(x, y) = \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2} \qquad (14)$$

If the distance is small, we say the images are similar and we can decide which the most similar images in the database are. Another distance metrics for comparison of the retrieval of images used is Mahalanobis metric:

$$d_{Mah}(x, y) = \sqrt{(x-y)^2 Cov(x)^{-1}(x-y)} \qquad (15)$$

The results of these classifiers are very much close to each other. In Mahalanobis metrics the time required for similarity measure is more due to involvement of the covariance matrix.

7. Experimental Results

The experimental results presented in this paper are divided in to two parts. Both parts evaluate the face representation using Laplacian of Gaussian (LoG) and Canny edge detection for feature extraction methods. In first part the face images having variation in illumination conditions are used. In second part face images are used with large rotation angles. There are four stages of the algorithm developed in MATLAB environment. In the first stage we calculate the edge information using LoG or Canny edge detection method. The

second stage is used for preprocessing of image matrix using PCA for dimension reduction. Third stage is used to find out independent components as feature vectors using ICA algorithm. In last stage we used different two classifiers for testing of input image to be recognized.

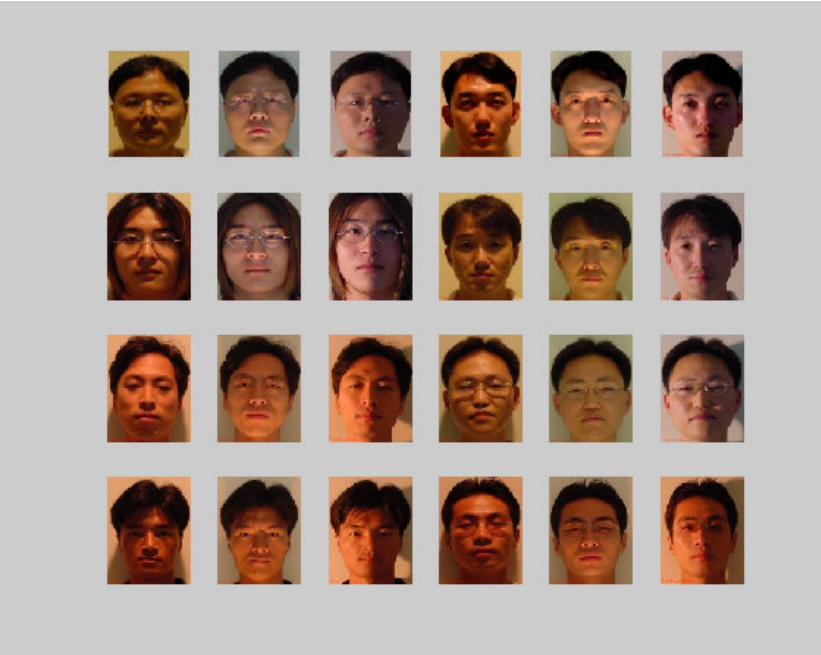


Figure 1: Various view of face images with different illumination conditions in Asian face database

Two standard databases are used for evaluation of these results. The first database is published by IIT Kanpur [16] widely used for research purpose known as *Indian face database*. In this database images of 60 persons with 10 sample images with different orientations and views are available. The second database known as *Asian face image database* [15] is from Intelligent Multimedia research laboratories having face images of 56 male persons with 10 samples each; which consist of variation in illumination conditions and different views. The resolution of all images we used in the algorithm is 128 x128 for computational purpose. Few face images are shown in Figure 1 and Figure 4 from both the databases with various views.

In this study we have explored feature selection techniques using Edge information and ICA for face recognition. Feature selection techniques are warranted especially for ICA features, since these are devoid of any importance ranking based on energy content. The study is carried out on the face database that contains both facial expression with pose variation and illumination variations. We implemented the algorithms for face recognition using Edge information and ICA on the different conditions of face images with different set of images. These set of images we used are by selecting the first 50, 100, 150 or 200 independent components.

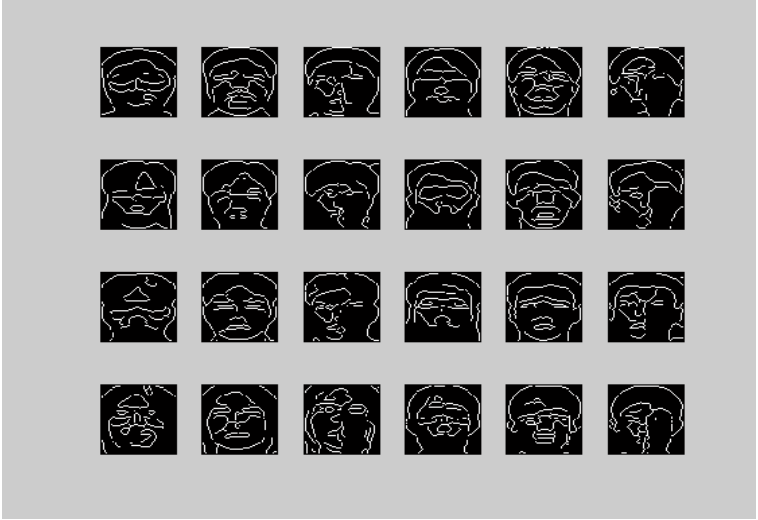


Figure 2: Edge information of few face images

The edge detected images are shown in Figure 2 and are then preprocessed by PCA. The independent components are obtained by ICA algorithm and few are shown in Figure 3. These are the independent components obtained for face images with variation in illumination. The light condition on the face images are from left and right sides as well as from top and down directions. Under different illumination conditions the results are encouraging as shown in table 1.The images used in this part are from Asian face database with different illumination conditions as shown above in Figure 1. For training we used different sets of independent components varying from 50 to 200. The edge information for the images with large rotation angles are shown in Figure 5 and corresponding few independent components are shown in Figure 6. For this part we used images from Indian face database with large rotation angle up to 180^0.

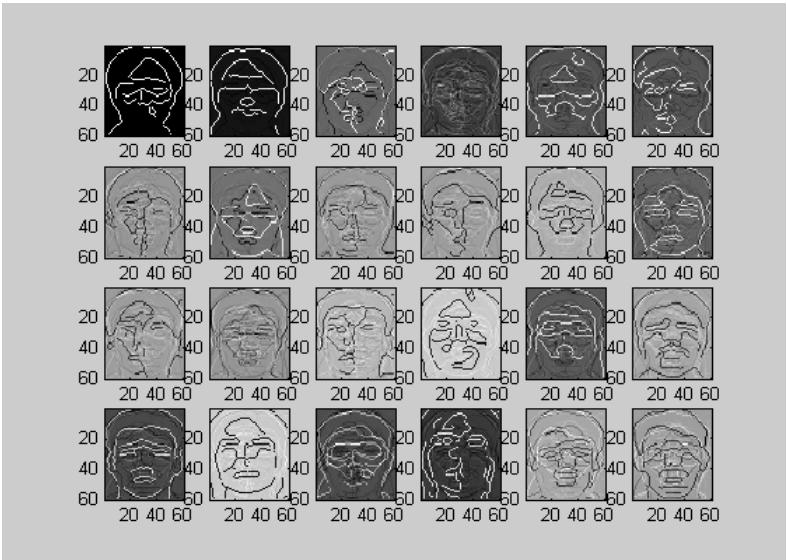


Figure 3: Independent Components of few face images

Figure 4: Various view of face images with different face orientations in Indian face database

Figure 5: Edge information of few face images

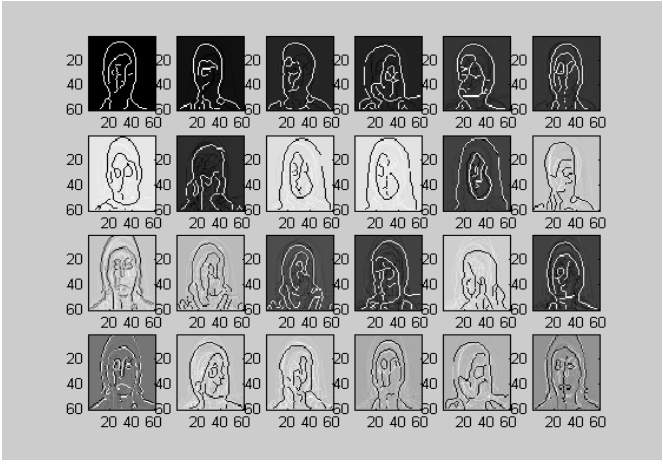


Figure 6: Independent Components of few face images

Conditions of images used for algorithm	Database used	No of Components used (PCs & ICs)	Recognition rate using LoG Edge detector and ICA (%)		Recognition rate using Canny Edge detector and ICA (%)	
Part I- Pose variation with large rotation angles	Indian face database.	Classifier used	Euclidean distance	Mahalanobis metric	Euclidean distance	Mahalanobis metric
		200	74	76.5	80.5	83.5
		150	79.33	82	84.67	86.66
		100	79	83	85	87
		50	82	86	88	91.5
Part II- Variation of illumination conditions.	Asian face database.	200	74	78	86	92.33
		150	81	83	90	93.33
		100	82	88.67	90	94
		50	84.5	87.5	91.5	95

Table 1: Results of the face recognition algorithm

8. Conclusion

In this paper an independent component analysis of Edge information of face images has been discussed and used for face recognition. Two standard databases are used, which contains the face images with different orientations, expressions and change in illumination. The performance of the algorithm suggested produces very good recognition rate varying from 74% to 95%. Applying ICA on face images for recognition; selection of ICs are required in order to give better performance. We used two different approaches for edge detection where Canny edge detector has given better results as compared to LoG edge detector. We adopted modified FastICA algorithm for computing the independent components. If we observe the results of face recognition using ICA under variation of illumination conditions the results are very good and encouraging. The results of the first part with ICA method are also close to the results of second part. This implies that ICA for edge information of face images is less sensitive to illumination change as compared to face orientations as shown in results.

9. REFERENCES

1. R. Hietmeyer. Biometric identification promises fast and secure processing of airline passengers. *The International Civil Aviation Organization Journal*, 55(9):10–11, 2000.
2. Machine Readable Travel Documents (MRTD). http://www.icao.int/mrtd/overview/overview.cfm.
3. Bruce A. Draper, Kyungim Baek, Marian Stewart Bartlett, “Recognizing faces with PCA and ICA”, Computer Vision and Image Understanding 91 (2003) 115-137.
4. Jian Yang, David Zhang, Jing-yu Yang, “Is ICA Significantly Better than PCA for Face Recognition?” Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV’05) 1550- 5499/05.
5. Marian Stewart Bartlett, Javier R. Movellan, Terrence J. Sejonowski, “Face Recognition by Independent Component Analysis”, IEEE Transactions on Neural Networks, vol-13, No-6, November 2002, PP 1450-1464.
6. H.K.Ekenel, B.Sankur, “Feature Selection in the Independent Component Subspace for Face Recognition”, Pattern Recognition Letters 25 (2004) 1377-1388.

7. Jiajin Lei, Chao Lu, "Face recognition by Spatiotemporal ICA using Facial Database Collected by AcSys FRS Discover System", Proceedings of the Seventh ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD'06).
8. Jiajin Lei, Chao Lu, "Fusion of ICA Spatial, Temporal and Localized Features for Face Recognition", Proceedings of the Seventh ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD'06).
9. Chengjun Liu; Wechsler, H." Independent component analysis of Gabor features for face recognition", Neural Networks, IEEE Transactions , Volume 14, Issue 4, July 2003 Page(s): 919 - 928
10. R. M. Bolle, J. H. Connell, and N. K. Ratha, "Biometric perils and patches," Pattern Recognition vol. 35, pp. 2727 – 2738, 2002.
11. Aapo Hyvärinen and Erkki Oja "Independent Component Analysis: Algorithms and Applications" Neural Networks Research Centre Helsinki University of Technology P.O. Box 5400, FIN-02015 HUT, Finland, Neural Networks, 13(4-5):411-430, 2000.
12. Rafael C. Gonzalez and Richard E. Woods. "Digital image processing", Second Edition, published by Pearson Education, 2003.
13. Aapo Hyvarinen, Juha Karhunen, Erkki Oja "Independent Component Analysis" Book by A Wiley Interscience Publication, John Wiley & sons, inc, New York.
14. Manesh Kokare, B.N.Chatterji and P K Biswas "Comparison of similarity metrics for texture image retrieval" International conference TENCON 2003, 571-574.
15. Asian face image database from Intelligent MultimediaLaboratory www.nova.postech.ac.kr / special / imdb /paper_pdf.pdf.
16. Indian face database www.cs.umass.edu / ~vidit / face database.