Extraction of Random Data from .png/.tif/.jpeg Image using Prewitt Operator

Tushar Nayyar
Department of Electronics Technology
Guru Nanak Dev University
Amritsar, India

Sarvpreet Singh
Department of Electronics Technology
Guru Nanak Dev University
Amritsar, India

Karamdeep Singh
Department of Electronics Technology
Guru Nanak Dev University
Amritsar, India

ABSTRACT
In this article, a method has been proposed which can be utilized for the extraction of random required data from .jpeg/.png/.tif images. Firstly, the concepts of edge detection in image processing and how it can be used for various applications are being introduced. Then, various steps that are involved in the process of edge detection are discussed in the paper. An algorithm has been developed for the extraction of required data from .jpeg/.png/.tif images. MATLAB® has been used to carry out numerical simulations. It has been found in the study that for the efficient extraction of data from .jpeg/.png/.gif/.tif images, the font size should be > 36 and the considered image should be a high contrast with threshold 0.1 to 0.34.

Keywords
Pixels, MATLAB®, Masking, Prewitt, Sobel

1. INTRODUCTION
Edge detection is an important part of image processing and is used basically to recognize the shape of an object. It was also seen that edge detection is not limited to object shapes only, it can also be used to detect alphabets and number using various algorithms.

P. S. Giri [1] proposed text information extraction and analysis from image using edge based and connected component based digital image processing techniques, has explained text isolation from its background in a digital image by creating a Gaussian pyramid of the image and down sampling to a lower resolution than the original. Analysis based on the precision and accuracy of both the techniques was also mentioned.

S. Sawant & S. Baji [2] have explained the concept of hand written character recognition through geometrical features using neural networks, in their approach they have specifically targeted the Marathi language and its recognition from a hand-written document.

G. Singla and P. Kumar [3] have studied Punjabi word recognition using sobel edge detector and then cropping the words one by one.

Tanuja. K, et al. [4] have developed hindi character recognition system using canny edge detection technique and neural networks. They proposed a 7-step recognition scheme that included scanning, pre-processing, segmentation, canny operator, distance transformation, feature extraction, feedback propagation of artificial neural network.

M. Deborah and S. Partap [5] have taken one step further and have used edge detection techniques to recognize fake currency notes. They have used a simple seven step algorithm which starts with scanning the currency note to be studied then converting the rgb image to a gray scale image followed by edge detection on the gray scale image and then using segmentation to extract the characteristics of the note and comparing it to the characteristics of the original currency note.

In this work, extraction of random data from an image file containing random data is discussed. The image file is processed in MATLAB® using the proposed algorithm. This technique can be very beneficial when secret information is to be shared. Prime focus is on binary data extraction but specific words can also be extracted through random data by implementing this technique.

This paper is organized as follows, in section 2 a basic introduction to edge detection methodologies utilizing 1st, 2nd order derivative has been provided. The proposed algorithm has been explained in section 3. Results and discussions have been presented in section 4. Conclusions have been addressed in section 5.

2. EDGE, POINT AND LINE DETECTION
Edges are very useful in image processing techniques as edge is generally the point where intensity of the image changes, these intensity changes can be identified and highlight the edges in the image. Highlighting only the edges can have numerous applications. Intensity changes or discontinuities can be of several types, few of which are shown below [6].

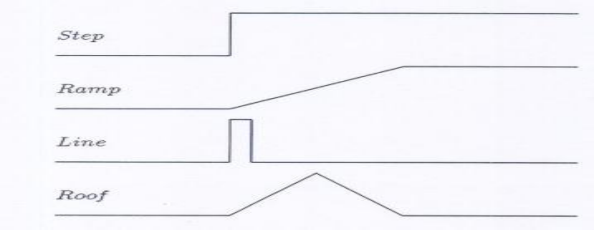

Fig 1: Types of edges [7]

As shown in Fig 1, a notable change in intensity can be observed on the edges and this is utilized to detect them.

2.1 First and second order derivatives
First order and second order derivatives are used to check the intensity difference between the pixels. First order derivatives are used to detect small intensity change whereas second order derivative is generally employed for sharp intensity changes. Derivatives of a digital function are defined in terms of difference, any approximation used in derivatives should have following properties- 1) It should be zero during uniform intensity 2) Non-zero during onset of intensity like step or

ramp 3) Should be non-zero along endpoints of intensity ramp. Second order derivative should also possess above stated properties except it should be zero along endpoints of intensity ramp. [f'(x)=f(x+1)-f(x)] is defined as first order derivative, as stated above the difference between neighboring pixels are calculated and on the basis of that difference further decisions are made. [f''(x)=f(x+2)-2f(x+1)- 2f(x)] is the equation for second order derivative, it is calculated in the same manner as first order derivative but however calculating the derivative on first order derivative will give us second order derivative around pixel (x+1) but one needs to calculate it on pixel x so 1 is subtracted from original equation.[f''(x)=f(x-1)+f(x+1)-2f(x)], is the modified equation for second order derivative. In this equation, instead of taking intensity difference of the next two pixels, intensity difference of the sum of immediately succeeding and preceding pixels are taken with the pixel at position (x) [8].

2.2 Gradient of an image

Edge detection also requires us to compute gradient operator. The gradient vector has the property that it points in the direction of greatest rate of change of (f) at location (x,y). gradient is denoted as f=g(x)=derivative of (f) w.r.t x; g(y)=derivative of (f) w.r.t y. magnitude (gradient (f))= $\sqrt{g(x)_2 + g(y)_2}$ [9].

2.3 Masking of an image

Masking of an image is a very important concept when it comes to edge detection, in the simplest terms masking can be defined as filtering of a small portion of an image on which the mask is applied, most commonly 3X3 masks are used. Response from a mask is calculated by taking sum of the products of each location. Different types of masks are available. For edge detection, we can use Robert, Sobel and Prewitt masks [10]. The gradient equations for different types of masks are shown in Table 1.

Table 1. Gradient Equation of respective mask

Gradient equation	Name of the mask
G(x)=Z9-Z5; G(y)=Z8-Z6	Robert
G(x)=(Z7+Z8+Z9) (Z1+Z2+Z3); G(y)=(Z3+Z6+Z9) (Z1+Z4+Z7)-	Prewitt
G(X)=(Z7+2Z8+Z9) (Z1+2Z2+Z3); G(y)= (Z3+2Z6+Z9) (Z1+2Z4+Z7)	Sobel

3. DATA EXTRACTION USING EDGE DETECTION IN MATLAB

Edge detection can be used to extract data in MATLAB® which can have numerous applications. Random numbers can be inserted into the image and from those random number and alphabets, useful data can be extracted. The flow chart of the proposed random data extraction from images is displayed in Fig 2 as follows.

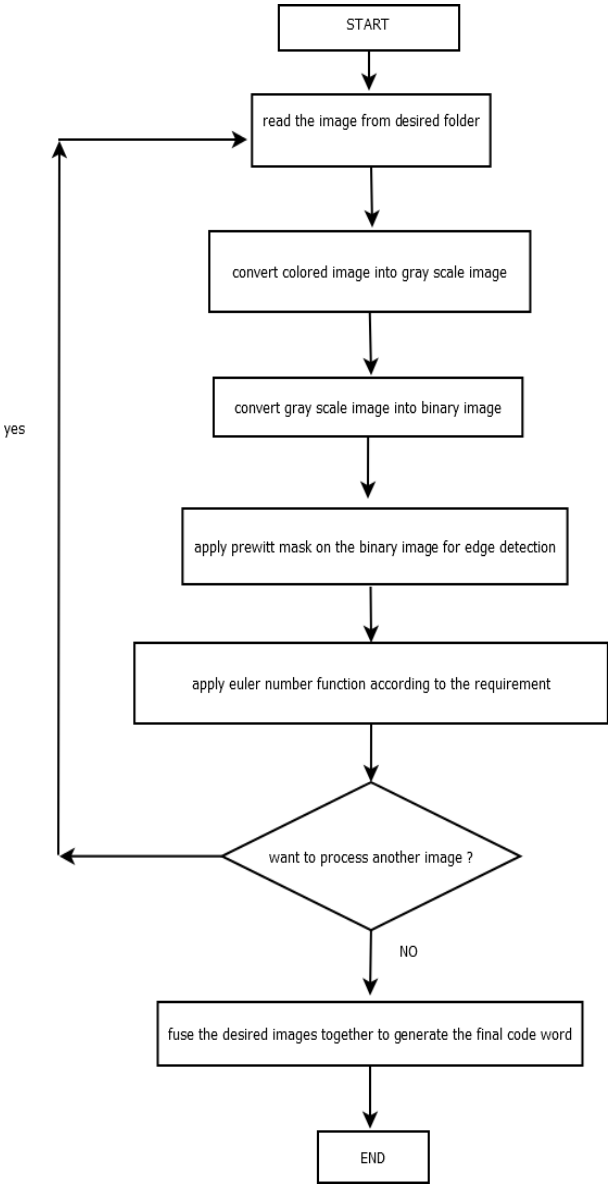

Fig 2: Flow chart of proposed algorithm for the extraction of random data from .jpeg/.tif/.png image files.

The proposed Algorithm can be explained in following steps-

1. Uploading the image in desired format.
2. The uploaded image is then converted to a gray scale image.
3. Image obtained in step 2 is then converted to a binary image. Step 2 is necessary for converting the image into a binary image, a coloured mage cannot be converted directly into a binary image.
4. Prewitt mask is then applied to the image obtained in step 3 to enhance the edges so that the Euler number function can identify the shapes and omit the necessary characters.
5. Apply Euler number function to the image obtained in step 4.

4. RESULTS & DISCUSSIONS

The performance of the proposed algorithm was evaluated for a number of image samples. In the proposed method, Euler

number function is used in two ways. Euler number [1 1] is used to omit the alphabets and numbers which contain holes and Euler number [0 0] is used to omit numbers/alphabets which do not contain any hole. After taking several samples following numbers/alphabets were completely omitted when subjected to Euler number [1 1] and Euler number [0 0] respectively as shown in Table 2.

Table 2. Type of Euler number function and omitted characters.

Function	Character omitted
Euler number [1 1]	Y U I O P D 6 8 9
Euler number [0 0]	Q W E R T U O P A S D F G H J K L Z X C V B N M 1 2 3 4 5 7

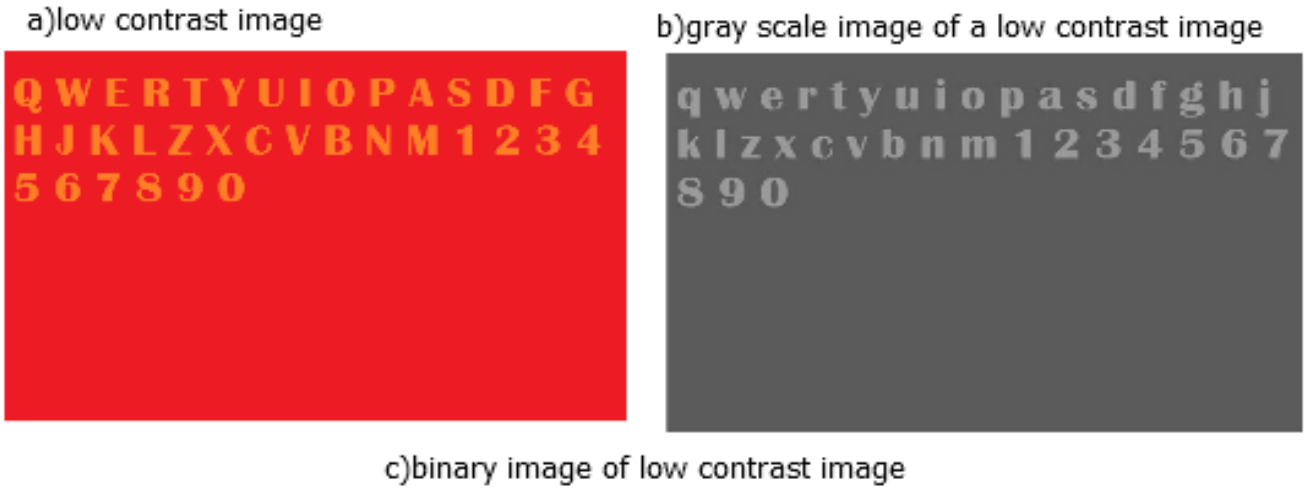

Fig 3: a) Low contrast and different font size; b) Gray scale image for low contrast image; c) Binary image for a low contrast image

Fig 3 depicts samples with low contrast, Fig 3(a) is a red background image with orange colored data, as both the colors are not very contrasting to each other their gray scale image is also low contrast as shown in Fig 3(b). the difference between the intensity of the data and the background is not very large hence it's binary conversion will give a completely white image because the threshold value is not sufficient enough to isolate the data from the background. A low contrast image can be directly processed for edge detection but that would omit one's from the image and as our primary motive is to extract binary codes, we would rather use edge detection on binary image of a high contrast image. Fig 4(a) depicts images with different font sizes and how edge detection works on them. Fig 4(a) shows a binary image with different font sizes, on observing closely we can see that as the font size increases, the shapes of the alphabets and the numbers becomes clear and hence can be easily processed for edge detection. Fig 4(a) shows edge detection on different font sizes and it is evident that alphabets with larger font size are clearly visible. Therefore, we will be working with font size of 36 and above to generate codes.

Fig 4: a) Binary image with different font sizes; b) Binary image for different font sizes.

As discussed above that it is very important to convert the image into a binary image and because the Binarize function uses Otsu's method to Binarize the image so it is very important to set a threshold value for the same. Figure 5 shows images with different threshold value. Fig 5 (a) and Fig 5 (b) show binary image with threshold value 1 and 0.9 respectively. Threshold 1.0 is very high as it can be seen that the image is completely black meaning that all intensity values are lower than this threshold whereas threshold of 0.9 gives us a clear binary image and final image obtained by setting threshold as 0.9 is shown in Fig 5 (d), this image is not the desired one but it gives an idea that as one decreases the threshold value the binary image becomes clear and the developed algorithm can work on that.

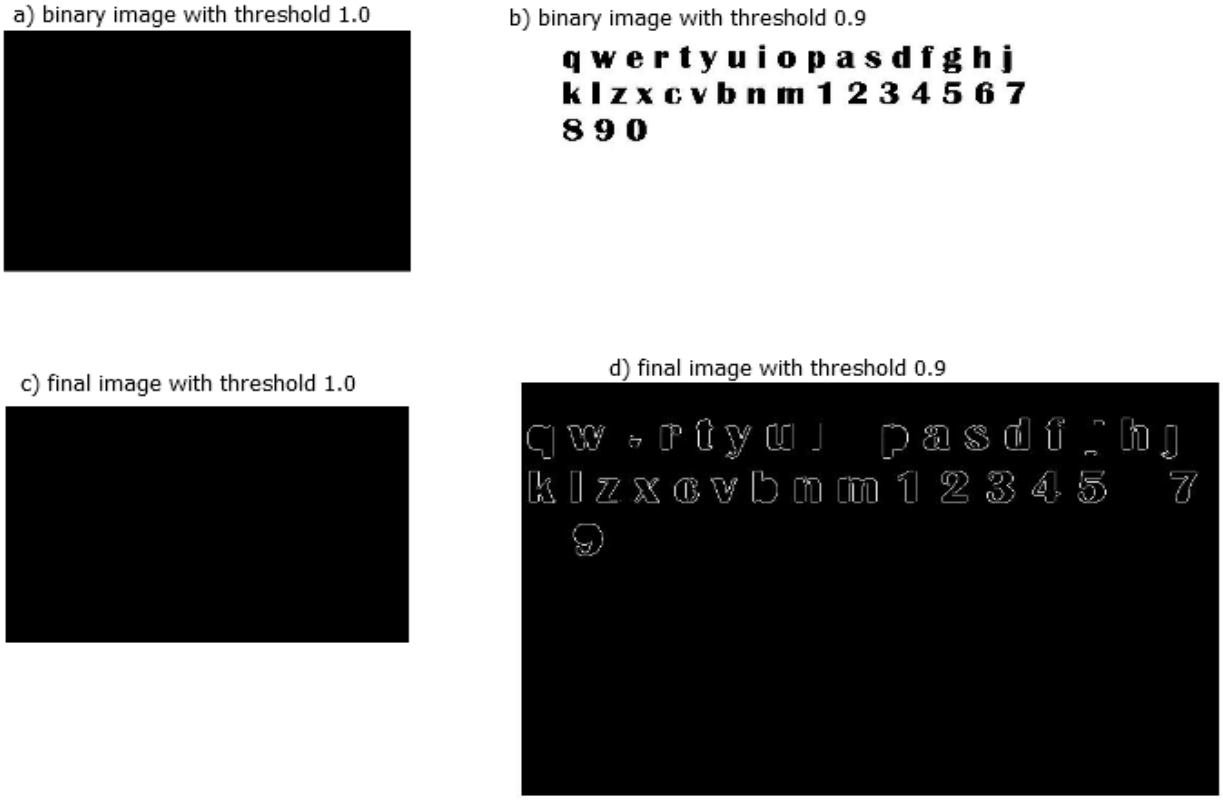

Fig 5: a) Binary image with threshold 1.0; b) Binary image with threshold 0.9; c) Final image with threshold 1.0; d) Final image with threshold 0.9.

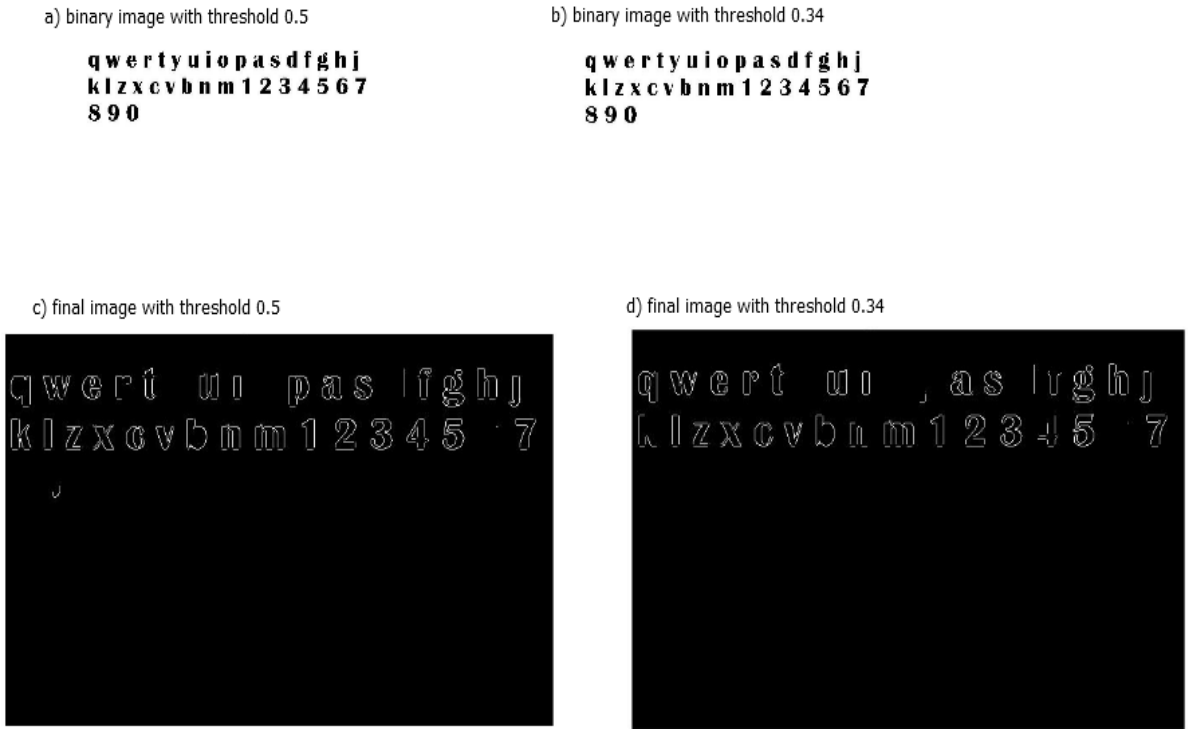

Fig 6: a) Binary image with threshold 0.5; b) Binary image with threshold 0.34; c) Final image with threshold 0.5; d) Final image with threshold 0.34.

Fig 6 (a & b) show images with a low threshold value and answer the question that, why lower threshold values are preferred. As one decreases the threshold value the binary image becomes clear i.e. the holes and one's are more clearly visible which are further enhanced by edge detection and when we apply Euler number function on the image it gives the desired result. Starting from threshold value 0.5, the results are satisfactory but on decreasing the value further it was observed that threshold value of 0.34 was best suited for this algorithm. Fig 6 depicts the images with threshold value 0.5 and 0.34.

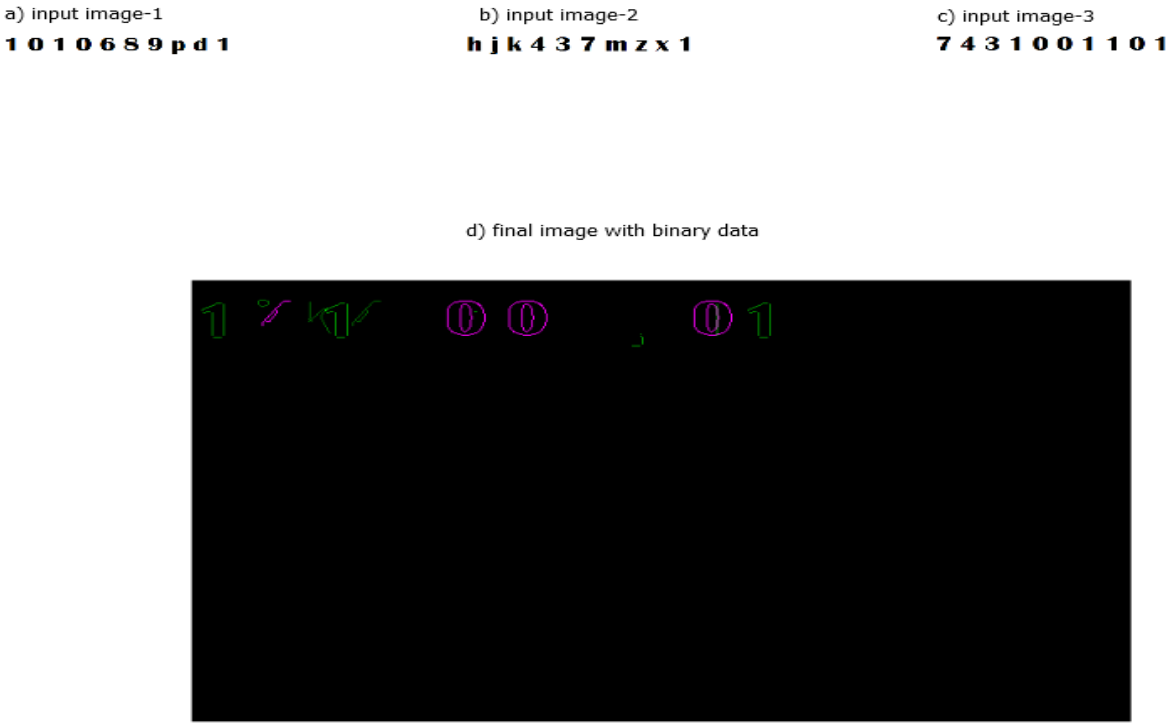

Fig 7: a) Input image No. 1; b) Input image No. 2; c) Input image No. 3; d) Final image with binary data.

Fig 7 (d) depicts the complete potential of the proposed algorithm. Desired binary data can be extracted from set of images using this algorithm. The proposed algorithm can also be used to extract textual data from a set of random images by carefully selecting the characters for the algorithm. Fig 8 (d) depicts textual data extraction from three input images.

Fig 8: a) Input image No. 1; b) Input image No. 2; c) Input image No. 3; d) Final image with extracted data.

5. CONCLUSIONS

The above algorithm has many applications especially for hiding secret data in a printed image, this algorithm can be most ideally used to extract binary data from random images, Other than extracting binary data this algorithm can also be used to create specific words but only a few words can be extracted using this algorithm because only some specific characters can be omitted. The most peculiar problem observed in this algorithm is that one has to select a specific font size and threshold level along with a specific contrast level which reduces the versatility of this algorithm. This algorithm can also be combined with other complex algorithms to detect binary numbers specifically and then using these binary numbers for some other function in the same algorithm. It can also be used as a base to design other accurate and complex algorithms to extract specific words from a printed document. The basic idea of this algorithm can be used to design neural networks that can be used to identify specific texts from a printed image. This algorithm also has the potential to extract data from a camouflaged image that can be very beneficial in hiding secret data.

6. REFERENCES

[1] Giri, P. S. (2003) Text information extraction and analysis from images using digital image processing techniques. International Journal on Advanced Computer Theory and Engineering (IJACTE), 2 (2013).

[2] Sawant S., Baji, S. (2016) Handwritten character and word recognition using their geometrical features through neural networks. International Journal of Application or Innovation in Engineering & Management (IJAIEM), 5 (2016).

[3] Singla, G., Kumar, P. (2013). Extract the punjabi word with edge detector from machine printed document images. International Journal of Computer Science & Engineering Technology (IJCSET), 4 (2013).

[4] Tanuja k, usha kumara v, suhma TM (2015). Handwritten hindi character recognition system using edge detection & neural network. International Journal of Advanced Technology and Engineering Exploration, 2 (2015).

[5] Deborah, M., Pratap, C. S. (2014). Detection of fake currency using Image Processing. (?), 1 (2014).

[6] https://www.tutorialspoint.com/dip/concept_of_edge_detection.html

[7] http://www.cse.usf.edu/~r1k/MachineVisionBook/MachineVision.files/MachineVision_Chapter5.pdf

[8] Digital image processing 3rd edition by Rafael c Gonzales and Richard e woods.

[9] https://web.cs.wpi.edu/~emmanuel/courses/cs545/S14/slides/lecture05.pdf

[10] http://www.xinapse.com/Manual/masking.html

[11] https://www.cse.unr.edu/~bebis/CS791E/Notes/EdgeDetection.pdf