Data Hiding Method Using Image Interpolation And Pixel Symmetry

Utku Esen and Berna Ors
Istanbul Technical University
Turkey
Email: esenu,siddika.ors@itu.edu.tr

***Abstract*—This article is about a new keyed steganography method for data hiding inside a digital image. Thanks to proposed method the most significant property of steganography, a large amount of secret data can be embedded while keeping a very high visual quality, is achieved. The proposed method uses image interpolation and pixel symmetry. We show our experimental results in Matlab. Similar speed and needed number of iterations when compared with previous studies can be attained.**

Keywords: Data hiding, Steganography, Image Interpolation

I. Introduction

Nowadays, electronic security systems have played significant role in our daily life. Banking and military security systems can be good example of this [3]–[5]. Electronic security systems can be divided into two main sections, which are safety providing systems [6] and safety testing or breaking [7] systems. In this article, a new data hiding method using image interpolation and pixel symmetry, which is one of the security providing systems, has been proposed.

Security providing systems are divided into two subsections which are cryptography and data hiding systems. Secured data (ciphertext) is obtained from changing unsecured data (plaintext) by utilizing certain algorithms and methods (encryption) in cryptographic systems [2]. Similar to cryptography, data hiding systems can be divided into two subsections, steganography and watermarking.

In data hiding systems, the most significant factor is median of sent message must be undistinguishable. For instance, when message is embedded on an image, it should not be figured out by others [1], [8], [9].

Image interpolation process is used for calculating unknown pixel values by utilizing known pixel values. It is one of the major operations about creating and producing digital image from a camera sensor [12]. Image interpolation methods are utilized while general processes are computed on screens of electronic devices like rotation, enlarging and reducing [13].

There are many methods in literature for image interpolation process. As examples for traditional interpolation methods, the nearest neighbour [14], linear [11], bilinear [15] and neighbour mean interpolation methods can be mentioned [1]. The nearest neighbour and linear interpolation methods are cited by simple interpolation methods and they are used for re-sampling [11]. Nearest neighbour interpolation suffers from normally unacceptable aliasing effects with regard to enlarging and reducing images. Bilinear interpolation determines the grey level from the weighted average of the four closest pixels to the specified input coordinates and it assigns a value to the output coordinates. This method generates an image that has a smoother appearance than nearest neighbour. In fact, bilinear interpolation requires three or four times higher computation time than the nearest neighbour method [1].

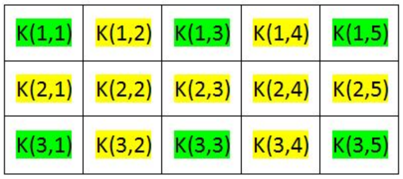
Neighbour mean interpolation similar to bilinear interpolation method. However, this method has less blurring and greater image resolution. Neighbour mean interpolation method is used for application in this article.

II. Neighbour Mean Interpolation Method

In neighbour mean interpolation method, unknown pixel values are calculated by using mean of two or three neighbour pixels. In example image on Fig. 1, $K(1,1)$, $K(1,3)$, $K(1,5)$, $K(3,1)$, $K(3,3)$ and $K(3,5)$ are known pixel values and shown by green color. $K(1,2)$, $K(1,4)$, $K(2,1)$, $K(2,2)$, $K(2,3)$, $K(2,4)$, $K(2,5)$, $K(3,1)$ and $K(3,4)$ are inter values which are wanted to be calculated and they are shown by yellow color. Value of $K(2,1)$ is calculated by meaning known values of $K(1,1)$ and $K(3,1)$. Value of $K(1,2)$ is calculated by meaning known values of $K(1,1)$ and $K(1,3)$. Value of $K(2,2)$ is calculated with Eq. 1 [1].

$$K(2,2) = \frac{K(1,1) + K(1,2) + K(2,1)}{3} \tag{1}$$

These processes can be repeated for $K(1,4)$, $K(2,3)$, $K(2,4)$, $K(2,5)$, $K(3,1)$ and $K(3,4)$ by following zig-zag way.



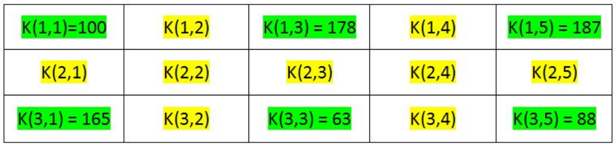
K(1,1)	K(1,2)	K(1,3)	K(1,4)	K(1,5)
K(2,1)	K(2,2)	K(2,3)	K(2,4)	K(2,5)
K(3,1)	K(3,2)	K(3,3)	K(3,4)	K(3,5)

Fig. 1: Pixels of image

III. Proposed Embedding Method

Data embedding process includes four basic elements, these are input image, cover image, binary message and stego image. Input image is the first step of selected image without modification. Cover image is the input image which is enlarged by interpolation process. Binary message is the data which want to be sent in secret. Stego image is the image which is combined by cover image and secret data by data embedding process.

Data embedding process can be summarized as follows: adding certain bits from binary message on values of interpolated pixels. The similarity between the cover and the stego image must be high in visual quality. Therefore, the pixel's value which is calculated with interpolation must be smaller than the maximum neighbour value. Maximum neighbour value is the biggest one of each neighbor pixel's values. It is illustrated by the examples below. $K(1,2)$ must be between 178 which is the maximum pixel's value and $[178+100]/2 = 139$ to achieve high visual quality in Fig. 2. In order to reach this requirement, the bit length of the message to be added must be $\lfloor log_2[178-139]\rfloor = 5$. The same process can be evaluated for calculating $K(2,1)$, but calculating $K(2,2)$ is different. For calculating $K(2,2)$, $\lfloor log_2(K(2,2)-K(1,1))\rfloor$ equation is used.



K(1,1)=100	K(1,2)	K(1,3) = 178	K(1,4)	K(1,5) = 187
K(2,1)	K(2,2)	K(2,3)	K(2,4)	K(2,5)
K(3,1) = 165	K(3,2)	K(3,3) = 63	K(3,4)	K(3,5) = 88

Fig. 2: Pixels of image

Symmetry can be found in various forms in geometry. Such as, symmetry by point and symmetry by line. Pixel symmetry is utilized similarly symmetry by point. In symmetry by point method; There are at least three points, these are named by main point, symmetry point and origin point. These points hold on the same strike, symmetry point and main point have same range from origin point but these points have different place on the strike.

At the start of the data hiding process, neighbour mean interpolation and calculation of how many bits can be added on each interpolated pixels operations are processed. Then message that will be hidden is converted to binary. Then binary message must be divided into segments according to bit number of symmetry of each interpolated pixels which can be added. Every segments of the message is added to certain symmetry points. Extracting data from stego image is the similar process.

Flow chart of the data hiding and extracting processes are shown in Fig. 3 and Fig. 4, respectively.

A. *Measuring Pixel Symmetry*

In this article, the key space, K, includes ten numbers each have ten bits. Six numbers of key space express actually three coordinate points, which are origin of getting symmetry point of pixel areas on Fig. 5. Other four numbers express the start coordinate points of embedding process on cover image.

For example, if $I(2,2)$ point is the origin of first pixel area, symmetry of $I(0,1)$ point from $I(2,2)$ point must be

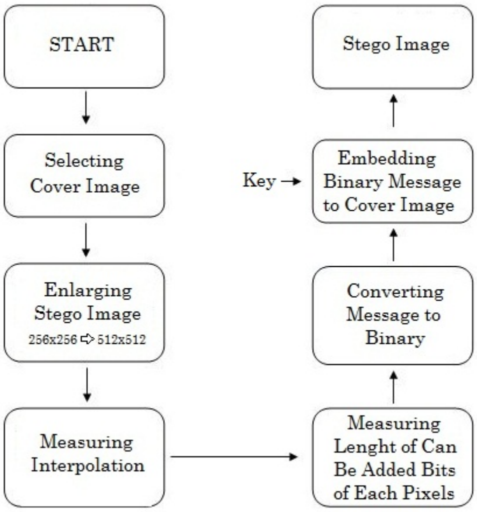

START
Selecting Cover Image
Enlarging Stego Image 256x256 ⇨ 512x512
Measuring Interpolation
Measuring Lenght of Can Be Added Bits of Each Pixels
Converting Message to Binary
Key →
Embedding Binary Message to Cover Image
Stego Image

Fig. 3: The flow chart of data hiding process

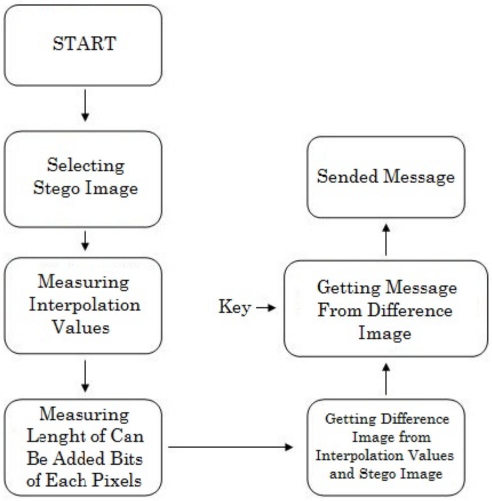

START
Selecting Stego Image
Measuring Interpolation Values
Measuring Lenght of Can Be Added Bits of Each Pixels
Getting Difference Image from Interpolation Values and Stego Image
Key →
Getting Message From Difference Image
Sended Message

Fig. 4: The flow chart of data extracting process

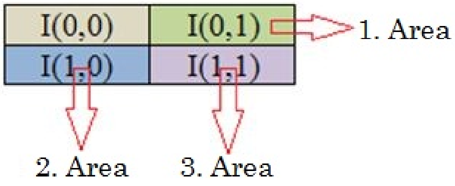
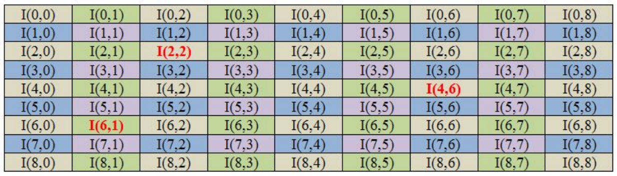


Fig. 5: Pixel areas

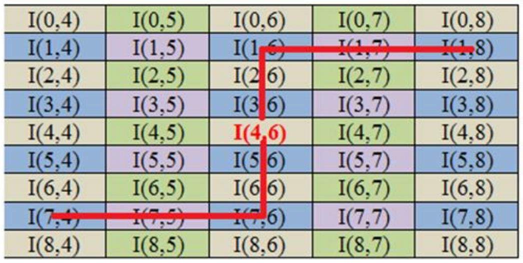
on the same area. Therefore, how many point of first pixel area are used on row and column until arriving $I(2,2)$ which is the origin point of first pixel area is measured. Then symmetry point of $I(0,1)$ can be found by using total number of points on row and column in the same way. If there is a problem about overflow in image size, counting process continues from starting or ending point of this pixel area. Thanks to this method, overlapping among symmetry points can be prevented, which can be named by injective function in mathematic. Moreover, this process is unlinear because ending and starting position of pixel areas on image is different from each other. For instance, it can be seen from Fig. 6, second pixel area starts from $I(1,0)$ and terminates at $I(1,8)$. If there is an overflow, when getting symmetry point of one of second pixel area points, mathematically, passing from $I(1,0)$ to $I(1,8)$ is different than passing from $I(1,0)$ to $I(1,3)$.



I(0,0)	I(0,1)	I(0,2)	I(0,3)	I(0,4)	I(0,5)	I(0,6)	I(0,7)	I(0,8)
I(1,0)	I(1,1)	I(1,2)	I(1,3)	I(1,4)	I(1,5)	I(1,6)	I(1,7)	I(1,8)
I(2,0)	I(2,1)	**I(2,2)**	I(2,3)	I(2,4)	I(2,5)	I(2,6)	I(2,7)	I(2,8)
I(3,0)	I(3,1)	I(3,2)	I(3,3)	I(3,4)	I(3,5)	I(3,6)	I(3,7)	I(3,8)
I(4,0)	I(4,1)	I(4,2)	I(4,3)	I(4,4)	I(4,5)	**I(4,6)**	I(4,7)	I(4,8)
I(5,0)	I(5,1)	I(5,2)	I(5,3)	I(5,4)	I(5,5)	I(5,6)	I(5,7)	I(5,8)
I(6,0)	**I(6,1)**	I(6,2)	I(6,3)	I(6,4)	I(6,5)	I(6,6)	I(6,7)	I(6,8)
I(7,0)	I(7,1)	I(7,2)	I(7,3)	I(7,4)	I(7,5)	I(7,6)	I(7,7)	I(7,8)
I(8,0)	I(8,1)	I(8,2)	I(8,3)	I(8,4)	I(8,5)	I(8,6)	I(8,7)	I(8,8)

Fig. 6: Coordinates of pixel

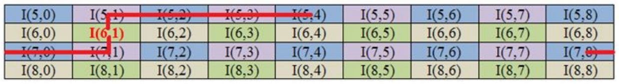
Getting symmetry point of $I(1,8)$ according to $I(4,6)$ is shown at Fig. 7. While stepping from $I(1,8)$ to $I(4,6)$ on image, two third pixel area on row and two third pixel area on column can be counted including $I(1,8)$. Likewise, counting process for getting symmetry point of $I(1,8)$ start from $I(4,6)$ to $I(7,4)$ because there are two row and column points of third pixel area.



I(0,4)	I(0,5)	I(0,6)	I(0,7)	I(0,8)
I(1,4)	I(1,5)	I(1,6)	I(1,7)	I(1,8)
I(2,4)	I(2,5)	I(2,6)	I(2,7)	I(2,8)
I(3,4)	I(3,5)	I(3,6)	I(3,7)	I(3,8)
I(4,4)	I(4,5)	**I(4,6)**	I(4,7)	I(4,8)
I(5,4)	I(5,5)	I(5,6)	I(5,7)	I(5,8)
I(6,4)	I(6,5)	I(6,6)	I(6,7)	I(6,8)
I(7,4)	I(7,5)	I(7,6)	I(7,7)	I(7,8)
I(8,4)	I(8,5)	I(8,6)	I(8,7)	I(8,8)

Fig. 7: Getting symmetry point of $I(1,8)$ according to $I(4,6)$

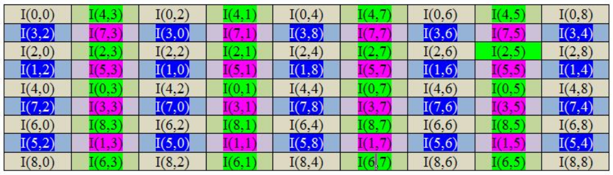
Similar to the previous process, getting symmetry point of $I(5,4)$ according to $I(6,1)$ is shown at Fig. 8. However, overflow is happened in this situation. While stepping from $I(5,4)$ to $I(6,1)$ on image, two third pixel area on row and one third pixel area on column can be counted including $I(5,4)$. counting process for getting symmetry point of $I(5,4)$ start from $I(6,1)$ to $I(7,8)$ because two row and one column points of third pixel area must be passed but there is not two column on this way and extra columns must be counted from the last point of third pixel area.



I(5,0)	I(5,1)	I(5,2)	I(5,3)	I(5,4)	I(5,5)	I(5,6)	I(5,7)	I(5,8)
I(6,0)	**I(6,1)**	I(6,2)	I(6,3)	I(6,4)	I(6,5)	I(6,6)	I(6,7)	I(6,8)
I(7,0)	I(7,1)	I(7,2)	I(7,3)	I(7,4)	I(7,5)	I(7,6)	I(7,7)	I(7,8)
I(8,0)	I(8,1)	I(8,2)	I(8,3)	I(8,4)	I(8,5)	I(8,6)	I(8,7)	I(8,8)

Fig. 8: Getting symmetry point of $I(5,4)$ according to $I(6,1)$

In order to understand well, points of first pixel area which are colored by green are measured according to $I(2,2)$, points of second pixel area which are colored by blue are measured according to $I(6,1)$ and points of third pixel area which are colored by purple are measured according to $I(4,6)$ at Fig. 6. Result of this process is shown at Fig. 9.



I(0,0)	I(4,3)	I(0,2)	I(4,1)	I(0,4)	I(4,7)	I(0,6)	I(4,5)	I(0,8)
I(3,2)	I(7,3)	I(3,0)	I(7,1)	I(3,8)	I(7,7)	I(3,6)	I(7,5)	I(3,4)
I(2,0)	I(2,3)	I(2,2)	I(2,1)	I(2,4)	I(2,7)	I(2,6)	I(2,5)	I(2,8)
I(1,2)	I(5,3)	I(1,0)	I(5,1)	I(1,8)	I(5,7)	I(1,6)	I(5,5)	I(1,4)
I(4,0)	I(0,3)	I(4,2)	I(0,1)	I(4,4)	I(0,7)	I(4,6)	I(0,5)	I(4,8)
I(7,2)	I(3,3)	I(7,0)	I(3,1)	I(7,8)	I(3,7)	I(7,6)	I(3,5)	I(7,4)
I(6,0)	I(8,3)	I(6,2)	I(8,1)	I(6,4)	I(8,7)	I(6,6)	I(8,5)	I(6,8)
I(5,2)	I(1,3)	I(5,0)	I(1,1)	I(5,8)	I(1,7)	I(5,6)	I(1,5)	I(5,4)
I(8,0)	I(6,3)	I(8,2)	I(6,1)	I(8,4)	I(6,7)	I(8,6)	I(6,5)	I(8,8)

Fig. 9: Result of measuring symmetry process

In the below, results of green points are cited.
$I(0,1)\to I(4,3)$, $I(0,3)\to I(4,1)$, $I(0,5)\to I(4,7)$, $I(0,7)\to I(4,5)$
$I(2,1)\to I(2,3)$, $I(2,3)\to I(2,1)$, $I(2,5)\to I(2,7)$, $I(2,7)\to I(2,5)$
$I(4,1)\to I(0,3)$, $I(4,3)\to I(0,1)$, $I(4,5)\to I(0,7)$, $I(4,7)\to I(0,5)$
$I(6,1)\to I(8,3)$, $I(6,3)\to I(8,1)$, $I(6,5)\to I(8,7)$, $I(6,7)\to I(8,5)$
$I(8,1)\to I(6,3)$, $I(8,3)\to I(6,1)$, $I(8,5)\to I(6,7)$, $I(8,7)\to I(6,5)$

In the below, results of blue points are cited. $I(1,0)\to I(3,2)$,
$I(1,2)\to I(3,0)$, $I(1,4)\to I(3,8)$, $I(1,6)\to I(3,6)$
$I(1,8)\to I(3,4)$, $I(3,0)\to I(1,2)$, $I(3,2)\to I(1,0)$, $I(3,4)\to I(1,8)$
$I(3,6)\to I(1,6)$, $I(3,8)\to I(1,4)$, $I(5,0)\to I(7,2)$, $I(5,2)\to I(7,0)$
$I(5,4)\to I(7,8)$, $I(5,6)\to I(7,6)$, $I(5,8)\to I(7,4)$, $I(7,0)\to I(5,2)$
$I(7,2)\to I(6,3)$, $I(7,4)\to I(6,8)$, $I(7,6)\to I(6,6)$, $I(7,8)\to I(6,4)$

In the below, results of purple points are cited.
$I(1,1)\to I(4,3)$, $I(1,3)\to I(4,1)$, $I(1,5)\to I(4,7)$, $I(1,7)\to I(4,5)$
$I(3,1)\to I(2,3)$, $I(3,3)\to I(2,1)$, $I(3,5)\to I(2,7)$, $I(3,7)\to I(2,5)$
$I(5,1)\to I(0,3)$, $I(5,3)\to I(0,1)$, $I(5,5)\to I(0,7)$, $I(5,7)\to I(0,5)$
$I(7,1)\to I(8,3)$, $I(7,3)\to I(8,1)$, $I(7,5)\to I(8,7)$, $I(7,7)\to I(8,5)$

IV. Experimental Results

Application is accomplished by four example image which are oktay, cameraman, sezen and lena. These image sizes are defined 256x256 pixels. In order to measure PSNR, cover image and stego image must be same size. Therefore, cover images which has 256x256 pixels are enlarged to 512x512 by imresize function of Matlab. Bicubic interpolation is utilized by default for this process. Then PSNR values between stego image and cover image can be measured. PSNR results are obtained generally higher than 35dB. In the below, cover images are shown at Fig. 10, stego images are shown at Fig. 11 and the PSNR results are given at Table I.

TABLE I: PSNR results between stego and cover images

Image	Oktay	Cameraman	Sezen	Lena
Size(Pixel)	256x256	256x256	256x256	256x256
Capacity(Bit)	766299	994662	819145	952652
Added Bit Number(Bit)	674496	674496	674496	674496
PSNR(dB)	45.7416	40.3941	43.9661	41.4403

V. Conclusions

In this article, data hiding inside a digital image process is made by neighboor mean interpolation method, which is the better than other interpolation methods for steganography applications. Then, in addition to this method, five tubles of cryptography is added by designing pixel symmetry algortihm and using it in process which is adding message's bit on interpolation pixels.

As a result of experimental results, the proposed embedding method holds the PSNR value above 35 dB similar to main reference of this article [1]. Therefore, The most significant property of steganography, a large amount of secret data can be embedded while keeping a very high visual quality, is achieved as well as getting same speed and computation rather than previous studies.

Number of key bits are 100 in this application. However, this number can be increased when it want by using pixel symmetry process recursively. In general, increasing number of key bits make robust the cryptography system but proposed method is not examinated by math methods. Therefore, the new algorithm may be weak about math methods. According to this situation, this article is the inter work of robust cryptographic data hiding method. If proposed method is examined by math, it can be used safely.

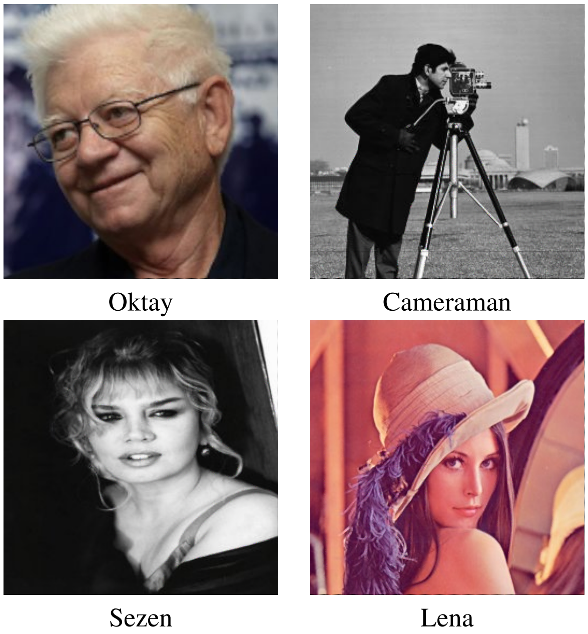
Oktay

Cameraman

Sezen

Lena

Fig. 10: Cover images



Oktay

Cameraman

Sezen

Lena

Fig. 11: Stego images

References

[1] Ki-Hyun Jung, Kee-Young Yoo,*Data hiding method using image interpolation*, Computer Standarts & Interfaces 31, pp. 465–470, (2009).

[2] Douglas R. Stinson, *Cryptography theory and practice*, Discrete Mathematics and Its Applications, pp. 1-25, (2006)

[3] Ronald L. Rivest, Adi Shamir, Leonard M. Adleman, *A method for obtaining digital signatures and public key cryptosystems*, Communications of the ACM 21(2),pp. 120–126, (1978)

[4] Dan Zhu, *Security control in inter-bank fund transfer*, Journal of Electronic Commerce Research, Vol. 3, No: 1, pp. 15–22, (2002)

[5] U. S. Army Department of Defense, *Electronic security systems*, Unified Facilities Criteria 4-021-02, (2013)

[6] Joan Daemen, Vincent Rijmen, *AES Proposal: Rijndael*, AES Algorithm Submission, September 3, (1999)

[7] Henri Gilbert, Thomas Peyrin, *Super-Sbox cryptanalysis: Improved attacks for AES-Like permutations*, FSE 2010, LNCS 6147, pp. 365-383, (2010)

[8] Minati Mishra, Priyadarsini Mishra, Flt. Lt. Dr. M.C. Adhikary, *Digital image data hiding techniques: A comparative study*, ISSN-0974-715X, ANSVESA, 7(2), pp. 105–115, (2012).

[9] A. Cheddad, J. Condell, K. Curran, P. Mc Kevitt, *Digital image steganography: Survey and Analyses of current methods*, Signal Processing, Volume 90, Issue 3, pp. 727–752, March (2010)

[10] Jan Allebach, Ping Wah Wong, *Edge-directed interpolation*, IEEE International Conference on Image Processing 3, pp. 707–710, (1996).

[11] Thomas M. Lehmann, Claudia Gonner, Klaus Spitzer, *Survey:interpolation methods in medical image processing*, IEEE Transactions on Medical Imaging 18(11), pp. 1049–1075, (1999).

[12] Bahadir K. Gunturk, John Glotzbach, Yucel Altunbasak, Ronald W. Schafer, Russel M. Mersereau, *Demosaicking: Color filter array interpolation*, IEEE Signal Processing Magazine, pp. 44–54, January (2005).

[13] Michael Unser, Philippe Thěvenaz, Leonid Yaroslavsky, *Convolution-Based interpoaltion for fast, High-Quality rotation of images*, IEEE Transactions on Image Processing, Vol. 4, N0: 10, October (1995).

[14] Rukundo Olivier, Cao Hanqianq, *Nearest neighbor value interpolation*, International Journal of Advaned Computer Science and Applications(IJACSA), Vol. 3, No: 4, (2012).

[15] Suhaib A. Fahmy, *Generalised parallel bilinear interpolation architecture for vision systems*, International Conference on Reconfugurable Computing and FPGAs, pp. 331–336, (2008).