IOSR Journal of Electronics and Communication Engineering (IOSR-JECE)
e-ISSN: 2278-2834, p- ISSN: 2278-8735.
PP 41-47
www.iosrjournals.org

Hardware Implementation of DCT Based Image Compression on Hexagonal Sampled Grid Using ARM

Jeevan K.M.[1], Anoop T.R.[2] , Deepak P[3].
[1,2,3](Department of Electronics & Communication, SNGCE, Kolenchery,Ernakulam, India)

Abstract : *The amount of data required to present image at an acceptable level of quality is extremely large .The basic goal of image compression is to represent an image with minimum number of bits of an acceptable image quality. High quality image data requires large amount of storage space and transmission bandwidth. One of the possible solutions to this problem is to compress the image so that the storage space and transmission time can be reduced. Discrete Cosine Transform (DCT) based compression is an effective way of compression in which good compression ratio without losing too much of information can be obtained. In this work DCT based Image compression on hexagonal sampling grid is performed using MATLAB and the same is implemented by using ARM processor. The conversion from rectangular grid to hexagonal grid and the compression using DCT are done using ARM LPC2148. MSE and PSNR is considered for the performance analysis.*

Keywords*: Sampling Grid, Image Compression, Hexagonal Image*

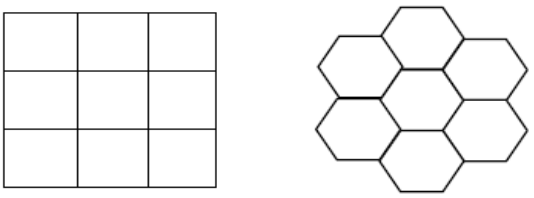
I. INTRODUCTION

New developing area in Digital Signal Processing is Hexagonal Image Processing. Image Processing in hexagonal grid is very advantageous than conventional rectangular grid. The advantages include higher angular resolution, consistent connectivity and higher sampling efficiency. Hexagonal Image system is well suited for creating the artificial human visual system, because the arrangements of the photo receptors in the human retina are in hexagonal form. Due to the non availability of hardware for capturing and displaying hexagonal based images, it limits the use of hexagonal image structure for further processing. Since there is no hardware for capturing the hexagonal images, conversion has to be done from square to hexagonal image before hexagonal image-processing.

Image compression on the hexagonal domain is a new field of research. Existing techniques used for compression are based on the conventional rectangular grid images and if we could able to perform these techniques in hexagonal grid, it will give us better performance. Research paper based on hexagonal image processing shows it gives better performance and the parameters compared for the performance analysis of Mean Square Error(MSE), Peak Signal to Noise Ration(PSNR) and Compression Ratio(CR).Image Transforms are used in image processing and image analysis. Transforms are used to move image from one domain to another. Fourier transform, Discrete Cosine Transform (DCT), Wavelet Transform etc. are the important transform used for image processing. In this work DCT based image compression on Hexagonal image is performed.

II. HEXAGONAL IMAGE REPRESENTATION

The Hexagonal Image is represented using hexagonal pixels. On a hexagonal image structure, each pixel has only six neighboring pixels which have same distance from the central pixels. Hexagonal structure closely resembles the structure of photo receptor in human visual system [5] [6]. Two type of different Architecture is shown below in Fig. 1.



(a) Rectangular Architecture (b) Spiral Architecture

Fig 1. Vision unit in Two Different Image Architecture

2.1 Spiral Architecture

Hexagonal grid is properly addressed and store data using one dimensional scheme called Spiral Architecture [12]. Spiral Architecture is shown in Fig. 2.

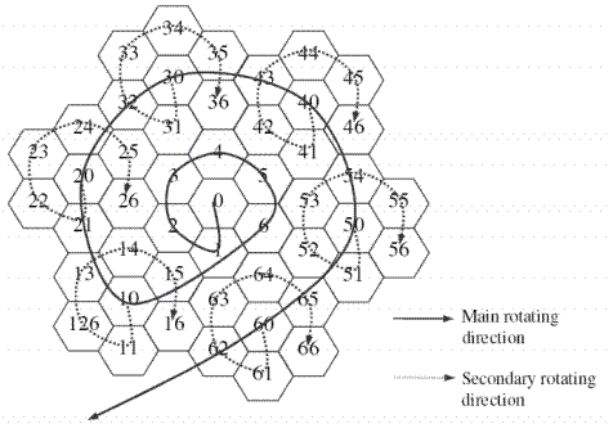


Fig 2. Spiral Architecture

In the Spiral architecture, pixels with the shape hexagons are arranged in spiral clusters. This cluster consists of the organizational units of vision. Each pixel is identified by a designated positive number. The numbered hexagons form the cluster of size 7n. The hexagons tile the plane in a recursive manner along the spiral direction.

III. HEXAGONAL RESAMPLING

Inability to capture hexagonal as it is, we are manipulate the conventional rectangle pixel to form hexagonal pixel. Hexagonal grid image based on alternate pixel suppressal method can be obtained from the conventional image by alternatively suppressing rows and columns of the existing rectangular grid and sub sampling it [4]. In this method alternate pixels are suppressed. The method is shown in Fig. 3.

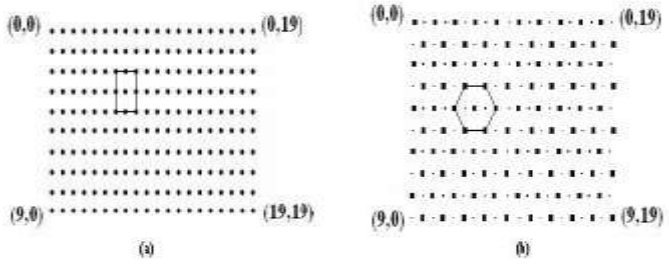


Fig 3. Alternate Pixel Suppressal Method

The suppressed pixels are not considered in the computations. There is another method to mimic the hexagonal grid called Half Shift method.

IV. DCT BASED IMAGE COMPRESSION

The Discrete Cosine Transform (DCT) transforms the time domain input into a linear combination of weighted basis functions. These basis functions are commonly the frequency components of the input. Since image is two dimensional we use 2-D Discrete Cosine Transform, which is just a one dimensional DCT applied twice, once in the 'x' direction, and again in the 'y' direction. . The DCT equation, (1), computes the ith and jth entry of the DCT of an image [7].

$$D(i,j) = \frac{1}{\sqrt{2N}} C(i)C(j) \sum_{x=0}^{N-1}\sum_{y=0}^{N-1} p(x,y) \cos\left[\frac{(2x+1)i\pi}{2N}\right] \cos\left[\frac{(2y+1)j\pi}{2N}\right] \quad \ldots (1)$$

$$C(u) = \begin{Bmatrix} \frac{1}{\sqrt{2}} \text{ if } u = 0 \\ 1 \text{ if } u > 0 \end{Bmatrix} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2)$$

The following is a general overview of the DCT process:

1) The image is broken into 8x8 block of pixels.

2) Working from left to right, top to bottom, the DCT is applied to each block.
3) Each block is compressed through quantization.
4) The array of compressed blocks that constitute the image is stored in a drastically reduced amount of space.
5) When desired, the image is reconstructed through decompression using IDCT.

The Proposed System for Hexagonal Resampling and DCT Compression is shown in Fig. 4.

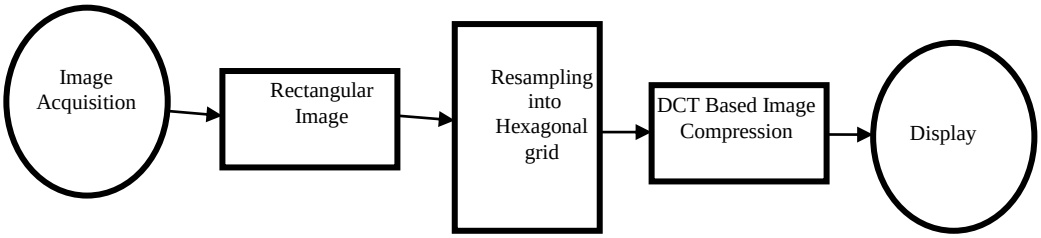


Fig 4. Proposed System for Hexagonal Resampling and DCT Compression

The process for Hexagonal representation and DCT compression using MATLAB is

1) Image acquisition: Input image is captured as rectangular pixels, and then it is stored as gray scale.
2) Resampling: Converted image is resampled into Hexagonal pixel image using Alternate Pixel Suppression Method.
3) Compression: Hexagonally resampled image is the compressed using DCT compression technique.
4) Display the output: Hexagonally resampled and DCT compressed image is taken as output

V. ARM BASED IMPLEMENTATION

Hardware implementation is done using ARM LPC2148. It is from Philips (NXP) ARM LPC21XX family; include necessary peripherals such as USB, ADC, SD/MMC, DAC, PWM, I2C etc.

ARM7LPC2148 is ARM7TDMI Core that use 16/32-bit instructions. The implementation begins by interfacing SD/MMC card. Initially card is format using FAT32 file system. Interfacing is done using SPI interface.

Image is store in card as raw value in a single raw. Then Hexagonal resampling; using Alternate Suppressal Method and corresponding DCT compression is done. Resampled value is store in the card. DCT compression is taken separately for column and raw.

Algorithm:

Hexagonal Resampling:

Step 1: Image specification declaration; define width and height of image.
Step 2: Inputting the rectangular image as raw value.
Step 3: Writing the raw value in a text file as single raw.
Step 4: Display the stored pixel value in the array (image height no: of row as and as image width as no: of column).
Step 5: Copy the pixel value in file copy.txt.
Step 6: Converting the alternative pixel value to zero and store the pixel value in the file hex.txt Hexagonal resampling is done.

DCT Compression for Row

DCT
Step 1: Read the pixel from the array ‘hex’, for row.
Step 2: Calculate the 1D DCT for length ‘N’ width of image(y=dct(x,n)).
Step 3: Pads or truncate value ‘x’ to ‘n’ before transformation.
Step 4: Input parameters: - vector ‘x’, length(x)-1, length ‘N’ and N should be >= length(x).
Step 5: Output is stored in vector ‘y’.

IDCT

Step 1: Read the value from the vector 'y'.
Step 2: Calculate the 1D IDCT for length 'N' width of image(y= idct(y,n)).
Step 3: Pads or truncate value 'y' to 'n' before transformation.
Step 4: Input parameters: - vector 'y', length(y)-1,length 'N' and N should be >= length(x).
Step 5: Output is stored in vector 'y'.
Step 6: Final output is text file row_dct.txt.

DCT Compression for Column

DCT
Step 1: Read the pixel from the array 'hex', for column.
Step 2: Calculate the 1D DCT for length 'N' heigth of image(y=dct(x,n)).
Step 3: Pads or truncate value 'x' to 'n' before transformation.
Step 4: Input parameters: - vector 'x', length(x)-1, length 'N' and N should be >= length(x).
Step 5: Output is stored in vector 'y'.
IDCT
Step 1: Read the value from the vector 'y'.
Step 2: Calculate the 1D IDCT for length 'N' width of image(y= idct(y,n)).
Step 3: Pads or truncate value 'y' to 'n' before transformation.
Step 4: Input parameters: - vector 'y', length(y)-1,length 'N' and N should be >= length(x).
Step 5: Output is stored in vector 'y'.
Step 6: Final output is text file column_dct.txt.parately for raw and column.

VI. RESULTS AND DISCUSSIONS

After implementation parameters such as Mean Square Error (MSE), Peak Signal to Noise Ratio (PSR), and Compression Ratio (CR) are calculate for rectangular grid and hexagonal grid. The values are shown in the Table 1.

TABLE I. RESULTS COMPARISON

Test Images	Sampling	MSE	PSNR	CR
water.png	Rectangular	230.30	24.51	22.87
	Hexagonal	191.30	25.31	30.46
lena.gif	Rectangular	249.5	24.16	30.46
	Hexagonal	207.5	24.97	34.49

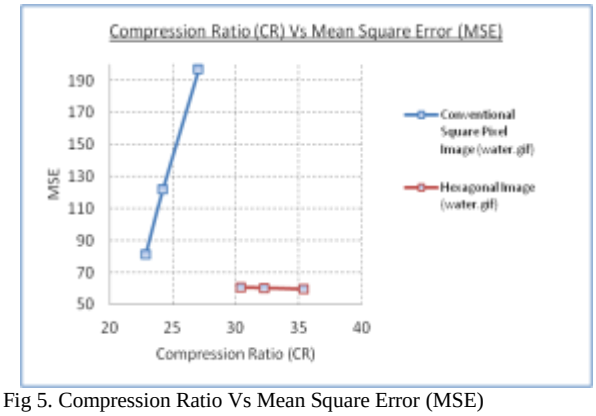


Fig 5. Compression Ratio Vs Mean Square Error (MSE)

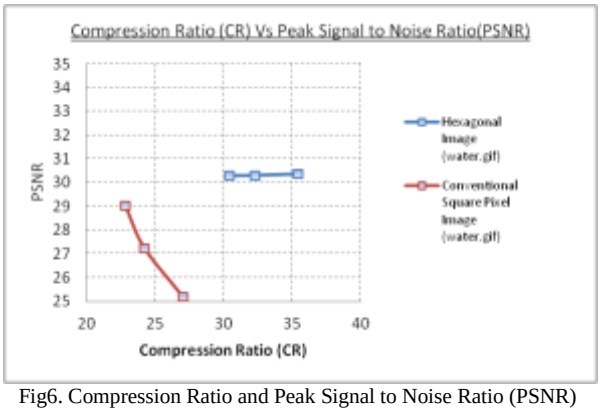
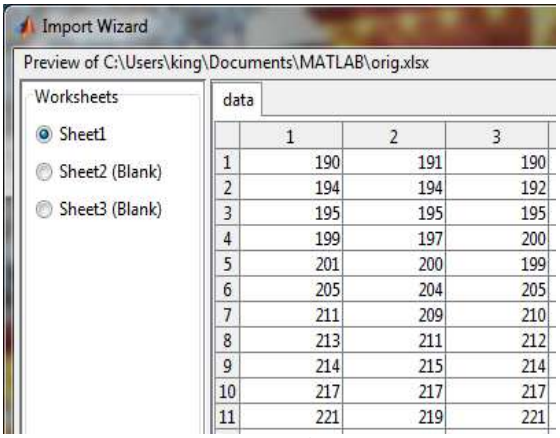


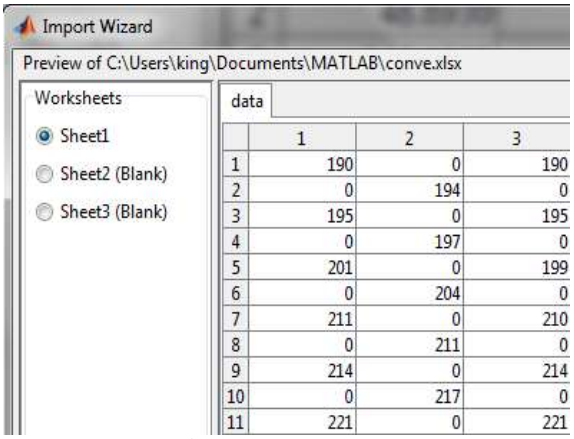
Fig6. Compression Ratio and Peak Signal to Noise Ratio (PSNR)

From the graph when CR increases the MSE is almost constant for hexagonal grid. Similarly when CR increases the PSN is almost constant for hexagonal grid.
Implementation is verified by comparing the MATLAB simulation and Coefficient value from the ARM.



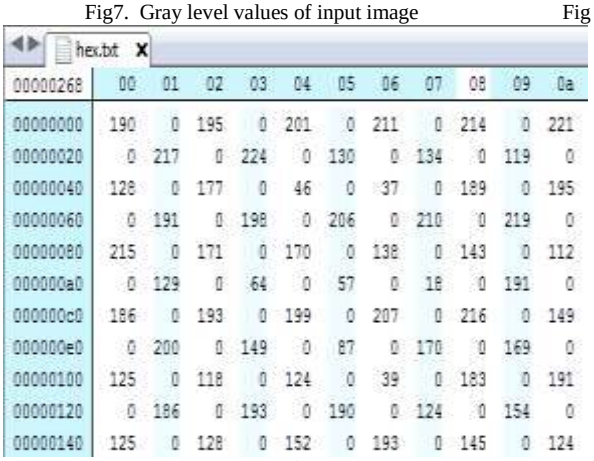
	1	2	3
1	190	191	190
2	194	194	192
3	195	195	195
4	199	197	200
5	201	200	199
6	205	204	205
7	211	209	210
8	213	211	212
9	214	215	214
10	217	217	217
11	221	219	221

Fig7. Gray level values of input image



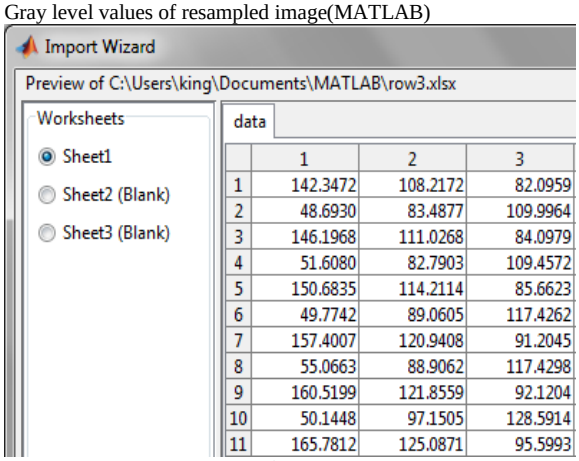
	1	2	3
1	190	0	190
2	0	194	0
3	195	0	195
4	0	197	0
5	201	0	199
6	0	204	0
7	211	0	210
8	0	211	0
9	214	0	214
10	0	217	0
11	221	0	221

Fig 8. Gray level values of resampled image(MATLAB)



00000268	00	01	02	03	04	05	06	07	08	09	0a
00000000	190	0	195	0	201	0	211	0	214	0	221
00000020	0	217	0	224	0	130	0	134	0	119	0
00000040	128	0	177	0	46	0	37	0	189	0	195
00000060	0	191	0	198	0	206	0	210	0	219	0
00000080	215	0	171	0	170	0	138	0	143	0	112
000000a0	0	129	0	64	0	57	0	18	0	191	0
000000c0	186	0	193	0	199	0	207	0	216	0	149
000000e0	0	200	0	149	0	87	0	170	0	169	0
00000100	125	0	118	0	124	0	39	0	183	0	191
00000120	0	186	0	193	0	190	0	124	0	154	0
00000140	125	0	128	0	152	0	193	0	145	0	124

Fig9. Gray level values of resampled image (ARM)



	1	2	3
1	142.3472	108.2172	82.0959
2	48.6930	83.4877	109.9964
3	146.1968	111.0268	84.0979
4	51.6080	82.7903	109.4572
5	150.6835	114.2114	85.6623
6	49.7742	89.0605	117.4262
7	157.4007	120.9408	91.2045
8	55.0663	88.9062	117.4298
9	160.5199	121.8559	92.1204
10	50.1448	97.1505	128.5914
11	165.7812	125.0871	95.5993

Fig 10 Gray level values of MATLAB Compressed Image

00000268	00	01	02	03	04	05	06	07	08	09	0a
00000000	190	0	195	0	201	0	211	0	214	0	221
00000020	0	217	0	224	0	130	0	134	0	119	0
00000040	128	0	177	0	46	0	37	0	189	0	195
00000060	0	191	0	198	0	206	0	210	0	219	0
00000080	215	0	171	0	170	0	138	0	143	0	112
000000a0	0	129	0	64	0	57	0	18	0	191	0
000000c0	186	0	193	0	199	0	207	0	216	0	149
000000e0	0	200	0	149	0	87	0	170	0	169	0
00000100	125	0	118	0	124	0	39	0	183	0	191
00000120	0	186	0	193	0	190	0	124	0	154	0
00000140	125	0	128	0	152	0	193	0	145	0	124

Fig 11 Gray level values of ARM Compressed Image

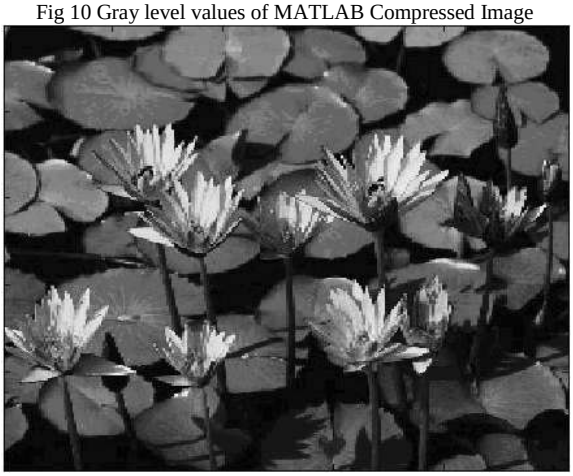


Fig 12 Image (water.png) in Rectangular Grid.

Fig 13. Compressed Image(water.png) in Rectangular Grid.

Fig 14. Image (water.png) in Hexagonal Grid.

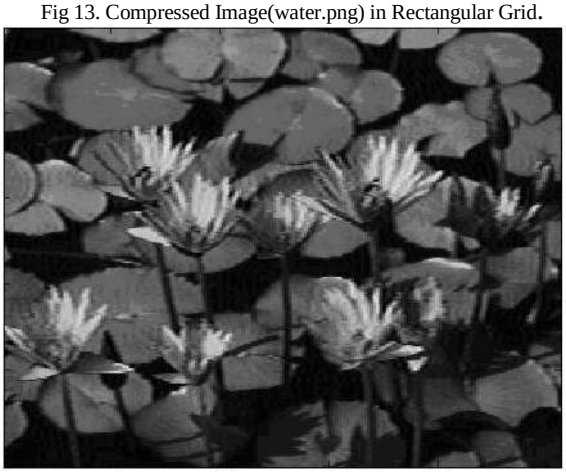


Fig 15. Compressed Image (water.png) in Hexagonal Grid.

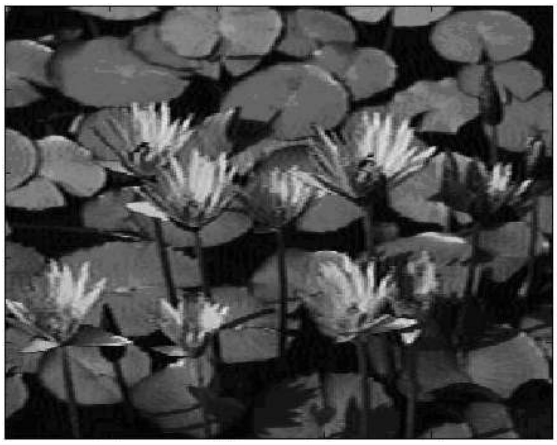


Fig 16. Compressed Image (water.png) in Hexagonal Grid.

VII. CONCLUSION

In this work DCT based image compression is performed on both rectangular grid and hexagonal grid images using MATLAB and the same is implemented using ARM processor The ARM generated values are compared with the values obtained by MATLAB and it is found that both are same. The performance is studied using MSE and PSNR. Performance comparison shows better results for DCT based image compression in hexagonal grid images. So by using hexagonal domain in image compression it is possible to get better results. conclusion section must be included and should indicate clearly the advantages, limitations, and possible applications of the paper. Although a conclusion may review the main points of the paper, do not replicate the abstract as the conclusion. A conclusion might elaborate on the importance of the work or suggest applications and extensions.

REFERENCES

[1] Jeevan K.M. "Comparative Study of DCT based Image Compression on Hexagonal and Conventional Square Pixel Images", International Journal of Computer Applications (0975 – 8887), Volume 43– No.7, April 2012, pp40-42
[2] Jeevan K.M, "Discrete Cosine Transform (DCT) Based Image Compression on Hexagonal Image", Proceedings of 2nd International Conference on Emerging Technological Trends on Advanced Engineering Research February 2012, pp 20-21. (ICETT-2012).
[3] Jeevan K.M, "Performance Comparison of DCT based Image Compression on Hexagonal and Rectangular Sampling Grid", Proceedings of 2012 IEEE 4th International Conference on Electronics Computer Technology Volume 1 ,pp 645-648.
[4] P.Vidya,S.Veni and K.A. Narayanankutty, "Performance Analysis of Edge Detection Methods on Hexagonal Sampling Grid", International Journal of Electronic Engineering Research, ISSN 0975 - 6450 Volume 1 Number 4 (2009) pp. 313–328
[5] Qiang Wu, Xiangjian He and Tom Hintz, "Preliminary Image Compression Research Using Uniform Image Partitioning on Spiral Architecture", Proceedings of the 2nd International Conference on Information Technology for Application (ICITA 2004).
[6] Lee Middleton, Jayanthi Sivawami,"Edge detection in Hexagonal Image Processing Framework," Image and Vision computing-19,2001
[7] Anil K. Jain. "Fundamentals of Digital Image Processing", Prentice Hall, 2001
[8] Mersereau, R.M., "The processing of Hexagonally Sampled Two-Dimensional Signals'. Proceedings of the IEEE, 67: p. 930-949,1979

[9] Gonzalez, R.C. and R.E.Woods, "Digital image processing", New Jersey: Prentice Hall, 2002
[10] FrameworkLee Middleton, Jayanthi Sivaswamy "Edge Detection in a Hexagonal-image Processing", Department of Electrical & Electronic Engineering, The University of Auckland Private Bag 92019, Auckland, New Zealand.
[11] N.N. Ganvir, A.D. Jadhav "Explore the Performance of the ARM ProcessorUsing JPEG" International Journal on Computer Science and Engineering, Vol. 2(1), 2010, 12-17
[12] Sheridan P, "Spiral Architecture for Machine Vision",Ph.D. thesis, University of Technology, Sydney, Australia.1996