

## **Design of Novel Algorithm and Architecture for Feature Based Corner Detection for Image Mosaicing**

Jayalaxmi H<sup>1</sup>, S. Ramachandran<sup>2</sup>

<sup>1</sup>(ECE, Acharya Institute of Technology, Bengaluru, India)

<sup>2</sup>(ECE, SJBIT, Bengaluru, India)

---

**Abstract:** A new algorithm has been developed for feature based corner detection for mosaicing images. Using this algorithm, a novel architecture suitable for FPGA/ASIC implementation has also been designed. The feature based corner detection technique proposed here is for extracting accurate corner positions than unstable corner points in existing algorithms. The design has been coded in Verilog conforming to RTL coding guidelines and fits in a single FPGA chip. The proposed design incorporates a high degree of pipelining and parallelism and hence offers high throughputs. The algorithm has also been coded in MATLAB in order to validate the hardware results. The Verilog design of the feature based corner detection architecture has been targeted on Xilinx Spartan6 xc6slx45-3fgg676 FPGA device. The design utilizes about 12,198 slices (58%) and the operating frequency is maximum of 292 MHz. The performance parameter PSNR has been computed for both the proposed and the existing methods for many number of mosaic images. The proposed method offers better reconstructed image quality and is 35 dB or more.

**Keywords:** Corner detection, Gradient Operator, Architecture, FPGA, Verilog

---

### **I. Introduction**

Image Mosaicing has grown enormously in the emerging field of Digital Signal Processing and plays a significant role in developing a panoramic view. Image mosaic finds applications in various fields such as video compression, video enhancement, digital libraries, interactive video analysis, virtual environments, low-bit rate video transmission, interactive video editing and manipulation systems [1]. The goal of image mosaicing can be stated differently depending on the specific application. Numerous methods are available in the literature for color image mosaicing [2]. FPGA platforms are suitable for the implementation of this system. Recently, high performance Field Programmable Gate Array technology has become a feasible target for the realization of algorithms suited to image processing applications [3].

Image Mosaicing methods can be further classified into Direct and Feature-based methods. Direct method for image mosaicing estimates the transformation parameters by minimizing intensity variation in the overlap region of the images [4]. However, they only work well when the dynamic areas are small compared to the overall image. Area based method uses the most similarity principle among images by calculating a cost function. This involves complex calculations on the overlapping images and undesirable artifacts appear among images mosaiced together. Frequency based methods involve the computation of translation, rotation and zoom using Fourier transforms [5]. However, these methods are limited since they require large amount of overlap areas among the images to be mosaiced.

Many alignment problems are caused in cases where the detected features are not uniform. In order to perform mosaicing of images without artifacts, one must select a suitable transformation for aligning the images [6]. Feature-based methods offer better mosaiced images than other methods mentioned earlier by minimizing alignment error depends on the gap between subsequent features [7]. Due to the property of illumination variation in images, better feature matching is possible. Corners are highly stable for variation of viewpoint in images, hence corners are good for feature matching [8].

The Feature based image mosaicing has advantage over the existing methods in terms of computational complexity. The number of similar pixel values of the detected window blocks of features should be adequately high, in spite of vary in image geometry, existence of noise, and of changes in the scanned view. The feature-based method does not work directly with intensity values of an image as compared to the area-based methods, These reasons mentioned above makes feature-based methods suitable for situations when illumination variations are expected, analysis is demanded [2]. Intensity-based Harris corner method cannot detect accurate corner positions in noisy environments. Similarly, edge-based corner method cannot identify stable corners due to incomplete edge detection. The proposed method can overcome the limitations of Harris corner and edge-based corner methods. The input image is convolved with 3×3 Gaussian kernel in order to smooth the image. This paper proposes a feature based algorithm for image mosaicing using block wise corner detection technique. The proposed design incorporates a high degree of pipelining and parallelism and hence offers high throughputs.

FPGA's are reconfigurable system has the flexibility to carryout computations in hardware to enhance performance. Digital Signal Processors [9] can be employed for mosaicing of images which provides some enhancement compared to general purpose computers. The proposed method can overcome limitations of both these methods and the irregularity encountered in the VLSI implementations [10].

The rest of the paper is organized as follows: Section 2 presents the review of related work. An overview of corner detection algorithm is given in section 3. Section 4 gives brief details of architecture development of proposed methodology suitable for FPGA/ASIC implementation. Section 5 provides experimental results and discussions. Finally conclusion is presented in section 6.

## **II. Related Work**

Image and Video Processing has been a very active field of research and development algorithms for image mosaicing, compression, filtering and smoothing. Only marginal improvement has been achieved since parallelism and pipelining incorporated in the design are inadequate.

S. Cadambi, et al., proposed Pipe Rench, an architecture that supports robust compilation and provides forward compatibility. The paper describes the need of hardware virtualization, and shows how it can be achieved using with the help of time scheduling of both configuration streams and data streams [11]. The authors have not justified how high throughput has been achieved in spite of time consuming operations. This leads to an additional computational cost for compilation of pipeline configuration.

Yiran Li has implemented Harris Corner Detection algorithm in simulation environment. It is realized in Matlab codes and then translated into VHDL by AccelDSP and the results are compared. The algorithms are implemented in typical Matlab style and in hardware style were developed [12].

Hongyan Wen, et al. projected a method with dynamic programming and grey scale variation analysis in order to find a seam line in the overlapping region in the edges of image. The seam line is removed by a hard correction method based on dynamic programming [13]. The method processes the center pixel and its neighboring area pixels so that the accuracy of the pixel error formed by image registration can be lower.

Ryo Yonemoto, et al. developed a video camera for teleconferencing without human interactions. The video camera outputs the image which is composed with panoramic images. The Fast Robust Correlation technique is applied for automatic correction processing technique and is realized on FPGA [14].

Sen Ma, et al. introduces the design of a panoramic mosaic camera, which is used in special circumstances [15]. The camera achieves panoramic perspective by analyzing optical structure and using multiple image sensors. An optimal panoramic mosaic principle which is suitable for executing in FPAG embedded system.

The limitations mentioned earlier are overcome in the proposed method in an efficient way. The image mosaicing technique is suitable for hardware implementation have been proposed in this research work to reduce the computational time. The implementation of image mosaicing algorithm can be classified into various categories based on the architectural approaches; these are General Purpose Processor (GPP), dedicated Digital Signal Processors (DSPs), Application Specific Integrated Circuits (ASICs) and Field Programmable Gate Array (FPGA) implementations [10]. The choice of the specific approach is based upon the flexibility, memory requirement, throughput and cost. The hardware implementation of image mosaicing techniques has been strongly influenced by the evolution of Very Large Scale Integration (VLSI) technology. The complex computations such as 2D convolution, correlation, filtering, smoothing, feature matching, image mosaicing and image compression operations encountered in image mosaicing technique are well suited for VLSI implementations. An enormous improvement of performance can be achieved by incorporating large pipeline and massively parallel processing techniques into hardware implementation.

Implementation of Image mosaicing algorithm on FPGA using Verilog Hardware Description Language (HDL) and obtaining real time results is a better option compared with the throughput achieved in GPP. FPGA can be reconfigured as many times as desired. In research and development, this option allows us to explore various versions of new designs. Hardware design techniques such as parallelism and pipelining techniques can be easily adapted in Verilog coding, and the same is not possible in dedicated DSPs. In terms of cost, FPGA implementation is far cheaper than ASIC design offering faster design cycle time compared with ASIC. There are remarkably few FPGA implementations reported for some of the image mosaicing algorithms. Most of them are based on schematic approach rather than Verilog or Very high speed integrated circuit HDL conforming to Register Transfer Logic (RTL) rules [16].

In the present work, in order to test the developed algorithm, standard test images have been used and results are favorably compared with that of other researchers. In order to assess the performance of the proposed algorithm, the metric Peak Signal to Noise Ratio (PSNR) has been used.

### III. Feature based Corner Detection Algorithm

Corner detection method is used in various vision tasks such as tracking, localization, image matching, recognition and image mosaicing. Algorithms involve extracting corners in particular block based on the size of the window of the image. The main region of concern for this corner detection algorithm was that edges were detected along with corners [17].

A Sobel edge detector is used to find the edges in an image where as Harris corner is used to find the corners in an image. It works by convolving a 3x3 pixels window with two horizontal and vertical filters in order to find the Sobel edge value of the center pixel in that 3x3 window. The result of applying the two filters is a gradient of image intensity at that point. By measuring the gradient, the objective is to measure change in the intensity of the pixel at that point. If the change in intensity is significant, then edge can be detected [18].

The Harris corner detector maintains 5x8 local window to calculate if the middle four pixels are corners or not. Unlike the Sobel edge detector, the Harris corner detector uses the pixel values to find corners but outputs the same values for the pixels. It uses extra four bits to mark if each of these four pixel values is a corner. As in the Sobel edge detector, it keeps track of the addressing of the pixels and passes it along with pixel output. Thus the output from the Harris corner detector is 36 bits, containing the four original pixel values of size 8 bits and, 4 bits to mark the corners and an address for the pixels [19].

Main challenge for running the edge detector came from the fact that it was in real time, giving very limited time to process a frame. Also, because edge detection for a pixel requires all the pixels surrounding it in order to store this information in memory first. Hence it was to decide to implement the Gaussian window.

Corner Detection method implementation was to take the derivatives of the image intensity in both the x and y directions [20]. This was computed by taking a pixel below or to the right (depending on the x or y direction) of the pixel for which the derivative is desired and subtracting the intensity value from that of the pixel above or to the left. In order to avoid image imperfections registering as corners, a block of 3 pixels in both the x and y directions was used in order to take the derivative as shown in Figure1.

I(x-1,y-1)	I(x,y-1)	I(x+1,y-1)
I(x-1,y)	I(x, y)	I(x+1,y)
I(x-1,y+1)	I(x,y+1)	I(x+1,y+1)

Figure 1 Matrix for finding derivatives

Based on these two derivatives, the rest of the derivatives necessary for the algorithms ( $I_x^2$ ,  $I_y^2$ , and  $I_xI_y$ ) were computed by simple multiplication. It works by convolving a 3x3 window of pixels with two horizontal and vertical filters in order to find the corner value of the center pixel in that 3x3 window as shown in Figure 2 and Figure 3. The result of applying the two filters is a gradient of image intensity of that point. By measuring the gradient, the objective is to measure change in the intensity of the pixel at that point. It convolves the window with the following two gradient kernels in the time domain.

p0	p1	p2
p3	p4	p5
p6	p7	p8

Figure 2 Pixel window 3x3

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

→ gx

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

↑ gy

Figure 3 gx and gy kernels

The calculation to obtain the gradients is shown in Eqn. (1) and Eqn. (2).

$$g_x = ((p_2 - p_0) + (p_5 - p_3) + (p_8 - p_6)) \tag{1}$$

$$g_y = ((p_0 - p_6) + (p_1 - p_7) + (p_2 - p_8)) \tag{2}$$

After calculating  $g_x$  and  $g_y$ , the module takes the magnitude of these gradients. The Harris corner detector uses both x and y gradients in order to find the value of pixel are a corner or edge. To be corner, there has to be significant changes in both directions [20, 21]. However, the corner algorithm calculates the likelihood that a pixel is a corner from summing all the surrounding gradients around a pixel and performing some calculations to determine its likelihood of being a corner as shown in Figure 4. In order to extract the corners, the corner detector needs a 3x3 window to compute a score for the pixel at location (2, 2). The values of each of these derivatives were computed for every point in the image as shown in Eqn. (3).

$$R_{out(x,y)} = \sum_{i=1}^m Ix^2 * \sum_{j=1}^n Iy^2 - (\sum_{i=1}^m Ix \sum_{j=1}^n Iy)^2 - k(\sum_{i=1}^m Ix^2 * \sum_{j=1}^n Iy^2)^2 \quad (3)$$

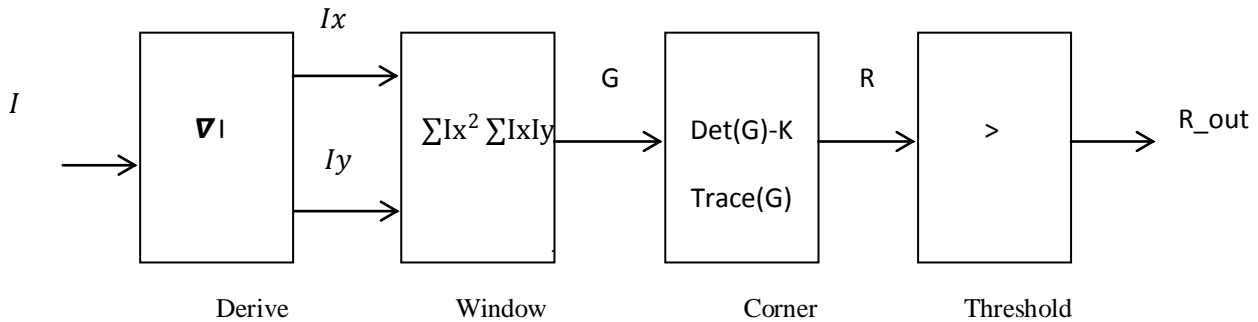


Figure 4 Block Diagram of Corner\_Top Module

Before performing the corner detection, a simple box averaging algorithm using Gaussian window was used in order to smooth the image. This algorithm took the average of the entire 3x3 area. The smoothing operation was computed for the derivatives mentioned above. Once the smooth derivatives were found, the matrix used for the Corner detection was constructed.

### 3.1 Convolution Operation

Image processing algorithms requires a simple mathematical operation convolution, which is frequently used in image processing applications such as smoothing, sharpening and edge detection. Convolution is known as spatial filters and use wide variety of convolution masks known as kernels in order to obtain different results depending on the function desired. In an image processing system, the convolution operation produces output pixel values which are a linear combination of certain input pixel values of the image [22]. The color image mosaicing algorithm proposed in this paper uses 3x3 pixel window, where a block of nine pixels of original image is convolved with Gaussian kernel of size 3x3. The two dimension (2D) Gaussian function is defined by Eqn. (4):

$$g(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (4)$$

where  $\sigma$  is the standard deviation of the distribution;  $x$  and  $y$  are spatial coordinates.

The Gaussian convolution matrix is given by Eqn. (5).

$$G(x,y) = g(x,y) \times I(x,y) \quad (5)$$

Where  $g(x,y)$  is a Gaussian kernel,  $I(x,y)$  is the original image, and  $G(x,y)$  is the convolved output.

Mathematically, 2D convolution can be represented by the following Eqn. (6):

$$G(x,y) = \sum_{i=1}^m \sum_{j=1}^n I(i,j) \times g(x-i,y-j) \quad (6)$$

where  $m = 3$  and  $n = 3$  for 3x3 Gaussian kernel. Convolution operation leads to enormous number of computations resulting in unacceptably long execution times in general purpose processors including DSPs. In contrast to this, FPGA implementation would be an ideal choice owing to massive parallelism and high pipelining that can be built into the design [16]. The input pixel window may still be chosen to be of the same size as the convolution mask without affecting the throughput. The output pixel is rounded to the nearest integer value. Commonly used convolution masks are 3x3, 5x5, 7x7 and 9x9 [22]. In this work, the convolution mask is chosen as 3x3 in order to enhance the implementation speed, at the same time minimizing blocking artifacts. The convolution mask of 3x3 implemented in hardware is shown in Figure 5.

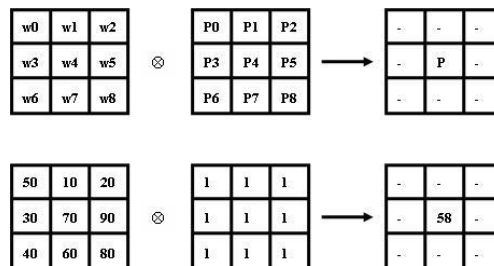
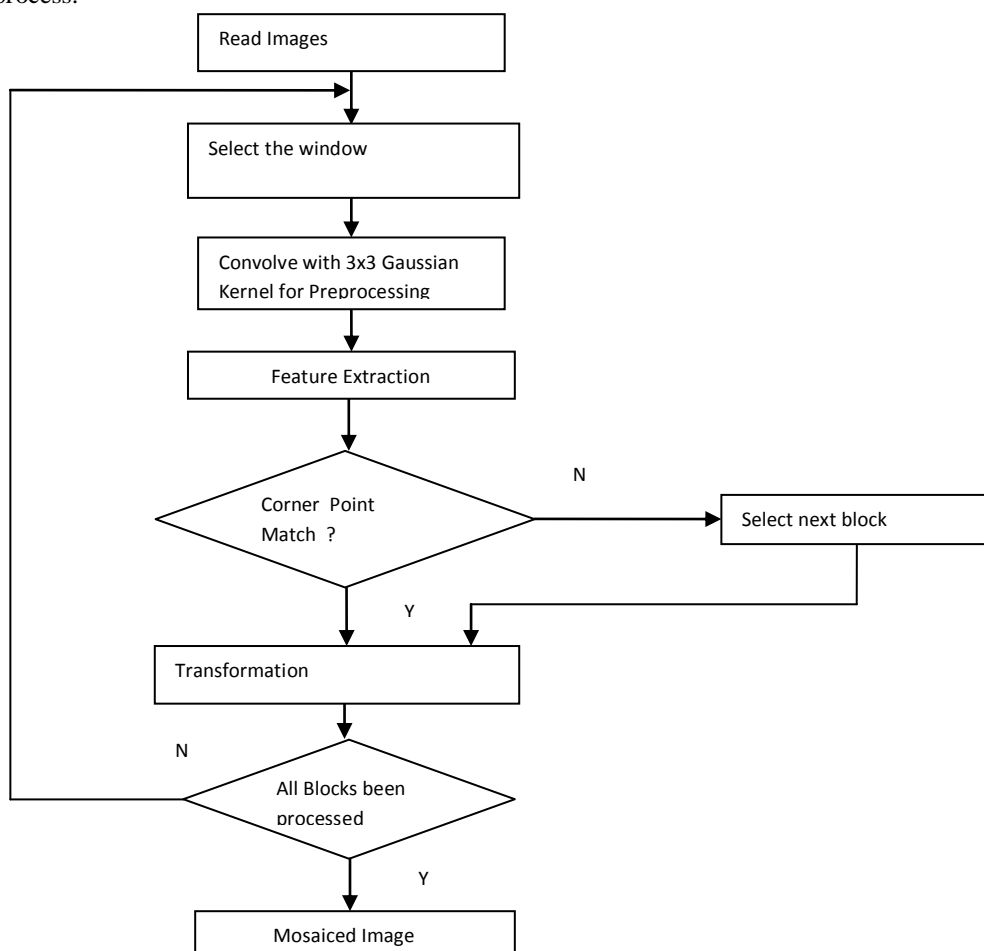


Figure 5 Two Dimensional Convolution Scheme

#### IV. Architecture of the Proposed Color Image Mosaicing System

Mosaic images must be interpolated to generate a three-color RGB values at each pixel. Methods to interpolate the missing color values, like simple nearest neighbor replication or bilinear interpolation can cause severe aliasing and false colors intensity variation near edges or corners. The Harris corner detector algorithm contains large amount of addition and subtraction computations [26-28]. The noise in the image is less than most of the other algorithms. In this Harris corner method, detection of corner is not feasible for variation of illumination in images. Scale Invariant Feature Transform (SIFT) technique for image mosaicing is used for translation, rotation, zoom and illumination changes [29]. Image mosaicing based on Image fusion using Harris is very limited in an application requiring a large amount of overlap area between the images to be mosaiced [30, 31]. However, these methods are computationally intensive. In each stage, the computations are different, and can be overcome by feature based block wise corner detection algorithm.

Figure 6 presents the flowchart for the proposed block wise corner detection algorithm. The input images are first acquired by the camera and stored in a memory. Thereafter, the pre-processing step is applied using Gaussian filter in order to improve the accuracy and its performance. As mentioned earlier, the image is accessed as 3x3 pixel blocks successively. Once features have been extracted from all the blocks of the images, they must be matched to get similar corner points. If the corner point of one image is matched with another image, then the images can be mosaiced using transformation method. Otherwise, the corner point match is searched in the next block. The above process is repeated for all the blocks of the images in order to complete the mosaicing process.



**Figure 6** Flow chart for the proposed method

The objective of the research work is to develop hardware architecture for the Feature based block wise corner detection algorithm. It is quite often that the digital images are corrupted by salt and pepper noise or image noise during image acquisition or transmission. The noise has to be removed, in order to improve the images for better image quality in image processing system applications. This algorithm involves complex computations such as window related operations, pixel sorting, median selection etc. Therefore, the algorithm is converted into architecture suitable for FPGA/ASIC implementations. This is evident from the Matlab simulation of the algorithm presented [32]. The proposed image mosaicing is computationally complex.

Therefore, it is necessary to convert the algorithm into architecture suitable for FPGA/ASIC realization in order to improve the throughput. The most important purpose of the research work is to develop the image mosaicing technique that offers satisfactory results. It is intended to use a Gaussian kernel of size 3×3 for the image mosaicing since the function is a point spread which smoothes the image. The intensity value of the corner point within a block is computed using window function  $w(x, y)$ . The blocks are pre-processed using Gaussian filter to improve their quality. The proposed block wise corner detection method is superior to Scale Invariant Feature Transform, Harris algorithm and image fusion using Harris method in terms of the computational speed.

The algorithm developed to process the color image mosaicing is as follows:

- Step1: Read the 2 input color images.
- Step2: The images are stored in memory.
- Step3: Select the window size as 3x3 for each of the images.
- Step4: Select a window in each image and convolve with Gaussian Kernel to pre-process the images in order to improve the accuracy and its performance.
- Step5: Convolve image with horizontal and vertical differential operator to obtain gradients  $g_x$  and  $g_y$ .
- Step6: Generate the three summations necessary from  $I_x$  and  $I_y$ .
- Step7: Compute the determinant and trace to come up with a value for the likelihood that the current pixel is a corner.
- Step8: Compare with a threshold to determine if it is a corner. If it is, output a “1” attached at the end of the 8-bit pixel value written to memory, so that it can read the pixel value as a corner.
- Step9: Extract features from all images that must be matched to get similar corner points.
- Step10: Match the corner point of one image with another image.
- Step11: Construct the transformation matrix for the image.
- Step12: The above process is repeated for all the blocks of the images.
- Step13: Finally display the mosaiced image.

#### 4.1 Architecture of Derive Pixels

The architecture of Derive pixels for an image is shown in Figure 7. This module computes the x and y gradients for a given pixel and requires 8 surrounding pixels. The detailed signal descriptions of this block are presented in Table 1. The inputs  $p_0, p_1, p_2, p_3, p_5, p_6, p_7, p_8$  are pixels, each of size 8-bits. During positive edge of the clock, the input pixels are valid in the module. This module outputs gradients “abs\_gx” and “abs\_gy” each of size 8 bits for the horizontal and vertical directions of an image. This module outputs the absolute value of each of the x and y gradients.

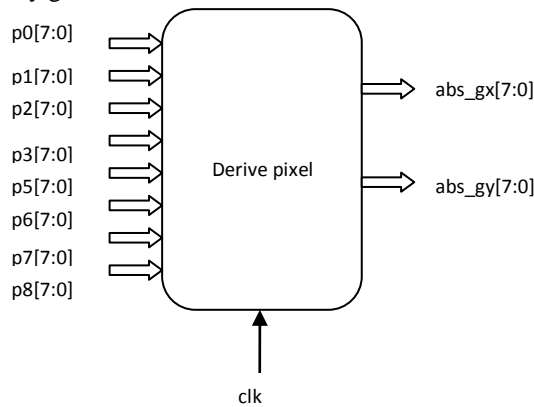


Figure 7 Architecture of Derive Pixel

Table 1 Signal Description for the Derive Pixel

Signals	Description
clk	System clock
p0[7:0]	Pixel p0 value size of 8 bits
p1[7:0]	Pixel p1 value size of 8 bits
p2[7:0]	Pixel p2 value size of 8 bits
p3[7:0]	Pixel p3 value size of 8 bits
p5[7:0]	Pixel p5 value size of 8 bits
p6[7:0]	Pixel p6 value size of 8 bits

p7[7:0]	Pixel p7 value size of 8 bits
p8[7:0]	Pixel p8 value size of 8 bits
abs_gx [7:0]	Output gradient for an image
abs_gy [7:0]	Output gradient for an image

### 4.2 Corner Module

This module computes the determinant and trace of the corner matrix and outputs the determinant –  $k * \text{trace}^2$ .  $K$  is a constant value of  $1/8$  needed for this calculation. Also, because this is the most computationally intensive calculation owing to the width of each bus, the module carries out the calculation in a pipelined manner using 3 stages. The pipelined calculation of  $R$  is shown in Figure 8. The architecture of the corner module is shown in Figure 9 and its descriptions are presented in Table 2.

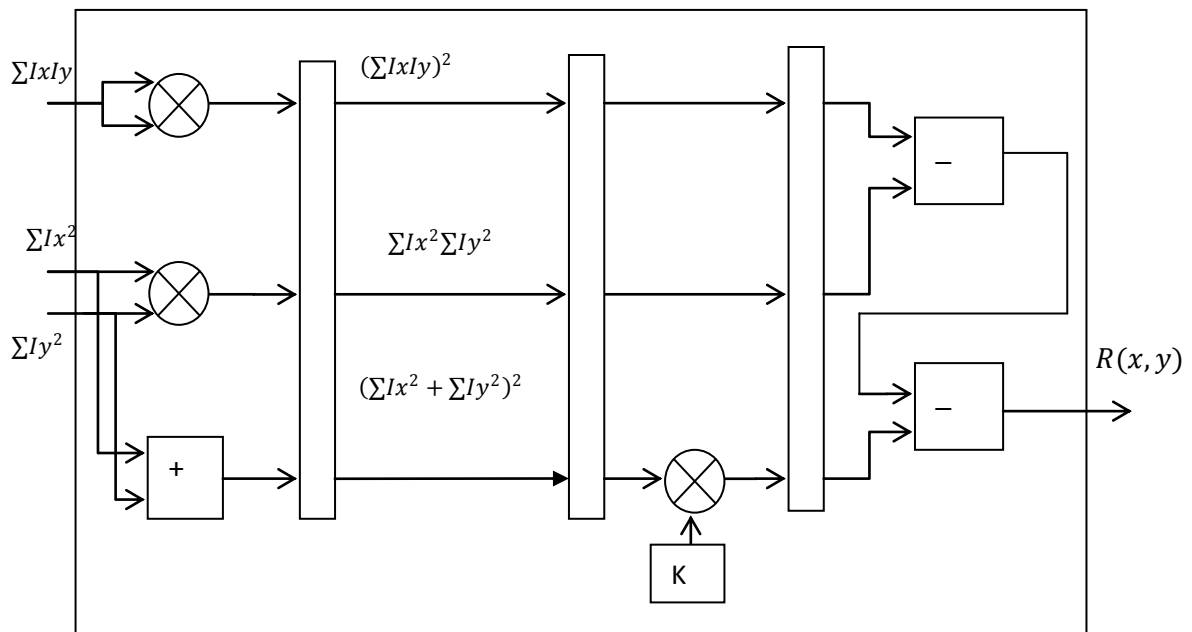


Figure 8 Pipelined Corner module

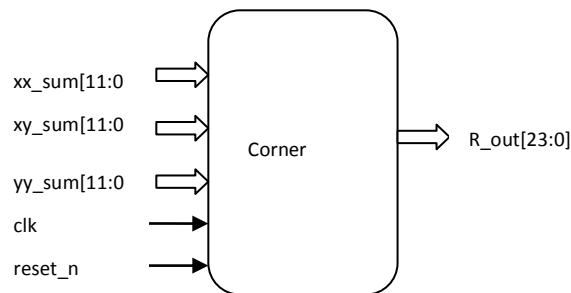


Figure 9 Architecture of Corner Module

Table 2 Signal Description for the Corner Module

Signals	Description
clk	System clock
xx_sum [11:0]	First operand $I_x^2$ for the corner module
xy_sum [11:0]	Second operand $I_x I_y$ for the corner module
yy_sum [11:0]	Third operand $I_y^2$ for the corner module
reset_n	Active low system reset
R_out[23:0]	Corner response

### 4.3 Architecture of Adder

The architecture of the adder is shown in Figure10 and its signal descriptions are presented in Table 3. This architecture has been modeled based on Verilog code of two Adders presented in one of the authors' book [16]. The adder module used in this work performs addition of n1 and n2, each of width 12 bits. The output from this module is of 13 bits width. It uses lots of parallelism and pipelining in its design as can be easily inferred. The unsigned addition has been realized with n1, n2 numbers with five pipeline stages. The unsigned adder was developed specially for use in Gaussian based Image Mosaicing applications.

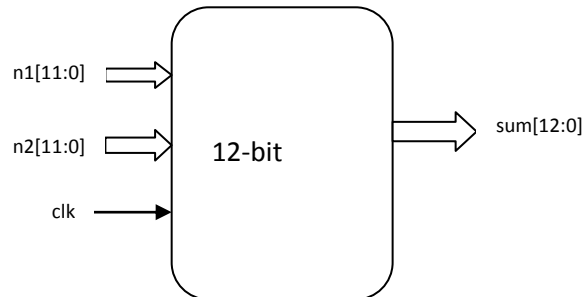


Figure 10 Architecture of 12-bit Adder

Table 4 Signal Description for the 12-bit Adder Module

Signals	Description
clk	System clock
n1 [11:0]	First operand for the adder module
n2 [11:0]	Second operand for the adder module
sum [12:0]	Output of the adder module

### 4.4 Architecture of Subtractor

Figure 11 shows the architecture of 24 bit subtractor and its descriptions are presented in Table 5. The subtractor module used in this work performs difference of n0 and n1, each of width 24 bits. The output from this module is of 24 bits width.

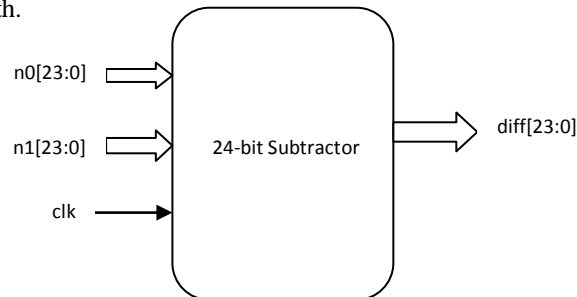


Figure 11 Architecture of 24-bit Subtractor module

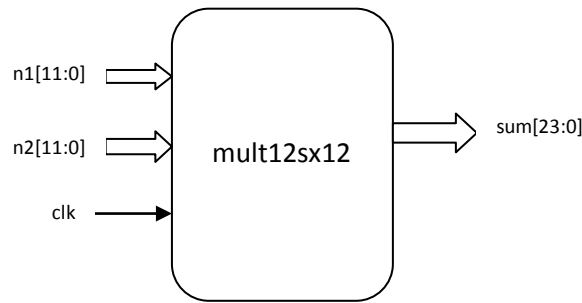
Table 5 Signal Description for the Subtractor module

Signals	Description
clk	System clock
n0 [23:0]	First operand for the Subtractor module
n1 [23:0]	Second operand for the Subtractor module
diff [23:0]	Output of the Subtractor module

### 4.5 Architecture of Multiplier

The multiplier design presented here incorporates a high degree of parallel circuits and pipelining of eight levels [16]. The architecture of a 12-bit multiplier block is shown in Figure 12, performs a multiplication of two unsigned numbers n1 and n2 each of 12-bits size. Table 6 gives the detail signal description for the 12-bit multiplier module. The result is of 24-bits width. In addition, the architecture utilizes many pipelined registers internally. Eight pipelined stage are exploited in order to increase the processing speed.





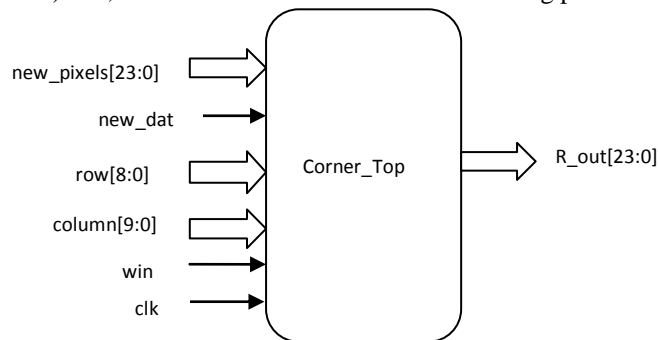
**Figure 12** Architecture of 12-bit Multiplier

**Table 6** Signal Description for the Multiplier Module

Signals	Description
clk	System clock
n1 [11:0]	First unsigned operand for the multiplier module
n2 [11:0]	Second unsigned operand for the multiplier module
result [23:0]	Output product of the multiplier module

#### 4.6 Architecture of Corner\_Top module

The “Corner\_Top” module manages the local 3 x 3 window of pixels that is loaded from memory. It also takes the row and column information for the left-most pixel in the pixel cluster. The row and column information will provide the location information for those four pixels, so that they can be later displayed on the screen in the proper location. The architecture of the “Corner\_Top” module is shown in Figure 13 and its descriptions are presented in Table 7. On the arrival of “new\_data” signal, the following 3 x 3 window of pixels is populated. Since the module can only take three pixels at a time, the win flag is used to determine whether the pixels correspond to columns 1-2 (win = 0) or column 3 (win = 1). The window is stored in a register array and the reason it is 3 x 3 is that the module can output at most three results (since it can only write 18 bits to memory and each pixel needs 8 bits) and, each of the corner needs the surrounding pixels for gradient calculations.



**Figure 13** Architecture of “Corner\_Top\_module”

The “Corner\_Top” module instantiates 9 “Derive\_pixel” modules in order to calculate 18 gradients per window (9 in the x-direction (gx) and 9 in the y-direction (gy)). “window\_ready” is used for synchronization in all the derive pixel modules. On the rising edge of the clock and once the “window\_ready” signal is high, it calculates xx\_sum, yy\_sum and xy\_sum values. Finally, the output of the Corner\_top module is compared against a threshold and the result is appended in location 4 of the R\_out signal, retaining other pixels of the original image. The threshold value “1” represents that the preceding pixel is a corner pixel, while “0” represents it is not.

**Table 7** Signal Description for the “Corner\_Top\_module”

Signals	Description
clk	System clock
new_pixels[23:0]	New pixel values
row [8:0]	Row values
columns[9:0]	Column values
new_data	New data for corner module
win	Gaussian window value
R_out [23:0]	Output Threshold value

### V. Results and Discussions

The proposed Corner detection based Mosaicing using feature based block wise method, whose architecture was presented in the previous section, has been coded and tested in Matlab (Version 8.0 R2012b) first in order to ensure the accurate functioning of the algorithm. Subsequently, the complete system has been coded in RTL compliant Verilog HDL so that it may be implemented on an FPGA or ASIC. The simulation of the feature based block wise method has been done using ModelSim and synthesized using Xilinx ISE 13.2. The algorithm has been implemented on Xilinx Spartan6 xc6slx45-3fgg676 FPGA device. The functional modules comprising of derive pixel, corner module, corner\_top module, multipliers, subtractors and adders were simulated using ModelSim and the waveforms are presented in Figure 14 to Figure 20. The Xilinx generated RTL schematic view of top module is shown in Figure 21.

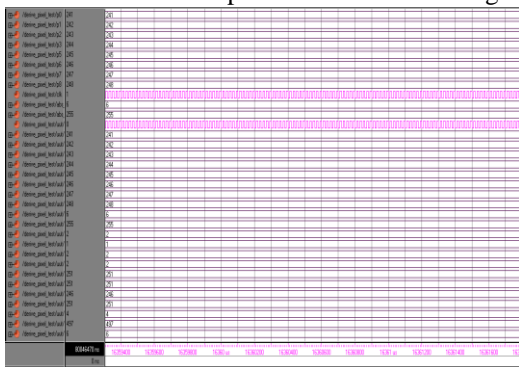


Figure 14 Simulation for “Derive Pixel module”

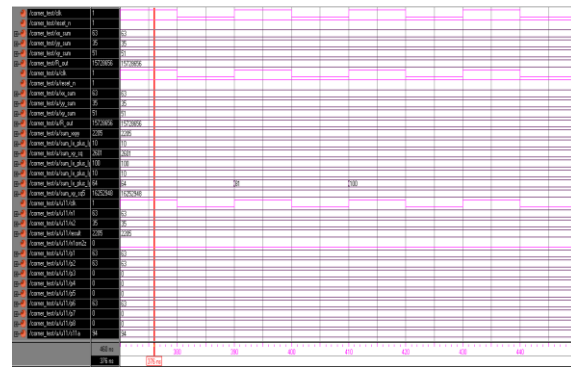


Figure 15 Simulation for “corner module”

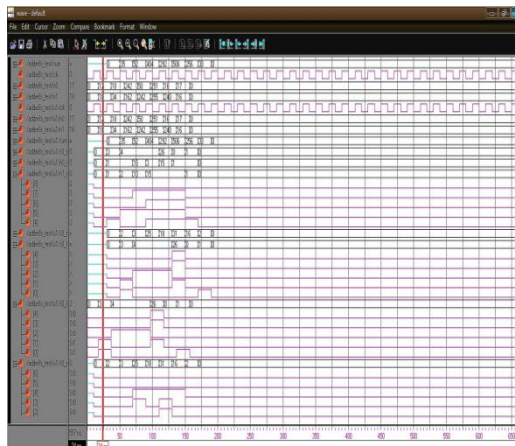


Figure 16 Simulation waveforms for Adder

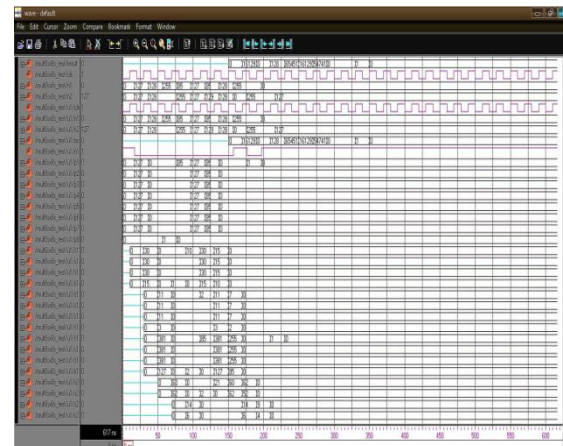


Figure 17 Simulation waveforms for Multiplier



Figure 18 Simulation for Subtractor

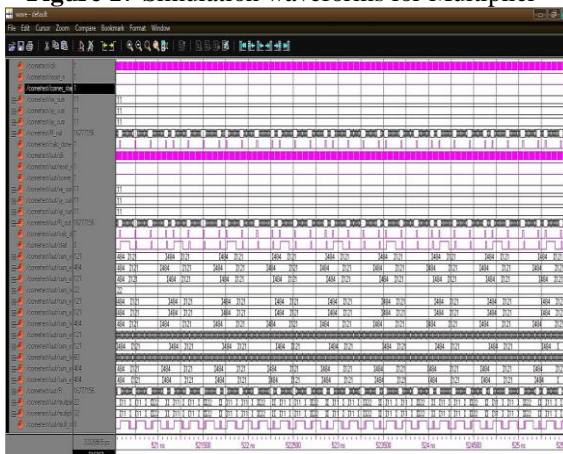


Figure 19 Simulation for “corner\_top module”

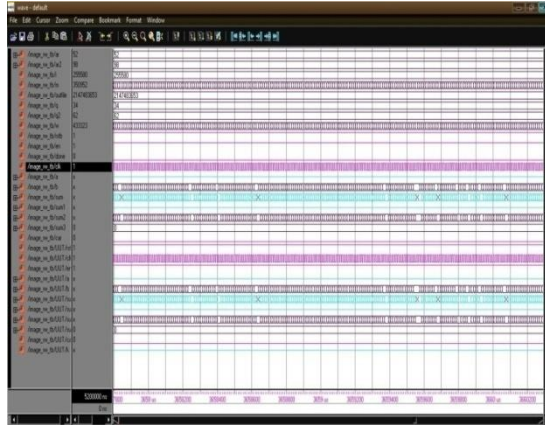


Figure 20 Simulation for Mosaicing Images

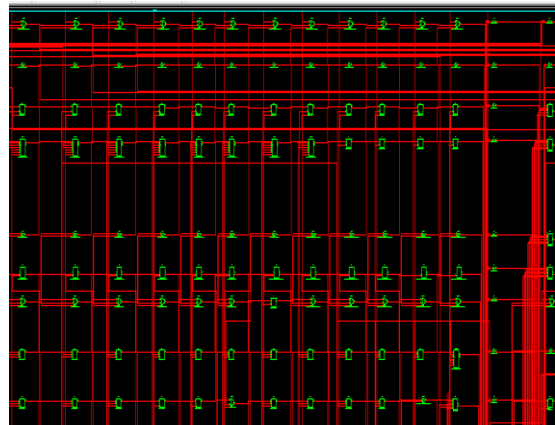


Figure 21 RTL for Mosaic images

When compared to Harris and Fusion methods, the proposed algorithm gives better results in terms of Power Signal to Noise Ratio (PSNR). The PSNR value of the mosaiced image is computed using equation (5).

$$PSNR = 20 \log_{10} \left( \frac{\text{Max}_f}{\sqrt{\text{MSE}}} \right) \quad (5)$$

where the MSE (Mean Square Error) is given by equation (6):

$$MSE = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i,j) - K(i,j)]^2 \quad (6)$$

where I is the matrix data of original image while K represents the mosaiced image using the proposed block wise method, m represents the number of rows of pixels of the images and i represents the index of that row, n represents the number of columns of pixels of the image and j represents the index of that column and Max<sub>f</sub> is the maximum signal value of the proposed image method. The MSE represents the average of the squares of the errors between the mosaiced image and other existing methods.

Figure 22 and 23 shows the input images. These images are mosaiced using the proposed Corner Detection Method. The mosaiced image is shown in Figure 24. The mosaiced image of the proposed block based corner detection method is superior to other methods in terms of the quality (PSNR).



Figure 22 Left Image1



Figure 23 Right Image2



**Figure 24** Mosaic Image Using the Proposed Corner Detection Method

The proposed algorithm is compared with Harris and Fused methods in Table 8 and gives better results in terms of PSNR.

**Table 7** Performance Analysis

Parameters	Proposed method	Fused method	Harris method
PSNR (dB)	42.9	41.1	39.9

The proposed work is the development of architectures and realization of the image mosaicing techniques suitable for FPGA/ASIC realization. The visual quality of the reconstructed pictures and image mosaicing achieved for various test images using wavelet energy metric were compared. The Verilog design of the feature block based algorithm system was targeted on Xilinx Spartan6 xc6slx45-3fgg676 FPGA device. The system utilizes about 12,198 slices as reported by the Integrated Software Environment tool. The maximum frequency of operation reported is 292 MHz.

## VI. Conclusions

The design of a novel algorithm and architecture for Feature based color image mosaicing for real time applications has been developed. A new algorithm has been proposed in this paper to compute corners of the image signals. The proposed algorithm has been coded in MATLAB and Verilog. Harris method is good at its angular point feature extraction, but is ineffective at matching effect of image rotation scale variation. SIFT method is slow, but it has the advantage of affine, rotating, scale invariable. This paper proposed a new method for automatic generation of mosaics using block based corner detection algorithm. The proposed method is superior to SIFT, Harris algorithm and image fusion using Harris and SURF method in terms of speed and quality. The proposed method yields better results than existing algorithms. The algorithm can be implemented on FPGA for high throughputs. The algorithm has been designed into architecture suitable for ASIC/FPGA implementation.

## References

- [1]. Marko Heikkila and Matti Pietikainen, "An Image Mosaicing module for Wide-Area Surveillance", VSSN'05, November 11, 2005, Singapore, ACM 1-59593-242-9/05/0011.
- [2]. B. Zitova and J. Flusser, "Image registration methods: a survey", Image and vision Computing, (21):977-1000, 2003, doi:10.1016/S0262-8856(03)00137-9.
- [3]. Nabeel Shirazi, M. Athansa and A.Lynn abbott, "Implementation of a 2-D Fast Fourier Trasform on FPGA based Custom Computing Machine", Proceedings of 12<sup>th</sup> Reconfigurable Architecture Workshop, Denver 2005.
- [4]. M. Brown and D. Lowe, "Automatic panoramic image stitching using invariant features", Intl. J. of Computer Vision, pp. 59-73, 2007.
- [5]. Alistair J Fitch, "Fast Robust Correlation", IEEE Transactions on Image Processing, Vol. 14, No. 8, August 2005.
- [6]. Y D Zhang, L. N. Wu Wang, "Improved Walsh images interpolation method", Computer Engineering & Applications 2011, 47(9): 156-159.
- [7]. P. Azzari, L. D. Stefano, M. Stefano, "An Evaluation Methodology For Image Mosaicing Algorithms", Advanced Concept for Intelligent Vision Systems, Springer, Vol. 52-59, pp. 89-100, 2008.
- [8]. Jain, D. K.; Saxena G., Singh, V. K., "Image Mosaicing Using Corner Technique", International Conference on Communication System and Network Technologies, pp. 79-84, 2012.
- [9]. Abdualah. M Alsuwailem, and Saleh. A. Alshebeili, "A New Approach for Real-Time Histogram Equalization using FPGA", Proceedings of International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS 2005), pp. 397-400, 13-16 Dec., 2005.
- [10]. Donald G. Bailey, "Design for Embedded Image Processing on FPGAs", John Wiley and Sons Inc., 2011.

- [11]. S. Cadambi, J. Weener, S.C. Goldstein, H. Schmit, D.E. Thomas, "Managing Pipeline-Reconfigurable FPGAs", Proceedings of the 1998 ACM/SIGDA 6th international symposium on Field Programmable GateArrays, pp. 55-64.
- [12]. Yiran Li, "FPGA Implementation for Image Processing Algorithms" EEL 6562 Course Project Report, December 2006.
- [13]. Hongyan Wen1 2 Jianzhong Zhou1, "An Improved Algorithm for Image Mosaic" 2008 International Symposium on Information Science and Engineering 978-0-7695-3494-7/08 2008 IEEE.
- [14]. Ryo Yonemoto, et al., "Panoramic Image Generation based on FFT Technique and its Hardware Realization", SICE- ICASE International Joint Conference 2006 Oct. 18-21, Bexco, Busan, Korea.
- [15]. Sen Ma. et al., "Design of Panoramic Mosaic Camera Based on FPGA Using Optimal Mosaic Algorithm", JOURNAL OF COMPUTERS, VOL. 6, NO. 7, JULY 2011
- [16]. S. Ramachandran, "Digital VLSI Systems design: A Design Manual for Implementation of Projects on FPGAs and ASICs using Verilog", Springer Verlag, 2007.
- [17]. S.M. Smith, J.M. Brady, "SUSAN – a new approach to low level image processing", International Journal of Computer Vision, Vol. 23(1), pp. 45-78, May 1997.
- [18]. Mark S. Nixon, Alberto S. Aguado, Feature Extraction and Image Processing, 2<sup>nd</sup> edition, Elsevier, 2008.
- [19]. Somani Patnaik, Jon Losh ,Dember Giraldez, " REAL-TIME FEATURE DETECTION", Final Project, 2009.
- [20]. Hemalatha Joshi and Khomlalsinha, " Novel techniques Image Mosaicing based on Image FusionUsingHarris and Surf", International conference on computer science and information Technology, March 2013, Hyderabad, ISBN:978- 93-82208-70-9.
- [21]. Jubiao Li, Junping Du, "Study on Panoramic Image Stitching Algorithm", Second Pacific Asia Conference on Circuits, Communications and System, Vol. 1, pp. 417-420, 2010.
- [22]. M. C Hanumantharaju Raju et al., "Design of Novel Algorithm and Architecture for Gaussian Based Color Image Enhancement System for Real Time Applications", Computer Vision and Pattern Recognition, [10.1007/978-3-642-36321-4\\_56](https://doi.org/10.1007/978-3-642-36321-4_56), 2014.
- [23]. R. Szeliski and S. Kang, "Direct Methods for Visual Scene Reconstruction", IEEE Workshop on Representations of Visual Scenes, pp. 26–33, Cambridge, MA, 1995.
- [24]. Bay H. Surf, "Speeded up Robust Features", Proceedings of the 9<sup>th</sup> European Conf. on Computer Vision, IEEE Press, 2006.
- [25]. C. G. Harris, M. J. Stephens, "Combined Corner and Edge Detector", Proceedings of the 4th Alvey Vision Conference, Manchester, pp. 147-151, 1988.
- [26]. L. Kitchen, A. Rosenfeld, "Gray level corner detection", Pattern Recognition Letters, pp. 95-102, 1982
- [27]. P. Tissainayagam, D. Suter, "Assessing the Performance of Corner Detectors for Point Feature Tracking Applications", Elsevier Image and Vision Computing, Vol. 22, Issue 8, pp. 663-679, 2004.
- [28]. Zhiqian.Y & Hao.W et al., "An Image Mosaic Algorithm of Pathological section based on feature points" International Conference on Information Engineering and Computer Science, 2009.
- [29]. D. G. Lowe, "Object Recognition from Local Scale-Invariant Features", Proc. of the International Conference on Computer Vision, pp. 1150 -1157, Sept. 1999.
- [30]. D. Ghosh, S. Park, N. Kaabouch, W. Semke, "Quantum Evaluation of Image Mosaicing in Multiple Scene Categories", IEEE Conference on Electro Information Technology, pp. 1-6, 2012.
- [31]. Brunet.Florent; &Gay Bellile.Vincet;et al."Feature Driven Direct Non Rigid Image Registration." International Journal of Computer Vision, 2011, vol. 93, pp . 33-52.
- [32]. R. Gonzales, P. Wintz, Digital Image Processing, Addison-Wesley,1987.
- [33]. A. Bovik, "Handbook of Image and Video Processing", 2<sup>nd</sup> edition, Elsevier Academic Press, 2005.