Spring 2003

A Pipelined Architecture for Real Time Correction of Barrel Distortion in Wide-Angle Camera Images

Hau Trung Ngo
Old Dominion University

Recommended Citation
Ngo, Hau T.. "A Pipelined Architecture for Real Time Correction of Barrel Distortion in Wide-Angle Camera Images" (2003). Master of Science (MS), Thesis, Electrical & Computer Engineering, Old Dominion University, DOI: 10.25777/v4st-4s68
https://digitalcommons.odu.edu/ece_etds/452

A PIPELINED ARCHITECTURE FOR REAL TIME CORRECTION OF BARREL DISTORTION IN WIDE-ANGLE CAMERA IMAGES

by

Hau Trung Ngo
B.S. May 2001, Old Dominion University

A Thesis Submitted to the Faculty of
Old Dominion University in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

COMPUTER ENGINEERING

OLD DOMINION UNIVERSITY
May 2003

Approved by:

[illegible]
Vijayan K. Asari (Director)

[illegible]
James F. Leathrum, Jr. (Member)

Min Song (Member)

ABSTRACT

A PIPELINED ARCHITECTURE FOR REAL TIME CORRECTION OF BARREL DISTORTION IN WIDE-ANGLE CAMERA IMAGES

Hau Trung Ngo
Old Dominion University, 2003
Director: Dr. Vijayan K. Asari

Images captured by wide-angle cameras show barrel type spatial distortion due to wide-angle configuration of the camera lens where image regions farther from the center are compressed in a nonlinear fashion. The barrel distortion correction technique based on least squares estimation attempts to correct a distorted image by expanding it nonlinearly so that straight lines in the object space remain straight in the image space. The intensity value of a pixel in the expanded image is calculated by back mapping its position to the corresponding position in the distorted image and performing linear interpolation on the neighboring pixels. Distorted images are analyzed to calculate the expansion coefficients and back mapping coefficients of the forward-mapping and the back-mapping polynomials in the distortion correction algorithm. These values are unique for each camera.

Demands for faster correction of distorted images in various real time applications lead to the development of a dedicated hardware architecture that can provide a high throughput rate. An absolute pipelined architecture is designed to correct barrel distortion in images by partitioning the distortion correction algorithm into four main modules. The architecture includes a CORDIC based rectangular to polar coordinate transformation module, a back mapping module for nonlinear transformation of corrected image space to distorted image space, a CORDIC based polar to rectangular coordinate

transformation module, and a linear interpolation module to calculate the intensity of a pixel in the corrected image space. The system parameters include the expanded/corrected image size, distorted image size, the back mapping coefficients, distortion center and the center of the corrected image. The hardware design is suitable for correcting 8-bit images of size up to four-million pixels. It can sustain a high throughput rate of 30 frames per second. The pipelined architecture design will facilitate the use of dedicated hardware that can be mounted along with the camera unit.

ACKNOWLEDGEMENTS

I would like to thank Dr. K. Vijayan Asari for his guidance and efforts as my thesis director. I am also very grateful for his encouragement throughout the years. I would also like to thank Dr. James F. Leathrum Jr. and Dr. Min Song for their time and consideration in serving on my thesis committee. I greatly appreciate their efforts. Finally, I would like to thank my family for their tremendous support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

A wide-angle camera can capture a large area of view into a single image, which is very useful for many applications in navigation, surveillance and medical fields. But it also introduces barrel distortion in images where images are compressed nonlinearly. Barrel distortion shows nonlinear compression in the outer areas of the image where objects appear smaller than their actual size. Figure 1.1b shows a typical distorted image of an experimental grid as shown in Figure 1.1a. The wide-angle configuration of the lens causes the images to appear spherical as noted in Figure 1.1b. Many applications rely on the capability of a wide-angle camera to provide a broad view of the targeted objects or areas, but they also demand accurate geometrical information since the correctness of such information is very important for the quantitative parameters estimation. An example of such application is the surgical procedure that requires the assistance of endoscope which uses wide-angle lens to provide a large view of internal structure of the gastrointestinal tract. Thus, images with barrel distortion must be corrected before they are presented to the display unit.

The approach to correct barrel distortion in wide angle camera images studied in this research is to analyze the curvature of the lines in a distorted image which is captured

Format of this thesis is *IEEE Transactions on Computers*

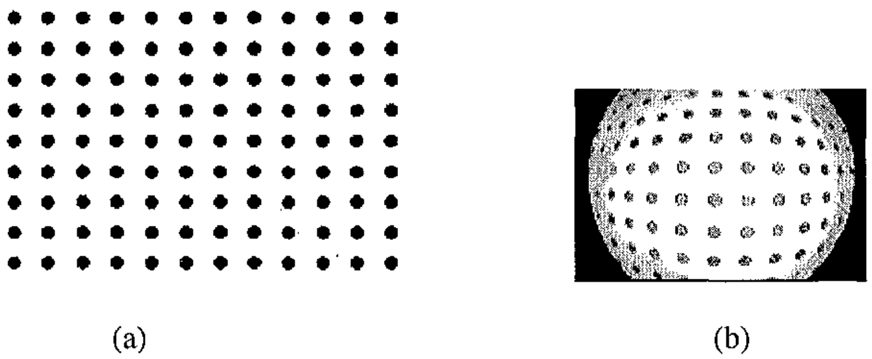


Figure 1.1: (a) Experimental grid; (b) Grid image captured by a wide-angle camera.

from a grid of straight lines. The analysis of the distortion would produce a set of estimated correction parameters to straighten the curved lines in the distorted image. These parameters are used to correct the barrel distortion in any images taken by the same camera. The first objective of this research is to perform the analysis based on a procedure of straightening the curved lines in the distorted image and estimating the correction parameters.

The correction parameters include a set of expansion coefficients and a set of back-mapping coefficients. The method of selecting expansion coefficients is based on least squares estimation which is a technique used to select the parameters of which the sum of the squared errors of the curved lines with respect to the expected values is minimized to produce straight lines. The expansion coefficients are used in an expansion polynomial to map a distorted image space to a corrected image space. The back mapping coefficients are used to convert the position of a pixel position in the corrected image to a corresponding position in the distorted image based on a back mapping polynomial. Back-mapping coefficients needed in the back-mapping procedure are obtained based on non-linear regression analysis on a finite number of points. The distortion correction method involves three main steps: (1) all pixels in the distorted image are forward-mapped onto the corrected image based on an expansion polynomial; (2) all vacant pixels in corrected image are back-mapped onto the distorted image based on a back-mapping polynomial; and (3) the intensity for each vacant pixel in corrected image is obtained by performing linear interpolation of the intensities of the four neighboring pixels in the distorted image.

In general, barrel distortion in wide-angle camera images can be corrected by many commercial software tools available in market today. However, they do not provide high-speed solutions that are often required for real time applications. General purpose image processors can also be used to correct barrel distorted images but they too have limitations in terms of performance. Real time distortion correction in images requires the processors to process at least 25 frames per second. A typical image size for a wide-angle camera is about 1 million pixels. Therefore, the distortion correction processor must be able to correct at least 25 million pixels per second. The image size in wide-angle cameras increase as technology advances, so the rate of distortion correction should also increase. To achieve a high-speed throughput in a distortion correction system with minimal overhead, dedicated hardware architecture design is necessary. The second objective of this research is to design an architecture that is capable of correcting barrel distortion in wide-angle camera images with a minimal throughput rate of 25 million pixels per second.

Image processing is one of the data driven problems where exploiting the maximum parallelism in data is desirable due to the repetitive nature in image processing application. Thus, a correction architecture must incorporate data parallelism into the design because time is the main criteria in real time correction of distorted images. Indeed, most of the dedicated architectures proposed in literature for image processing applications do address the parallel nature since they usually employ architecture design methodologies such as systolic array or pipelined architecture. A pipelined architecture exploits the data parallelism available in a given application by allowing overlapped operations at all stages in the pipeline. Thus, the maximum data parallel level is determined by the number of stages in the pipeline. Another advantage of a pipelined architecture is that it requires simple controlled logic which would reduce the overhead of the controller unit considerably.

In this thesis, a dedicated architecture to correct barrel distortion in wide-angle camera images for real time applications is presented. The entire architecture is pipelined to fulfill the demand for a high throughput correction system in real time applications such as video-endoscopy. The designed architecture is expected to maintain a minimal throughput of 25 million-pixel frames per second and it can be incorporated into a high performance FPGA which supports fast access to RAM modules.

The remainder of this thesis is organized as follows: many different approaches to correct distortion in images and various architectures for image processing are explained in chapter 2. The algorithm to correct barrel distortion in images and its simulation results are discussed in chapter 3. Software development to simulate the distortion correction algorithm and its simulation results are discussed in Section 3.4. The design of the pipelined architecture to correct barrel distortion is presented in chapter 4. Chapter 5 provides the hardware implementation results of the pipelined architecture and the evaluation of architecture performance. Chapter 6 concludes with a summary of the research work performed for this thesis and possible suggestions for future work.

CHAPTER 2

LITERATURE REVIEW

2.1. BARREL DISTORTION AND WIDE-ANGLE CAMERA APPLICATIONS

Due to the wide-angle configuration of the lens in cameras, images taken by these cameras can capture a much larger field of viewing in a single image, but barrel distortion is also introduced to these images. Barrel distortion causes the images to appear spherical (curved outward) where image areas near the distortion center are compressed less while areas farther from the center are compressed more. As a result of barrel distortion, sizes of objects of interest in wide-angle camera images might appear far different than their actual sizes.

Due to the advantage of capturing a large area, wide-angle cameras are used in various fields. A wide-angle camera or fisheye lens camera image is used by Green [1] to create a texture for an environmental map. The wide-angle configured camera permitted the author to capture the entire area of interest without moving the camera as compared to the method of using the standard camera mounted on a motor that has to pan back and forth throughout the area. Wide-angle cameras are also used in applications that require compact image systems due to limited space. Karass et al. [2] has mentioned the use of wide-angle cameras in architectural and archaeological applications where the available spaces for imaging systems in places such as tombs, corridors, passages are very limited. Wide-angle cameras are also frequently used in the medical field to assist physicians in observation, diagnostic procedures, and surgical procedures. A visual assisting method based on endoscopes has been proposed tract to assist surgeons in laparoscopic surgery [3]. Endoscopes facilitate observation, documentation and electrical manipulation of the images of internal structure of the gastrointestinal. For applications where quantitative parameters such as area and perimeter are very important, as in clinical endoscopes, estimation errors due to distortion could cause grave concerns. The distortion in images must then be corrected [4][5].

2.2. DISTORTION CORRECTION METHODS

Distortion correction by lenses is possible for many wide-angle cameras. However, it is not possible for cameras used in confined spaces such as video endoscopy because the size of these cameras is very small. Calibration techniques to correct the distortion in images, methods to estimate the distortion based on careful observation and comparison and mathematical models to estimate parameters for distortion correction algorithms were proposed by many researchers. Some of the proposals made to correct distortion in images are discussed in this section.

2.2.1. Camera Calibration

Tsai [6] proposed a technique using a radial lens distortion model that describes a two-dimensional image to correct distortion. This technique relies on the 3×3 rotation matrix and 3×1 translation vector. Thus, the operations involved in this method are complicated. A special technique for frequency measurement is also required for calibration of the one-pixel width, which is added to the complication of this correction method. Nomura et al. [7] presented a calibration technique for high distortion TV camera lenses. The calibrated parameters obtained from this proposed method are effective focal length, one-pixel width on an image plane, distortion center in the distorted image, and distortion coefficients. This method requires careful and precise placement of the calibration chart that consists of parallel straight lines as a reference in the calibration procedure. Thus, a small shift of the chart prompts considerable errors in the obtained calibrated parameters.

Weng et al. [8] explained radial and thin prism types of distortions and techniques to model them mathematically. In this technique, the calibration parameters are first estimated by using close-form solution based on a distribution-free camera. They are then recalculated by an iterative procedure that utilizes the nonlinear optimization based on different types of distortion which are present in any particular camera. Each of the three models described above give reasonable results for images obtained from cameras with normal viewing objective lens but these models are not effective for cameras with wide-angle lens.

Another method to estimate the distortion in endoscopes was proposed by Classen et al. [9]. In this method, a known-size rubber disc is placed in an ulcer base for the purpose of comparing the size of the disc captured by endoscope to the size of the ulcer. A similar and improved method was presented by Okabe et al. [10], which permitted a better estimation method. Both of these techniques require careful placement of a rubber disc of known size into the center of the ulcer base.

2.2.2. Distortion Correction

Smith et al. [11] proposed a mathematical model to estimate coefficients in order to correct distortion in endoscopic images. This method estimates the coefficients based on a set of orthogonal Chebyshev polynomials. Each pixel in the corrected image is mapped onto the distorted image for its intensity, which is the nearest intensity value in the distorted image. The mapping procedure is based on an expansion polynomial involving the expansion coefficients obtained from the set of orthogonal Chebyshev polynomials. Hideaki et al. [12] presented a different method for estimation of the model parameters for the expansion procedure similar to the method proposed by Smith. A set of image points obtained from the distorted grid lines is used to create a moment matrix in the procedure to estimate the expansion coefficients. The expansion coefficients are estimated based on the smallest characteristic root of the moment matrix and the corrected image is obtained from the expansion procedure based on the expansion polynomial.

Another method to estimate parameters for a mathematical model for distortion correction was presented by Asari et al. [13]. This method estimates the coefficients based on least squares estimation. The coefficients are obtained from the best fit polynomials for each distorted column of dots in the distorted image. The detailed explanation of this method is discussed in chapter 3. Helferty et al. [14] proposed another method to obtain model parameters based on the least squares estimation method presented by Asari. The later technique incorporates both columns and rows of dots in the distorted image in the procedure to estimate distortion coefficients. All of the proposed methods that are based on a mathematical model require extensive computations that obviously consume significant time to process each distorted image.

2.3. IMAGE PROCESSING ARCHITECTURES

Most imaging software packages available in the market today provide a tool to correct distortion in images. However, for real time applications that use a wide-angle camera such as laparoscopic surgery in which distortion in video-endoscopy must be corrected, the required amount of time for correcting a sequence of images is very short. Thus, software solutions to correct distortion in wide angle camera images do not fulfil the high performance demand of those real time applications.

Alternatively, general-purpose image processors can also be used to correct distorted images if they are equipped with distortion correction function. However, the required computational power and data throughput for real time applications usually exceed the limitations of general-purpose processors. Another limitation associated with general-purpose processors which is attached to high performance workstations is the data rate on system bus. It usually limits the processors from achieving peak performance [15]. The advantage of a general purpose image processor is the scalability and flexibility it can provide. Ranganathan et al. [16] presented the results of the performance evaluation of general purpose processors in various image processing applications. The results show that the instruction overhead, penalties due to cache misses, and the inability to take advantage of the high level of parallelism in many image processing applications have considerably limited the performance of general purpose image processors.

2.3.1. Dedicated Architectures

On the other hand, dedicated architectures are designed to provide high performance in terms of speed and throughput rate. The draw back of dedicated architectures is that they are not as flexible as general purpose image processors are. For many real time image processing applications, flexibility is not as important as high processing speed. Due to the enormous amount of data to be processed in a short time and the repetitive nature, real time applications in image processing are ideal candidates for dedicated architectures [17]. The advancement in VLSI technology also helps to reduce the physical size of the chip, which is very important for specialized image processors because these processors are to be attached to various mobile systems such as pinhole cameras and robots. Many researchers have designed specialized architectures to process various image processing applications which facilitate high-speed performance. Do and Yun [18] have presented a VLSI architecture for full-search block-matching motion estimation. A VLSI implementation of a focal plane image processor for the realization of the near-sensor image processing concept has been proposed in Eklund et al. [19]. Both architectures in these papers address the demand for high performance and low power consumption processors in image processing applications.

2.3.2. Parallel/Systolic Architectures

Image processing has a natural characteristic of data parallelism. Many identical and independent operations can be applied to all pixels in an image at the same time. Because of this property in image processing, architecture design methods that exploit the maximum parallelism are the most appropriate techniques. Many researchers have proposed image processors that contain identical processing units that can process multiple data simultaneously. One such parallel processor design is the one-dimensional processor array for image processing presented by Hammerstrom and Lulich [20]. Each processing unit is a sub-processor with its own on-chip memory that is capable of performing the specific task for one pixel. An implementation of a systolic array processor for digital image and digital signal processing has been presented by Hemkumar et al. [21]. The processor array consists of multiple CORDIC array processor elements, which can perform vector rotation and inverse tangent calculation. The systolic implementation of Hopfield and Hamming neural networks has been presented by Asari et al. [22]. The proposed neural networks provide massive parallelism for multiple operations. Parallel architectures are appropriate for image processing applications because of the natural data-parallelism in these applications. These architectures also require high bandwidth for I/O operations because they need to access multiple data simultaneously for multiple operations in various processing units.

2.3.3. Pipelined Architectures

Pipelining the architecture is another hardware design technique used to improve throughput of the system by exploiting the operation parallelism. Pipelining is the method to partition the workload of any task into smaller sub-tasks so that time to complete each sub-task is greatly reduced compared to the time to complete the whole task. Therefore, the system can run with much higher clock frequency. Pipelining the architecture provides not only the advantage of reducing cycle time, but it also introduces parallelism into the serial architecture because many sub-tasks can be performed simultaneously [23][24]. Pipelined architectures are found in many processors, whether the processor is dedicated to perform a specific task or it is the core for a high-speed general purpose CPU [25]. A pipelined architecture for image segmentation by adaptive progressive thresholding has been presented by Asari et al. [26]. High performance online segmentation procedure is achieved by mapping the segmentation algorithm onto a linear pipelined architecture in which the computation is fully overlapped with I/O operations. A pipelined design of a specific purpose processor based on the CORDIC algorithm for the calculation of the Hough transform has been presented in Bruguera et al. [27]. The data parallelism is exploited by partitioning the angle space into several subspaces that can be simultaneously processed.

Pipelined architectures require simple control logic because controlling signals are partitioned into stages in the pipeline. This fact is particularly useful for reducing the instructions overhead which is significant for an architecture that is controlled by a centralized unit. Since a distortion correction architecture for real time applications should be a compact architecture that can sustain a high throughput rate with minimal control logic for less power consumption in mobile systems, a pipelined and dedicated hardware architecture is designed to correct barrel distortion in wide-angle camera images in this research.

CHAPTER 3

BARREL DISTORTION CORRECTION ALGORITHM

The algorithm to correct barrel distortion based on least squares estimation and its simulation are discussed in this chapter. The chapter is organized as follows: the distortion correction algorithm is described in section 3.1; the methods to estimate the expansion coefficients and back mapping coefficients are described in section 3.2 and 3.3 respectively. In section 3.4, the software development procedure and the simulation results are presented.

3.1. BARREL DISTORTION CORRECTION ALGORITHM BASED ON LEAST SQUARES ESTIMATION

In this section, we discuss the algorithm to correct barrel distortion based on least squares estimation presented by Asari et al. [13]. The distortion correction procedure assumes that the distortion is purely radial about the distortion center. It attempts to correct the distortion in images by mapping all pixels in the distorted image space onto the corrected image space. Since the corrected image would be larger than the distorted image, there are many vacant pixels in the corrected image whose intensities need to be computed. The intensity of a vacant pixel in the corrected image is based on the intensities of the four neighboring pixels in the distorted image. The algorithm is summarized by the following three major steps:

Step 1: Forward mapping all pixels in the distorted image space (DIS) onto the corrected/expanded image space (CIS).

Step 2: Back mapping every vacant pixel in CIS onto DIS

Step 3: Computing intensity of each vacant pixel based on the intensities of neighboring pixels in DIS by linear interpolation.

3.1.1. Forward Mapping

Let (u_c', v_c') represents the distortion center in DIS, and let (u_c, v_c) represents the corrected center in CIS. In DIS, magnitude ρ' and the angle $\theta'$of a vector **P′** from the distortion center (u_c', v_c') to any pixel location (u', v') are given by:

$$\rho' = \sqrt{(u' - u_c')^2 + (v' - v_c')^2} \tag{3.1a}$$

$$\theta' = \arctan\left(\frac{v' - v_c'}{u' - u_c'}\right) \tag{3.1b}$$

This pixel at (u', v') in DIS can be transformed to a new location (u, v) in CIS and the corresponding magnitude ρ and angle θ of the vector **P** from the corrected center (u_c, v_c) to the pixel position (u, v) are computed as:

$$\rho = \sqrt{(u - u_c)^2 + (v - v_c)^2} \tag{3.2a}$$

$$\theta = \arctan\left(\frac{v - v_c}{u - u_c}\right) \tag{3.2b}$$

The mathematical relationship between magnitudes ρ' and ρ is defined with an expansion polynomial of degree N and the angle remains unchanged as:

$$\rho = \sum_{n=1}^{N} a_n (\rho')^n \tag{3.3a}$$

$$\theta = \theta' \tag{3.3b}$$

where a_n's are the expansion coefficients that are obtained using least squares estimation. The technique to estimate the expansion coefficients a_n's will be discussed in more details in section 3.2. Since the distortion is assumed to be radial about the distortion center, the angles θ and θ' are the same. The coordinates of the new location (u, v) of the pixel in the CIS can be obtained by:

$$u = u_c + \rho \cos\theta \tag{3.4a}$$

$$v = v_c + \rho \sin\theta \tag{3.4b}$$

3.1.2. Back Mapping

Once all the pixels in the DIS are forward-mapped onto the CIS, there will be many vacant pixels in CIS. A vacant pixel (u, v) in CIS is mapped back onto the DIS to find the corresponding location (u', v'). The back mapping procedure is described by a back-mapping polynomial of degree N and the angle remains unchanged as:

$$\rho' = \sum_{n=1}^{N} b_n \rho^n \tag{3.5a}$$

$$\theta' = \theta \tag{3.5b}$$

where b_n's are the back-mapping coefficients that are calculated by a non-linear regression analysis employing least squares estimation for a finite number of points in the distorted image [28]. The estimation of the back-mapping coefficients will be discussed in more detail in section 3.3. The pixel location (u', v') can be obtained as:

$$u' = u_c' + \rho' \cos\theta' \tag{3.6a}$$

$$v' = v_c' + \rho' \sin\theta' \tag{3.6b}$$

3.1.3. Linear Interpolation

The pixel location (u', v') obtained by the back mapping procedure need not be an integer location. Hence, the intensity value of this location can be computed by linearly interpolating with the intensity values of the four neighboring pixels of (u', v') in DIS. Let the integer parts of the coordinate values (u', v') be represented by A and B as:

$$A = \lfloor u' \rfloor \tag{3.7a}$$

$$B = \lfloor v' \rfloor \tag{3.7b}$$

and let the fractional parts of the coordinate values be represented by A' and B' as:

$$A' = u' - A \tag{3.8a}$$

$$B' = u' - B \tag{3.8b}$$

Then, the four neighboring pixels of (u', v') are (A, B), $(A+1, B)$, $(A, B+1)$ and $(A+1, B+1)$ as illustrated in Figure 3.1.

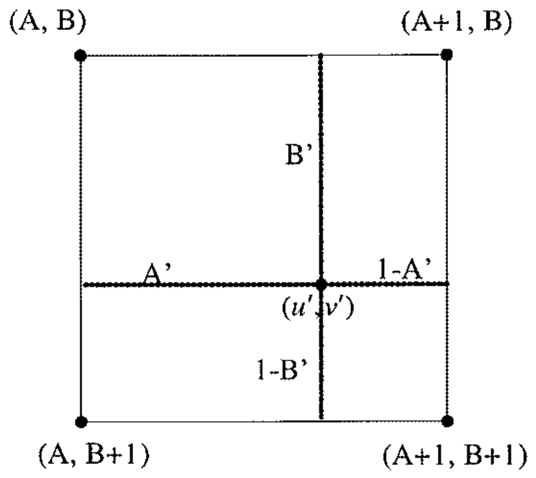


Figure 3.1: Neighboring pixels for linear interpolation.

The contribution of each neighboring pixel to the intensity of the pixel at (u,v) is computed based on the distance between the corresponding pixel location (u',v') in DIS to each neighboring pixel as:

$$I_1' = I'(A,B)\times(1-A')\times(1-B') \tag{3.9a}$$

$$I_2' = I'(A+1,B)\times(A')\times(1-B') \tag{3.9b}$$

$$I_3' = I'(A,B+1)\times(1-A')\times(B') \tag{3.9c}$$

$$I_4' = I'(A+1,B+1)\times(A')\times(B') \tag{3.9d}$$

where $I'(A,B)$, $I'(A+1,B)$, $I'(A,B+1)$ and $I'(A+1,B+1)$ are the intensities for the four neighboring pixels at locations (A, B), (A+1, B), (A, B+1) and (A+1, B+1) respectively.

Values of I'_1, I'_2, I'_3 and I'_4 are the contributed weights of the four neighboring pixels to the intensity of pixel at (u,v) in CIS which is computed as:

$$I(u\ ,v\ ) = \sum_{j=1}^{4} I'_j \tag{3.10}$$

3.2. ESTIMATION OF EXPANSION COEFFICIENTS

To estimate the expansion coefficients for a particular camera, an experimental grid image similar to the image shown in Figure 1.1(a) is created for the estimation of the expansion coefficients. The estimation of the expansion coefficients is conducted based on the least squares estimation analyzing the degree of straightness of the dots, which lie on a line of the grid captured by the wide-angle camera. Let (x_i^c, y_i^c) be the coordinate of the center of the dot lying in the i^{th} row and j^{th} column of the grid, and let there be L columns of test dots in the grid image with k_j dot centers in the j^{th} column as illustrated in Figure 3.2. All the dot centers in j^{th} column are used to obtain the best fit polynomial curve for j^{th} column, which is the basis to compute the expansion coefficients. To obtain a best fit polynomial curve for each j^{th} column, a polynomial of degree M is defined as:

$$R_j(x) = \sum_{\alpha=0}^{M} b_{\alpha j} x^{\alpha} \tag{3.11}$$

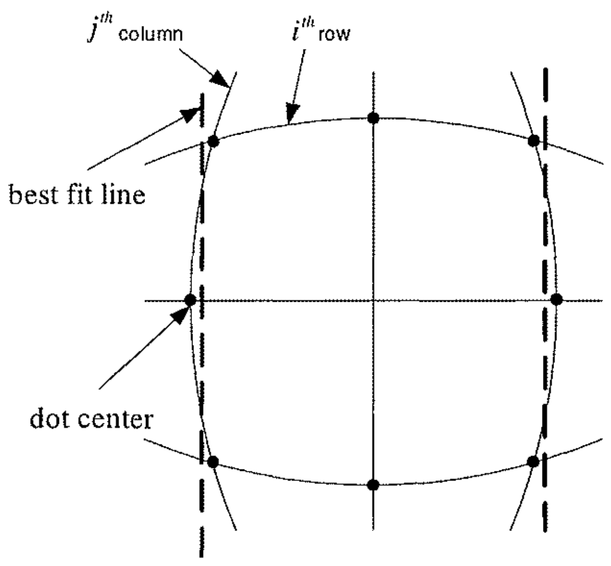


Figure 3.2: Illustration of dot centers in the distorted grid used to select expansion coefficients and back mapping coefficients.

Coefficients $b_{\alpha j}$'s are estimated by least squares estimation, which sufficiently emphasizes all points that are not close to the approximation without allowing them to dominate. To find the best linear fit for the set of k_j points in j^{th} column of the test dots, a first degree polynomial based on (3.11) is used. Therefore, two optimum polynomial coefficients are obtained from the best fit line analysis. A normalized error function e_j is defined as the normalized sum of magnitudes of the perpendiculars drawn from each of the k_j points on the best linear fit of j^{th} column as:

$$e_j = \frac{1}{k_j} \sum_{i=1}^{k_j} \left| \frac{b_{1j} x_{ij} - y_{ij} + b_{0j}}{(1 + b_{1j}^{\ 2})^{1/2}} \right| \tag{3.12}$$

and the total error for the whole grid image is obtained by:

$$\mathrm{E} = \sum_{j=1}^{\mathrm{L}} e_j \tag{3.13}$$

When the image does not have distortion, it is known to be in an ideal condition. Therefore, the total error E for that image would be zero. However, when an image has some degree of distortion, the total error E for that image would be a positive value. The main objective of the mathematical model is to determine a set of expansion coefficients a_n's that would minimize the total error E. Iterative computations are needed to find a set of expansion coefficients a_n's that would produce a minimal total error E. Various sets of coefficients a_n's are found during the iterative computations to search for the right set based on a "globalization strategy" to select the new set of coefficients [29]. On the basis of this strategy, the new set of expansion coefficients can be obtained by using the following recursive relationship:

$$a_n(\Delta+1) = a_n(\Delta) - \alpha n^{\beta} E(\Delta)\frac{1}{\left(\dfrac{\partial E(\Delta)}{\partial a_n}\right)} \quad \text{for n = 1, ..., N} \tag{3.14}$$

where α is the convergence rate parameter, β is the expansion index, and $\dfrac{\partial E}{\partial a_n}$ is the error gradient. Here α and β are chosen to ensure that for every (Δ+1)th iteration, E(Δ+1)< E(Δ). The iterative procedure ends when the total error E becomes smaller than a pre-specified limit ε i.e., $E(\Delta) \leq \varepsilon$ or when the total error E of the next iteration becomes larger than or equal to the total error E of this iteration i.e., $E(\Delta + 1) \geq E(\Delta)$. The set of expansion coefficients a_n's at the point when the iterative procedure ends is the expansion coefficients used to correct distortion.

3.3. ESTIMATION OF BACK MAPPING COEFFICIENTS

As discussed in section 3.1, the correct image is the expanded version of the distorted image. Once all pixels contained in the distorted image are forward-mapped onto the corrected image space, there will be many vacant pixels in the correct image. To obtain the correct intensity information for these vacant pixels, a back mapping polynomial is derived and is used to map every pixel from the CIS onto the DIS as in (3.5) in section 3.1.2. The back-mapping coefficients b_n's are calculated by using non-linear regression analysis employing least squares estimation for a finite number of points in the distorted image.

3.4. SOFTWARE SIMULATION

The various steps involved in the distortion correction algorithm have been simulated using different software tools viz. Visual C++, Excel, and Adobe Photoshop.

3.4.1. Estimation of Expansion Coefficients and Back Mapping Coefficients

The distorted grid image shown in Figure 1.1b was first binarized to obtain an image with black dots on a white background by using Photoshop imaging software. Because of the clear contrast in the grid image between the black dots and the white background, a simple thresholding technique works well to produce the binary image.

The next step in the algorithm is to calculate the centers of the black dots in the distorted image since the correction algorithm relies on the coordinates of these centers of dots to correct the image. The center (x_i^c, y_i^c) for the i^{th} dot is obtained by calculating the average of the locations of N_i pixels making up dot i as:

$$x_i^c = \frac{1}{N_i}\sum_{j=1}^{N_i} x_{ij} \tag{3.15a}$$

$$y_i^c = \frac{1}{N_i}\sum_{j=1}^{N_i} y_{ij} \tag{3.15b}$$

Using the best-fit line analysis available from numerical analysis tool Excel, the L best-fit polynomials for L columns of dots in the grid image are obtained. Coefficients b_{0j}'s and b_{1j}'s are extracted from these best-fit polynomials for calculation of the expansion coefficients a_n's.

The procedure of selecting the new set of expansion coefficients a_n's is based on the iterative procedure described in section 3.2. A C++ program was developed to perform this iterative procedure. Figure 3.3 describes the behaviour of this program. Based on the current set of expansion coefficients a_n's, new locations for dot centers are computed. The total error E is then calculated and compared to the total error in previous iteration and the pre-specified limit ε. If the total error E is determined that it can be reduced further as the results of the comparisons, a new set of expansion coefficients a_n's are computed and the whole procedure is repeated. The iterative procedure ends when the total error E is either less than ε or it is greater than the total error in previous iteration.

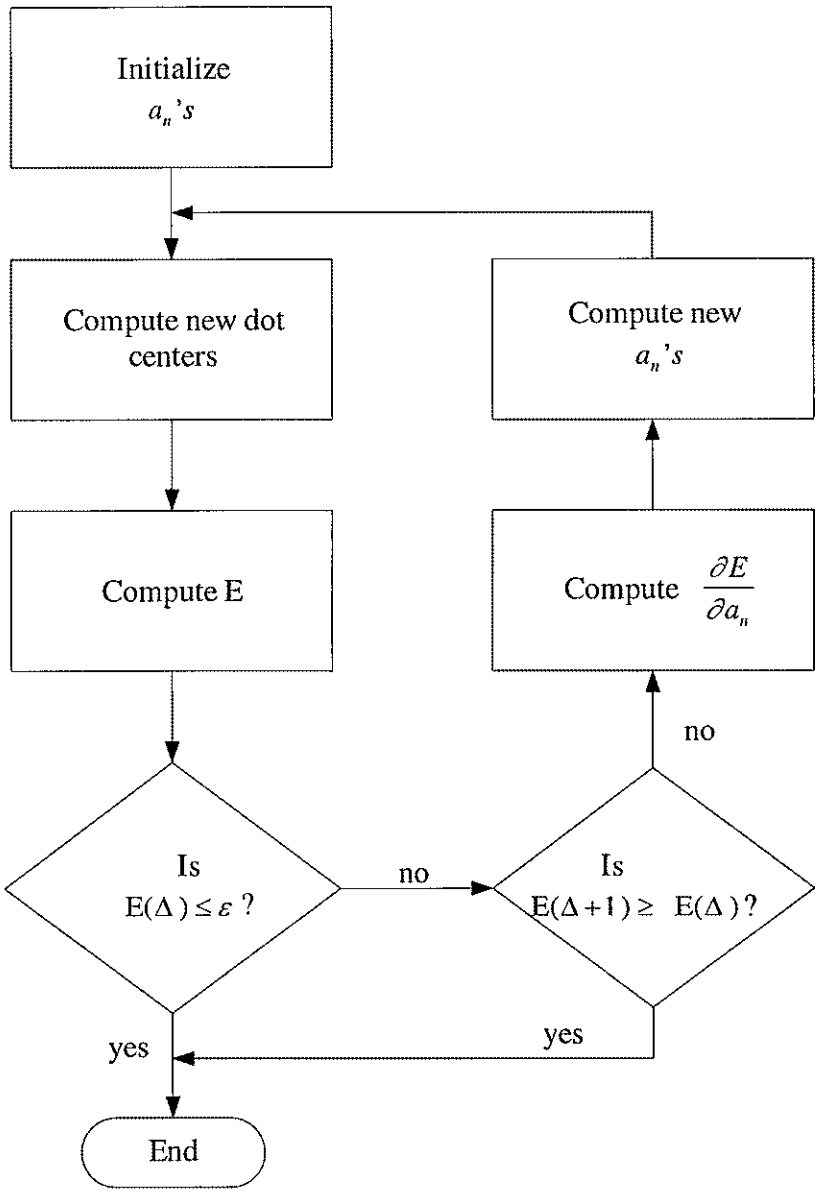

Initialize
a_n's
Compute new dot centers
Compute E
Is $E(\Delta) \le \varepsilon$?
no
yes
End
Is $E(\Delta+1) \ge E(\Delta)$?
no
yes
Compute $\frac{\partial E}{\partial a_n}$
Compute new a_n's

Figure 3.3: Flow chart for software simulation to obtain expansion coefficients.

A set of expansion coefficients obtained by this procedure is presented in Table 3.1. Once the final set of expansion coefficients a_n's is selected, coordinates of all centers of the dots in the distorted image are mapped onto the corrected image space. With two sets of coordinates of the same dots, distorted coordinates and corrected coordinates, numerical analysis tool Excel is used to estimate the back mapping coefficients b_n's.

Table 3.1: Expansion coefficients obtained from simulation results.

a_1	0.96156
a_2	0.00168
a_3	-0.000045
a_4	0.0000001597

3.4.2. Simulation of the Distortion Correction Algorithm

With both sets of expansion coefficients a_n*'s* and back mapping coefficients b_n*'s* available, a C++ program was developed to correct the barrel distortion in images. The program executes the sequence of operations described in section 3.1. The size of the DIS is expanded to the correct size of CIS based on equation (3.3). Then, every pixel in DIS is forward-mapped onto CIS. Using (3.5), every vacant pixel in CIS is then back mapped onto the DIS in order to calculate the intensity value based on the linear interpolation described in (3.7)-(3.10).

The distorted image captured from the experimental grid is shown in Figure 3.4(a) and the corrected result obtained from the simulation of the correction algorithm based on least squares estimation is shown in Figure 3.4(b). It is shown that the grid is straightened as a result of applying the correction method described in section 3.1. It is observed that images captured by wide angle camera for many practical purposes can be corrected based on the same technique described here. A typical gastrointestinal image, which is captured by wide angle camera, is shown in Figure 3.5(a), and the corrected image after applying the same algorithm to correct the distortion is shown in Figure 3.5(b).

It is also observed that if all pixels in CIS are assumed to be vacant pixels, the intensity for every pixel in the entire CIS can be computed based on back mapping and linear interpolation provided that the size of DIS, size of CIS, distorted center, corrected center, and back mapping coefficients b_n*'s* are given. This fact is useful for the hardware design because it is no longer necessary to implement the forward mapping step which would reduce the complexity of the hardware architecture. The C++ program was altered to simulate this modification in the implementation of the correction algorithm and the result is shown in Figure 3.6. The result validates the expectation that hardware implementation of the correction algorithm can bypass the forward-mapping step provided that system parameters can be obtained from software means.

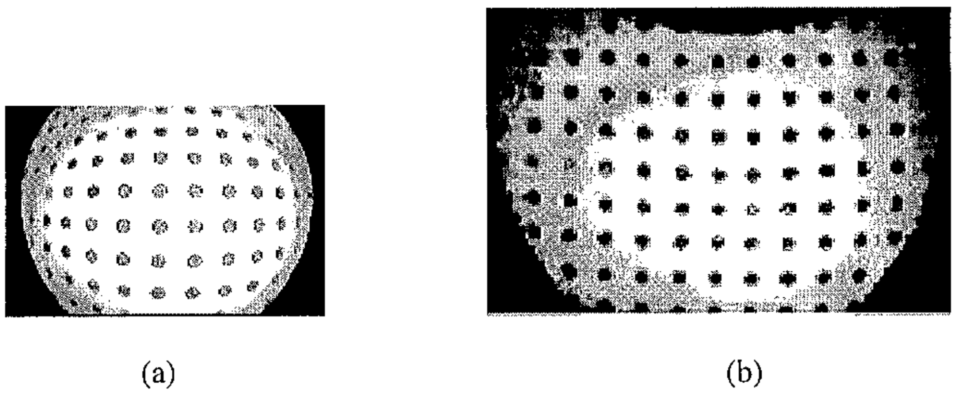
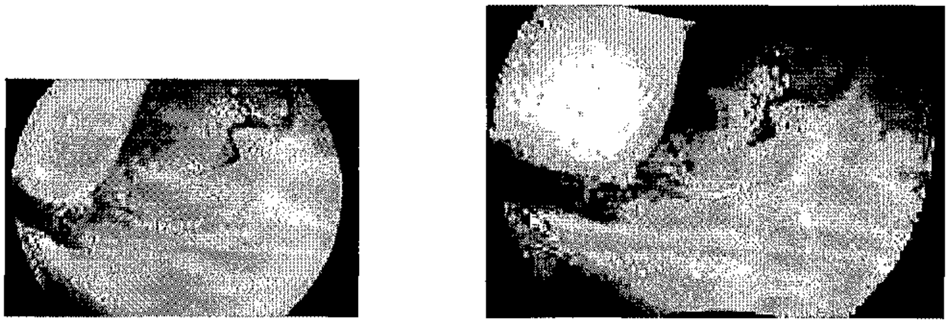
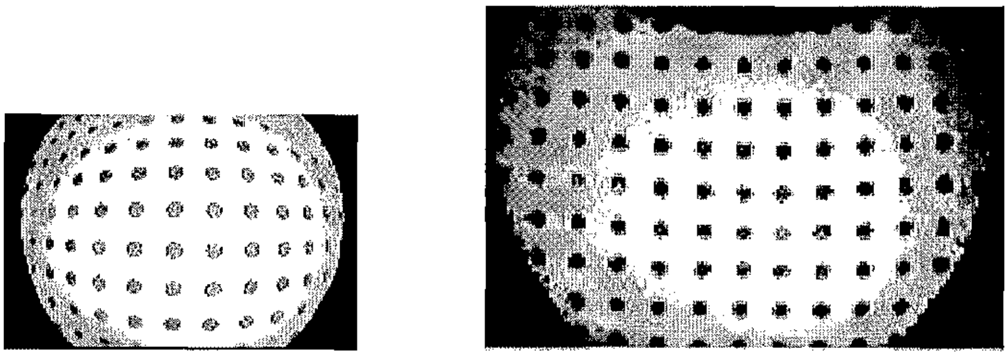


Figure 3.4: (a) Distorted image of experimental grid; (b) Corrected image of (a).



(a) (b)

Figure 3.5: (a) Typical endoscopic image (b) corrected image of (a).



(a) (b)

Figure 3.6: (a) Distorted image of experimental grid; (b) Corrected image of (a) with only back mapping and linear interpolation.

CHAPTER 4

DISTORTION CORRECTION ARCHITECTURE

The distortion correction algorithm presented in chapter 3 provides a method to correct barrel distortion in an image by the three main steps:

Step 1: Forward mapping of all the pixels in DIS onto CIS

Step 2: Back mapping of all vacant pixels in CIS onto DIS

Step 3: Calculating the intensities for all vacant pixels in CIS by linear interpolation.

However, as noted in chapter 3, we recognize that if the size of the distorted image, the size of the corrected image, the back mapping coefficients, the distorted center, and the corrected center are given to the system, the intensities of every pixel in CIS can be calculated by back mapping and linear interpolation procedures. By bypassing the forward mapping step, we greatly reduce the complexity of the system. So the development of the hardware architecture for the distortion correction is based on step 2 and step 3 only. The back mapping coefficients, the size of the distorted image, the size of corrected image, the distorted center, and the corrected center are computed by software means discussed in chapter 3 and they are presented to the hardware architecture as system parameters.

For each pixel at (u_i, v_i) in CIS, the back-mapping step relies on the magnitude ρ_i and angle θ_i. The rectangular coordinate (u_i, v_i) of a pixel is first transformed to the polar coordinate (ρ_i, θ_i). Then, the pixel is mapped back onto the DIS based on the back mapping polynomial (3.5). To determine the four neighboring pixels, the corresponding pixel location (ρ_i', θ_i') in DIS is transformed back to rectangular coordinate representation (u_i', v_i'). The computation of the intensity of the pixel is performed based on the linear interpolation of the four neighboring pixels as illustrated in (3.7) to (3.10). Thus, the correction of barrel distortion can be achieved by accomplishing four main tasks for each pixel in CIS as:

Task 1: Transformation of the rectangular representation (u_i, v_i) of a pixel in CIS to the polar representation (ρ_i, θ_i)

Task 2: Back-mapping the magnitude ρ_i to ρ_i', with the angle θ_i remains the same

Task 3: Transformation of the polar representation (ρ_i', θ_i') of the corresponding location in DIS to the rectangular representation (u_i', v_i')

Task 4: Reading the intensities of the four neighboring pixels, and calculating the intensity of the pixel at (u_i, v_i) in CIS by linear interpolation.

4.1. PIPELINED ARCHITECTURE DESIGN

The design of the entire architecture assumes the presence of the size of the corrected image, 4 back-mapping coefficients (b_n's), distortion center (u_c', v_c'), and corrected center (u_c, v_c). Figure 4.1 shows the block diagram of the distortion correction architecture with four main modules where each module performs one particular task from the four main tasks discussed earlier.

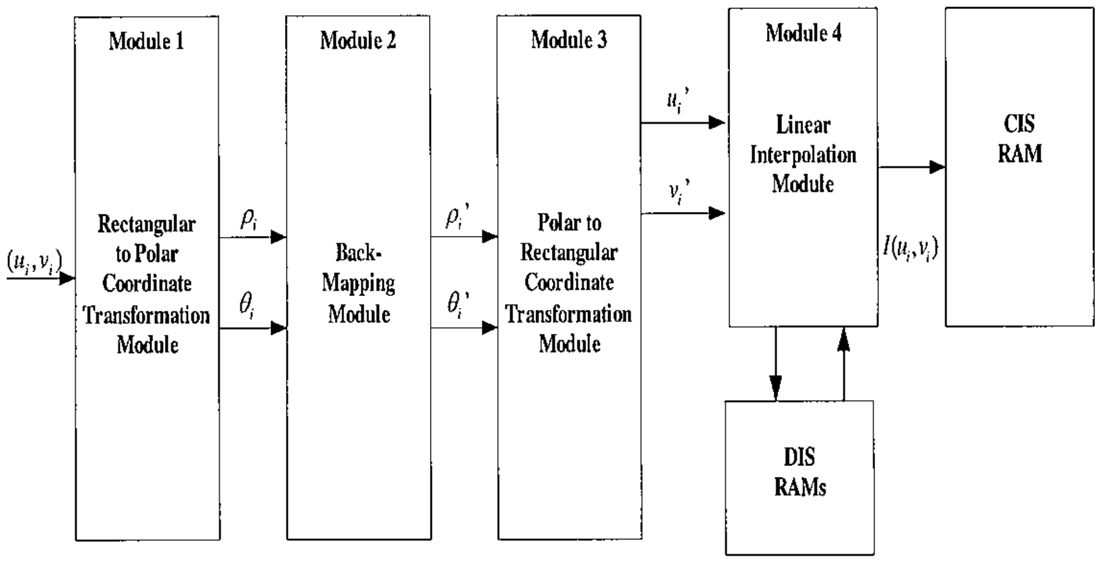


Figure 4.1: Block diagram of four main modules in the distortion correction architecture.

Since the architecture is designed for real time applications, which requires fast computations of all mathematical models presented in chapter 3, every module is pipelined to improve the throughput of the architecture. The detailed descriptions for module 1, module 2, module 3 and module 4 are discussed in the following sections.

4.1.1. Module 1: Rectangular to Polar Coordinate Transformation

The first major task of the distortion correction architecture is to convert the rectangular coordinate (u_i, v_i) of the location of i^{th} pixel in CIS to the polar coordinate representation (ρ_i, θ_i). Given the correct image size of U×V, each pixel in CIS is referenced by (u, v) where

$$u = 1, 2, ..U$$

$$\text{and } v = 1, 2, ..V$$

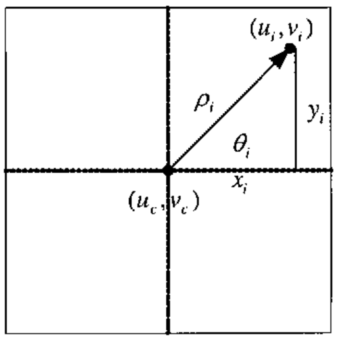


Figure 4.2: x_i and y_i components from a pixel (u_i,v_i) to correct center (u_c,v_c).

Two counters are used to advance (u,v) to the position of the next pixel in CIS. The x and y components of the pixel location are calculated based the location of the corrected center (u_c,v_c) as illustrated in Figure 4.2. The values of x_i and y_i components of i^{th} pixel are then presented to the inputs of a CORDIC (COordinate Rotation Digital Computer) module to perform the coordinate transformation. The CORDIC algorithm is a method for calculating various trigonometric and transcendental functions by rotating vectors, and it is one of the widely used algorithms in digital image processing applications. One of many useful applications of CORDIC algorithm is coordinate transformation. One major advantage of CORDIC algorithm that digital designers find very useful is that the CORDIC algorithm performs the transformation and many other calculations by series of simple shift and add operations, which can be designed with simple digital logic components [30][31].

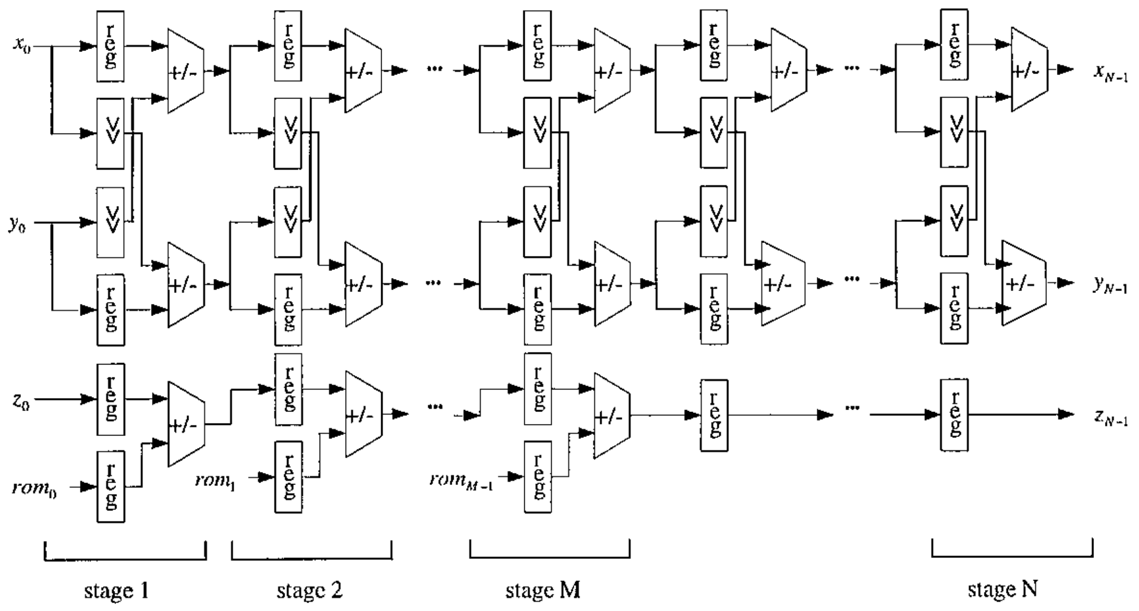


Figure 4.3: Architecture of CORDIC module (module 1).

By appropriately selecting the operational mode of the CORDIC algorithm, vectoring mode in this case, the magnitude ρ_i and angle θ_i are obtained from the rectangular coordinate representation (u_i, v_i). A pipelined CORDIC module is designed for this transformation, and the detailed architecture of the module is shown in Figure 4.3. At each j^{th} stage in the module, it performs the tasks of three simple CORDIC operations:

$$x_{(j+1)} = x_j - s_j.y_j.2^{-j} \tag{4.1a}$$

$$y_{(j+1)} = y_j + s_j.x_j.2^{-j} \tag{4.1b}$$

$$z_{(j+1)} = z_j - s_j.\tan^{-1}(2^{-j}) \tag{4.1c}$$

where $s_j = \begin{Bmatrix} +1 & y_j < 0 \\ -1 & otherwise \end{Bmatrix}$ and values for x_0, y_0 and z_0 are initialized to $x_0 = x_i$, $y_0 = y_i$ and $z_0 = 0$ for the vectoring mode of the CORDIC algorithm.

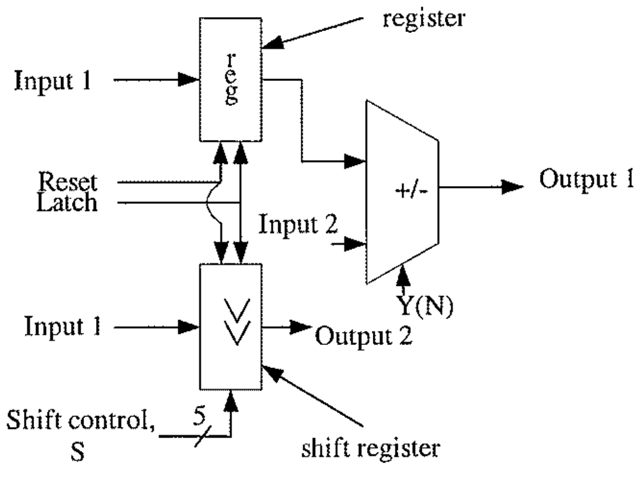


Figure 4.4: x and y sub-module in CORDIC module (module 1 in Figure 4.3).

At each state of the module, there are three sub-modules that are responsible for performing the operations in each of the CORDIC equations. The architecture for the x sub-module is shown in Figure 4.4, which consists of an n-bit register, an n-bit shift register and an add/subtract unit. Registers and shift registers operate in synchronization with the system clock. Each register has two control signals: "Latch" for latching the input into the register and "Reset" for resetting the register to zero at the active edge of the clock. Shift register also has two signals "Latch" and "Reset" with the addition of a 5-bit shift number j. The addition or subtraction operation of each add/subtract unit in every stage of the module is determined by the most significant bit (MSB) of the y value in that stage as described in (4.1). At each j^{th} stage, the operations of x sub-modules are:

(a) Input 1 is latched into the register

(b) Input 1 is right-shifted j positions and latched into the shift register, output 2 = output of the shift register

(c) If Y(MSB) = 1

Output 1 = output of register + input 2

Else

Output 1 = output of register - input 2

A y sub-module has the same architecture as that of the x sub-module, the only difference is the operation described in step (c) at the add/subtract unit as follows:

(c) If Y(MSB) = 1

Output 1 = output of register - input 2

Else

Output 1 = output of register + input 2

The value of input 2 in the x sub-module in the j^{th} stage is the value of output 2 of the shift register in the y sub-module in the same stage. The value of input 2 in the y sub-module is the value of output 2 of the shift register in the x sub-module.

A z sub-module has a similar architecture as those of the x and y sub-modules as shown in Figure 4.5. A z sub-module does not have a shift register as an x sub-module or a y sub-module does; it has two n-bit registers instead. At the j^{th} stage, the operations of z sub-modules are:

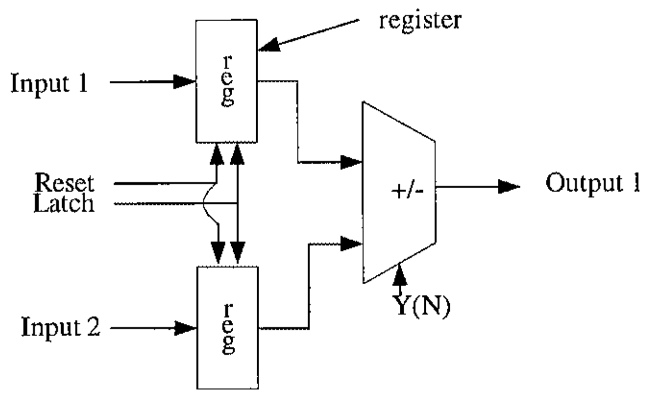


Figure 4.5: z sub-module in CORDIC module (module 1 in Figure 4.3).

(a) Input 1 is latched into register 1

(b) Input 2 (pre-computed $\tan^{-1}(2^{-j})$ term in ROM) is latched into register 2 at system initialization

(c) If Y(MSB) = 1

Output 1 = output of register 1 + output of register 2

Else

Output 1 = output of register 1 - output of register 2

For the first M stages in the module, the z-path in the CORDIC module is performed as described above; however, after M iterations, the term $\tan^{-1}(2^{-j})$ become too small that it can't be represented with M bit precision. Therefore, no further operation is needed and the result of the z value at stage M is passed along the pipeline to synchronize with x and y values at the outputs of the module.

One important fact that needs to be considered in the design for CORDIC module is the gain factor introduced by CORDIC algorithm. As the vector being rotated at each j^{th} stage of the module, a small gain g_j is introduced to the immediate value of magnitude ρ_j, which is described by a CORDIC equation for gain as:

$$g_j = \sqrt{1+2^{-2j}} \tag{4.2}$$

The product of all gain values approaches a constant as the number of iterations extends to infinity, and that constant gain is approximately equal to 1.647. For the design presented here, the number of iterations N is the number of stages in the module, which is 24. Therefore, the exact value of the gain that needs to be accounted at the output of the module is:

$$G = \prod_{j=1}^{24} \sqrt{1+2^{-2j}} = 1.646760258 \tag{4.3}$$

After N iterations, the scaled ρ_i is available at the output as x_{N-1} and angle θ_i is available as z_{N-1}. The correct magnitude ρ_i is finally obtained by multiplying the result ρ of the module with the reciprocal of the total gain, G^{-1}, which is 0.607252935. Since the module is designed as an absolute pipelined architecture, there are 24 stages in the module (one stage for each iteration). The multiplication of the scaled result ρ and reciprocal of the total gain G^{-1} adds 5 clock cycles to the latency of this module.

4.1.2. Module 2: Back-Mapping

Once ρ_i and θ_i are available from the previous vectoring-CORDIC module, ρ_i is mapped back onto DIS to obtain ρ_i' based on (3.5), and the angle θ_i remains unchanged since the distortion is purely radial. The purpose of this back-mapping procedure is to determine the corresponding location of the working pixel in DIS. The architecture for this module is shown in Figure 4.6 where each stage in the module consists of a set of registers and an appropriate number of functional units such as multipliers and adders. For each 24-bit multiplier in the architecture, the propagation delay is reduced to satisfy the high-speed demand by pipelining the multipliers with 5 stages. A pipelined multiplier takes 5 clock cycles to complete and an adder takes 1 cycle. Five-cycle delay, denoted by a 'D', units are introduced in this module as shown in Figure 4.6 to synchronize with data in the same stage since all functional units in the stage have various latency to complete their operations.

There are four back-mapping coefficients, b_1, b_2, b_3 and b_4, that are computed by software mean as discussed in chapter 3, and they are latched into this back-mapping module at system initialization. One issue in this module is the various word lengths of the terms ρ_i^2, ρ_i^3 and ρ_i^4 in (3.5) since the values of these terms may require 2N bits, 3N bits, and 4N bits to represent them where N=24. The values of the back-mapping coefficients are much less than 1 from the results of the simulation to select coefficients. The set of four back-mapping coefficients obtained in the experiment are listed in Table 4.1. To address this issue, appropriate data manipulation is conducted before they are fed to the multipliers. This is achieved by scaling one operand of the multiplier unit with a

Table 4.1: Back-mapping coefficients.

b_1	0.7115879058837890625
b_2	0.00035633612424135208l
b_3	-0.000002730343112489209
b_4	0.000000002741288085417817

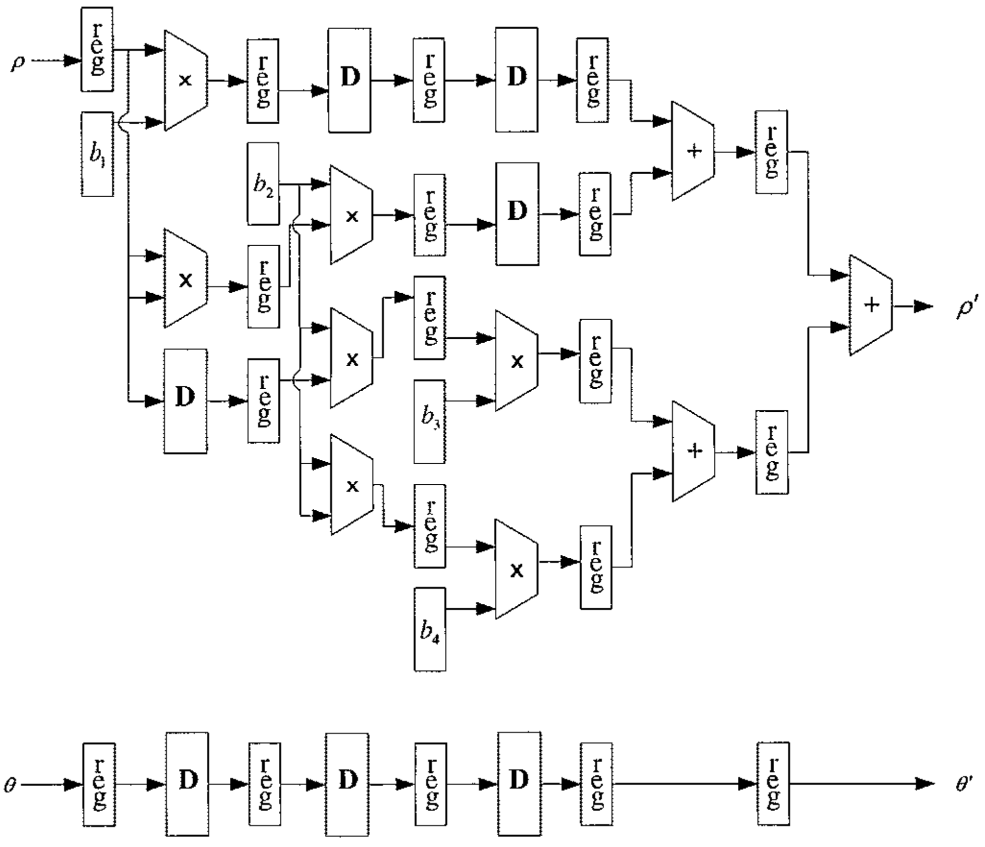


Figure 4.6. Architecture of the back-mapping module (module 2).

pre-determined scaled factor and scaling the second operand with the inverse of the scaled factor. For example, let's assume that one wants to compute the product **p** of a number **n** and another number **m,** $p = n \times m$. It is true that the equation $p = n \times m$ can be rewritten as $p = (n \times s) \times (m \times s^{-1})$ where **s** is the pre-determined scale factor. By scaling the two operands **n** and **m**, one knows that product **p** is computed accurately with the advantage that two numbers **n** and **m** can be manipulated as one wishes. With considerations of the magnitudes of the back-mapping coefficients, the word length in the architecture, and the precision in data representation, equation (3.5a) described in section 3.1.2. is rearranged as:

$$\begin{aligned} \rho_i' = & (\rho_i \times 2^{-7}) \times (b_1 \times 2^{7}) + (\rho_i^2 \times 2^{-14}) \times (b_2 \times 2^{14}) \\ & + (\rho_i^3 \times 2^{-21}) \times (b_3 \times 2^{21}) + (\rho_i^4 \times 2^{-24}) \times (b_4 \times 2^{24}) \end{aligned} \qquad (4.4)$$

The purpose of this manipulation of data is to scale down the value of ρ_i so that the terms ρ_i^2, ρ_i^3 and ρ_i^4 can still be represented with 24-bit word for any ρ_i within the range of representation. As shown in Figure 4.6, the values of back-mapping coefficients are determined and scaled up with appropriate factors described in (4.4) and they are latched into registers at system initialization.

The details of the operations at each stage in this back-mapping module are:

Stage 1: Calculations of ρ_i^2 and $\rho_i \times b_1$ from input ρ_i and b_1 complete after 5 clock cycles (multiplier latency). The results and ρ_i are passed to the next stage.

Stage 2: Calculations of ρ_i^3, ρ_i^4 and $\rho_i^2 \times b_2$ from b_2, ρ_i and ρ_i^2 complete after 5 clock cycles (multiplier latency). The results and $\rho_i \times b_1$ are passed to the next stage.

Stage 3: Calculations of $\rho_i^3 \times b_3$ and $\rho_i^4 \times b_4$ from ρ_i^3, ρ_i^4, b_3, and b_4 complete after 5 clock cycles (multiplier latency). The results and $\rho_i \times b_1$, and $\rho_i^2 \times b_2$ are passed to the next stage.

Stage 4: Calculations of $(\rho_i \times b_1)$ + $(\rho_i^2 \times b_2)$ and $(\rho_i^3 \times b_3)$ + $(\rho_i^4 \times b_4)$ from $\rho_i \times b_1$, $\rho_i^2 \times b_2$, $\rho_i^3 \times b_3$ and $\rho_i^4 \times b_4$ complete after 1 clock cycle (adder latency). The results are passed to the next stage.

Stage 5: Calculation of $\rho_i' = [(\rho_i \times b_1) + (\rho_i^2 \times b_2)] + [(\rho_i^3 \times b_3) + (\rho_i^4 \times b_4)]$ is completed after 1 clock cycle (adder latency). The result is passed to the output of the module.

While the input ρ_i is transformed to ρ_i' as it goes through the stages in the module, the input θ_i is passed along in synchronization with ρ_i through the stages as shown in Figure 4.6.

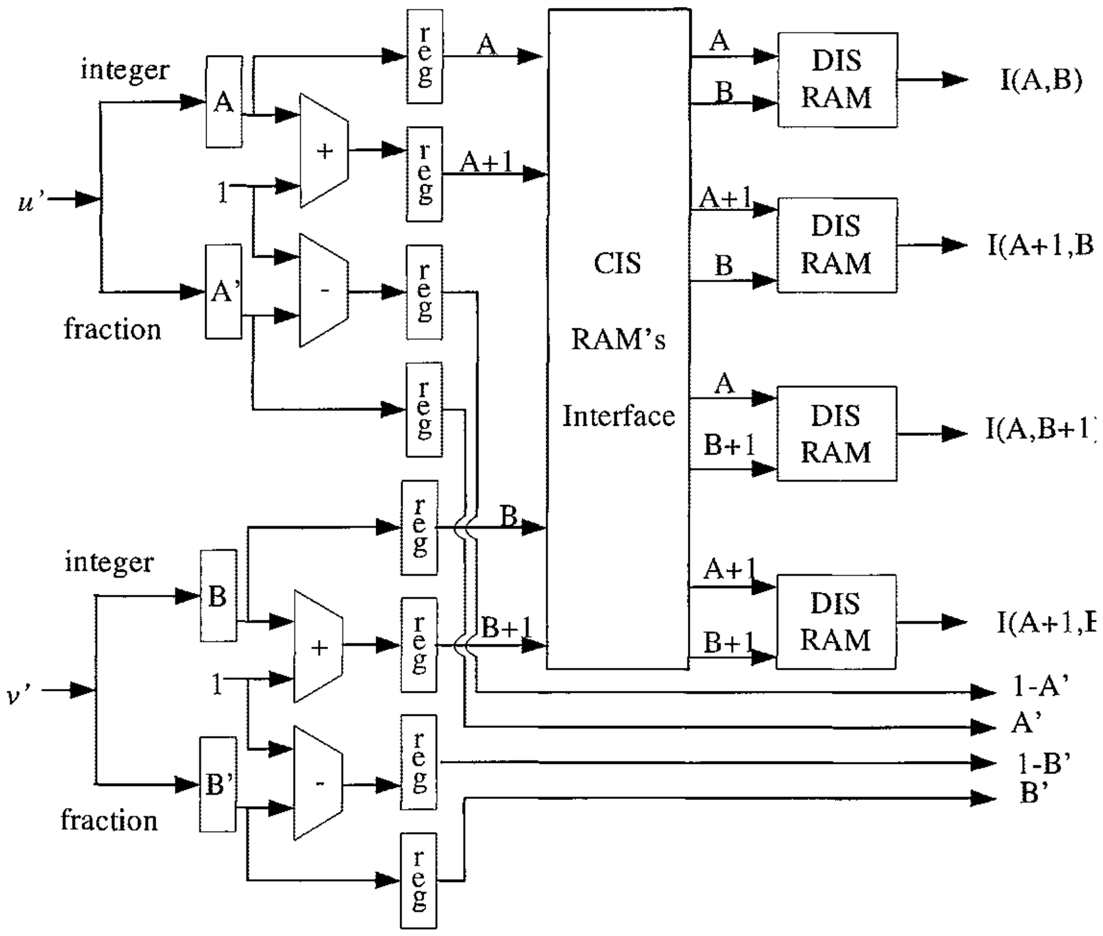
4.1.3. Module 3: Polar to Rectangular Coordinate Transformation

The third task of the correction algorithm is to transform the polar coordinate (ρ_i', θ_i') of the corresponding pixel location in DIS that is calculated by module 2 (back-mapping module) to the rectangular coordinate representation (x_i', y_i'). Once again, the CORDIC algorithm is implemented to perform this coordinate conversion, but this time the operational mode of the CORDIC algorithm is the rotation mode. The pipelined architecture for this module is similar to the architecture of module 1 shown in Figure 4.3. The rotation CORDIC module performs the same basic operations described in (4.1). However, there are some differences compared to the vectoring-CORDIC module. The first difference between the rotation-CORDIC module and the vectoring-CORDIC module is the fact that the gain factor needs to be applied to the initial value of ρ_i' before it is presented to the input of this rotation-CORDIC module. The second difference is the three initial values for x_0, y_0 and z_0 of this module. Specifically, three initial values for this rotation-CORDIC module are ρ_i', 0 and θ_i' for x_0, y_0 and z_0 respectively. The third difference is that the addition or subtraction operation of each add/subtract unit in every stage is determined by the MSB of the immediate value of z value in that stage, which can be described as $s_j = \begin{Bmatrix} +1 & z_j < 0 \\ -1 & otherwise, \end{Bmatrix}$ rather than the y value as in the vectoring-CORDIC module.

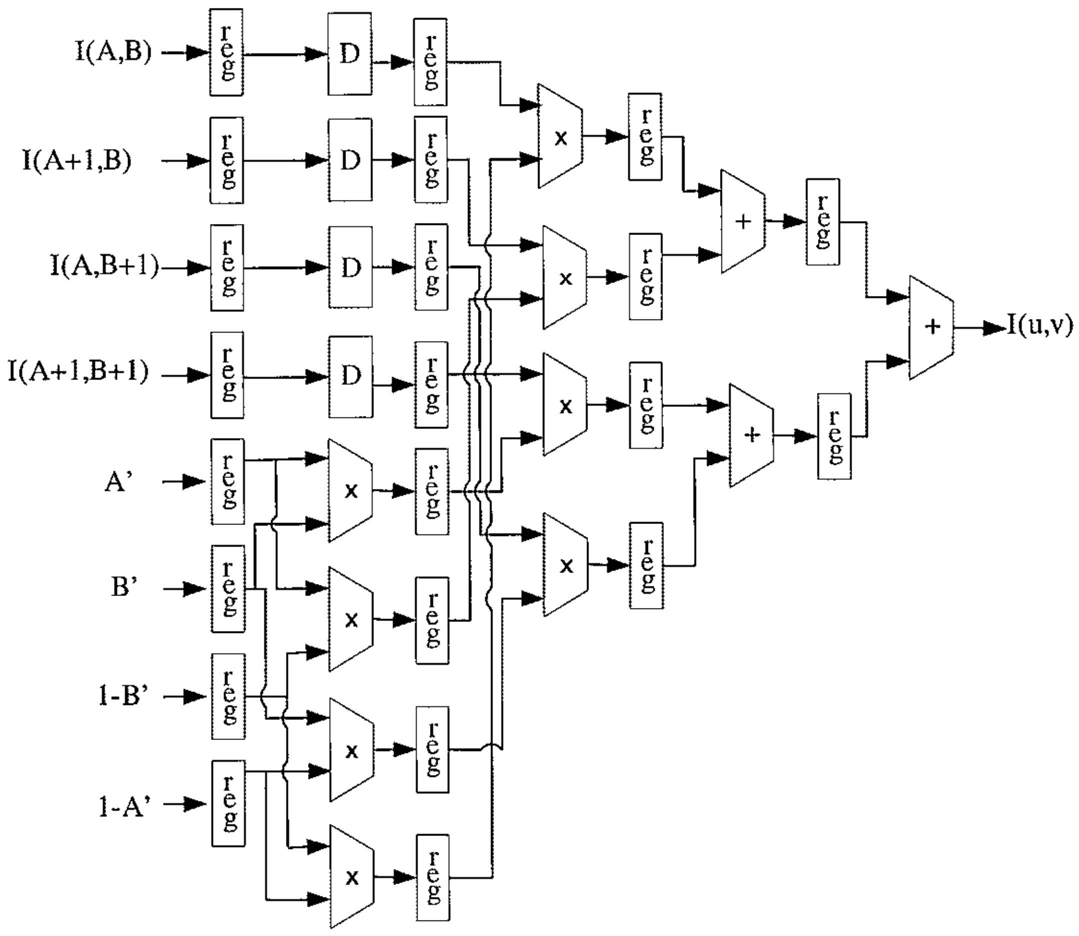
Excepting the differences mentioned above, all functional x, y and z sub-modules in the rotation-CORDIC module perform the exact operations as those sub-modules in vectoring-CORDIC module. This rotation-CORDIC module is also a 24-stage architecture as that of the vectoring-CORDIC module. Including the time to compute the correct (not scaled) value of ρ_i', the latency for this rotation-CORDIC module is the same as the latency of module 1 which is 29 cycles.

4.1.4. Module 4: Linear Interpolation

The x_i' and y_i' components of the pixels, which is computed by the rotation-CORDIC module are used to determine the exact location (u_i', v_i') of the pixel in DIS by adding/subtracting with the distorted center (u_c', v_c') that is pre-stored in the module when the system is initialized. The main task of this module is to calculate the intensity of the current working pixel in CIS by calculating the weighted average intensity value from the intensities of the four neighbouring pixels. Figure 4.7 shows the pipelined architecture for this linear-interpolation module, which performs the operations described in (3.7) to (3.10).



(a): Architecture for computations of four neighboring pixels addresses.



(b): Architecture for computation of linear interpolation.

Figure 4.7: Architecture for linear interpolation module (module 4).

The functionalities of this module are to read the intensity values of the four neighboring pixels of the location (u_i', v_i') and to calculate the intensity value for the pixel at (u_i, v_i) based on linear interpolation. Since all modules in this system, including this linear-interpolation module, are pipelined for the purpose of completing the computation of one pixel intensity per clock cycle, the task of reading four surrounding pixels intensities must be finished in one clock cycle. Obviously, four READ operations from a DIS RAM can be performed in one long clock cycle. The issue is that the system should be driven as fast as possible since performance is our main criteria. In addition to the fact that RAM accessing rate has been steadily improved for many years, the rapid expanding rate of chip size has made it simple to use four RAM modules to store duplicated DISs, all of which can be updated simultaneously by the frame grabber of the camera.

All mathematical calculations described in (3.7) to (3.10) are broken down to various phases that are implemented in separate stages in this module. The details of the operations in each stage are:

Stage 1:

- Input u_i' and v_i' are separated into integer parts and fractional parts as: $A = floor(u_i')$, $A' = u' - A$, $B = floor(v_i')$ and $B' = u' - B$
- Operations of $1 - A'$, $1 - B'$, $A + 1$ and $B + 1$ are completed after 1 clock cycle (adder latency).
- A, $A + 1$, B, $B + 1$, A', $1 - A'$, B' and $1 - B'$ are passed to the next stage

Stage 2:

- A, $A+1$, B and $B+1$ are used to interface with DIS RAM modules to access four neighboring pixels (A,B), $(A+1,B)$, $(A,B+1)$ and $(A+1,B+1)$
- The intensities of four neighboring pixels $I(A,B)$, $I(A+1,B)$ $I(A,B+1)$ and $I(A+1,B+1)$ received from DIS RAM modules and A', $1-A'$, B' and $1-B'$ are passed to the next stage

Stage 3:

- Computations of $[(1-A')\times(1-B')]$, $[(A')\times(1-B')]$, $[(1-A')\times(B')]$, and $[(A')\times(B')]$ are completed after 5 clock cycles (multiplier latency).
- Delay units are used to synchronize values of $I(A,B)$, $I(A+1,B)$, $I(A,B+1)$ and $I(A+1,B+1)$ with results of computations in this stage.

Stage 4:

- I_1, I_2, I_3 and I_4 are computed based on (3.9) with operands passed from stage 3. Operations in this stage complete after 5 cycles (multiplier latency).

Stage 5 and stage 6:

- The sum of I_1, I_2, I_3 and I_4, which is the intensity $I(u_i,v_i)$ of pixel at (u_i,v_i) in CIS, is computed as described in (3.10) in these two stages, and the result is available at the output of the module for writing to the CIS RAM.
- Operations in these two stages take 2 clock cycles (1 cycle per stage).

4.2. AN IMPROVED ARCHITECTURE DESIGN

In the previous design, we presented an architecture that processes all the pixels in CIS [32]. However, we recognize that the image is symmetric with respect to the corrected center since the distortion in image is assumed to be purely radial. What this means is that for each pixel at position (u_1, v_1) in quadrant I, there are three other pixels with positions (u_2, v_2), (u_3, v_3), and (u_4, v_4) in quadrant II, III, and IV respectively, which have the same level of distortion, same radial magnitude ρ and angle θ as shown in Figure 4.8. With this symmetric property in images, we recognized that once a pixel at location (u_i, v_i) in quadrant 1, e.g. location (u_1, v_1) in Figure 4.8, is mapped back onto a location in DIS, e.g. location (u_1', v_1'), we can also determine the other three pixels in quadrant II, III and IV in DIS, e.g. positions (u_2', v_2'), (u_3', v_3'), and (u_4', v_4'), that have the same magnitude ρ' and angle θ' with respect to the distorted center (u_c', v_c'). That means we can effectively calculate the intensities for four pixels simultaneously by performing the interpolation procedure for four pixels in parallel. This will improve the throughput of the system by a factor of 4. And since we eliminate 75 percent of redundant computations in most of the major modules, except the interpolation module, we also reduce the dynamic power consumption by the system compared to the earlier design. Figure 4.9 shows the block diagrams for the complete system with main modules of the system. Note that there are four interpolation sub-modules in module 4, and each interpolation sub-module is a replicate of the interpolation module discussed in section 4.1.4. for which the architecture is shown in Figure 4.7.

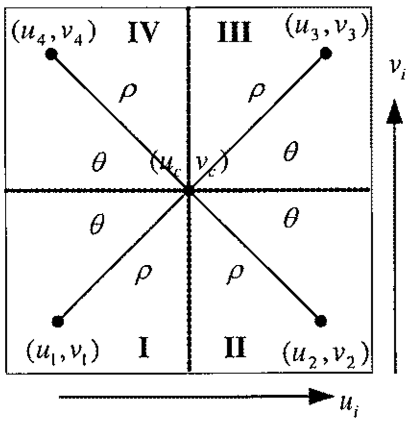


Figure 4.8: Symmetry in image.

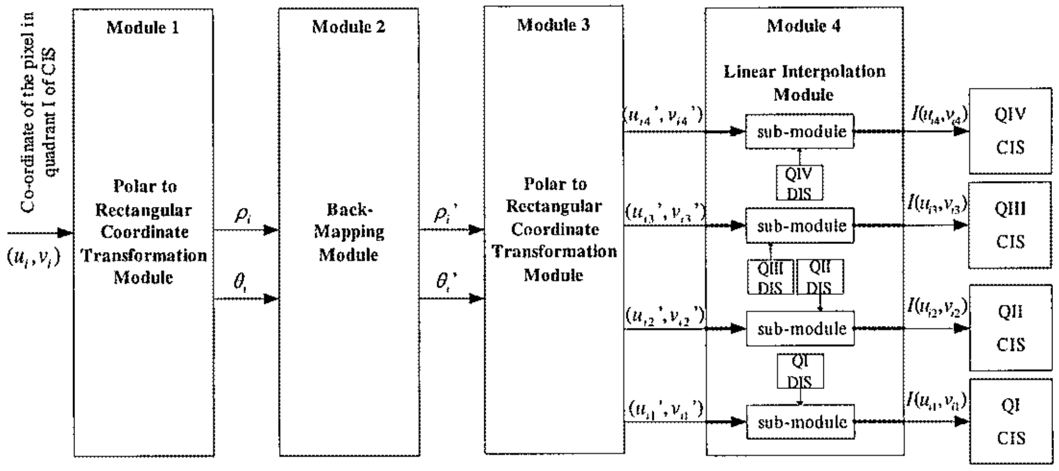


Figure 4.9: Block diagram of the distortion correction architecture with consideration of the symmetry property in distorted image.

Besides the additional hardware resources needed to calculate four intensities simultaneously, there are slight modifications in some of the main modules compared to the earlier design presented in section 4.1. In module 1, the vectoring-CORDIC module, the location of the pixel in CIS is the main parameter needed to compute its intensity and this is true for all pixels. Now, with the consideration of the symmetric property in the image, only locations of pixels in quadrant I of CIS are needed to compute the intensities of all pixels. Each pixel location in quadrant I is presented to two add/subtract units first to determine the other three locations of pixels in the three remaining quadrants based on the corrected center (u_c, v_c). All four locations are then fed to the vectoring-CORDIC module, but only the location of the pixel in quadrant I is transformed to polar representation in the same way each pixel is transformed in the earlier design. The other three pixel locations are passed along the stages for future use in subsequent modules. All computations in this vectoring-CORDIC module remain the same as described in section 4.1.1. Therefore, the architecture to perform these operations is the same as the one shown in Figure 4.3.

No major changes are needed in module 2 (back-mapping module) and module 3 (rotation-CORDIC module) because the back-mapping procedure and the polar-to-rectangular transformation procedures are done to one pixel at a time as in original design. The only upgrade to these modules is to provide resources along the stages of the two modules for all four pixel locations.

The x_i' and y_i' components of the pixel that are available at the outputs of the rotation-CORDIC module are used to compute the four pixel locations, (u_1', v_1'), (u_2', v_2'), (u_3', v_3'), and (u_4', v_4'), in four different quadrants in DIS. The four locations (u_1', v_1'), (u_2', v_2'), (u_3', v_3'), and (u_4', v_4') are used as the inputs to four sub-modules in the interpolation module as shown in Figure 4.9. Each sub-module functions in the same way as the interpolation module or module 4 in the original design, which is discussed in detail in section 4.1.4. Since there are four interpolation sub-modules, the intensities for all four pixels in CIS, (u_1, v_1), (u_2, v_2), (u_3, v_3), and (u_4, v_4), are computed simultaneously. These intensities are then written back to the corresponding pixel position in each quadrant of CIS based on the locations (u_1, v_1), (u_2, v_2), (u_3, v_3), and (u_4, v_4) as illustrated in Figure 4.8.

CHAPTER 5

HARDWARE SIMULATION AND PERFORMANCE EVALUATION

The pipelined architecture to correct barrel distortion in wide-angle camera images presented in this thesis is implemented with Altera's Quartus II design tools using VHDL (Very High Speed Integrated Circuit Hardware Description Language). The entire architecture is fitted into a FPGA of the APEX family from Altera Inc. A top down design technique is used to design the architecture. Each main system module is implemented and tested separately to validate its functionality before all four main modules are integrated to become the distortion correction system. As shown in the flow diagram in Figure 5.1, the operations of the entire system are very simple. Once the system is initialized and a signal is given to begin the correction process, the system performs the correction algorithm continuously.

For the earlier design that does not consider the symmetry in images, the summary of the simulation results and performance evaluation are listed in table 5.1. Based on simulation results, the maximum clock frequency that can be applied to the system is 40 MHz. The entire architecture consumes 18,344 logic elements and 15,355 flip-flops, which is about 75 percent of the total area in the selected chip (EP20K600EBC652-1X FPGA) from Altera. The total latency for the first result to be produced at the output of the entire system is 89 clock cycles, and the architecture produces an output in every clock cycle after this initial latency while it is in the processing mode. So, it would take

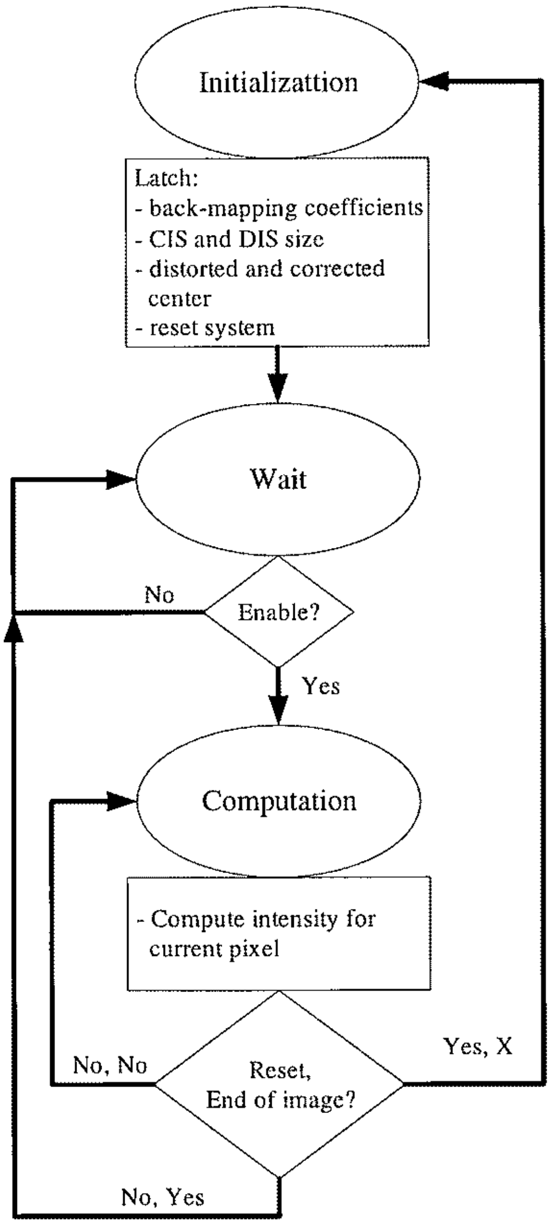


Figure 5.1: Flow diagram of the operations in the distortion correction architecture.

Table 5.1: Implementation results of the pipelined architecture to correct barrel distortion in wide-angle camera images.

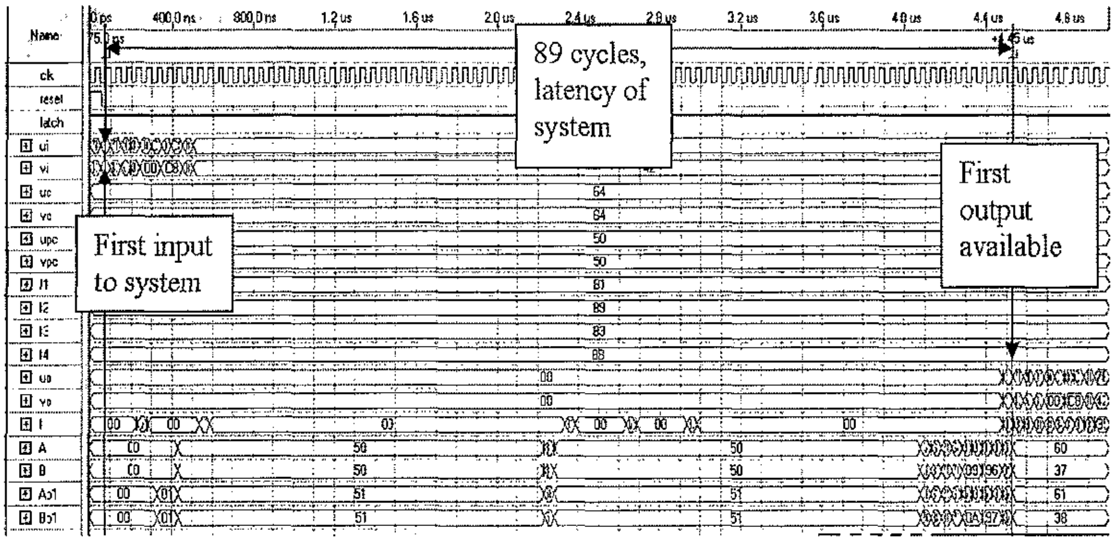
Name	**Result**
Chip Name	EP20K600EBC652-1X FPGA
Maximum clock frequency	40 MHz
Logic elements	18,344
Flip flops	15,355
I/O power	1.04621e-002 mW
Internal power	360.87 mW
Latency	89 cycles
RAM accesses	~7 ns
DIS RAMs refresh time (L pixels)	<= 7L ns
CIS processing time for 1 image (L pixels)	(L + 88) cycles
Time to correct an 1-Mpixel image with clock frequency of 40 MHz.	33 ms
Throughput of the correction system running with a 40 MHz clock to correct 1-Mpixel images	30 frames or 30 Mpixels per second

$(L+88)$ cycles to process an image with L pixels. The time to refresh the duplicated DIS RAMs would be less than or equal to $(L\times 7)$ ns because one write operation to the SRAM board takes 7 ns for a typical SRAM board available in market today. The total time to correct one distorted image is the sum of the time to process all pixels in CIS RAM and the time to refresh DIS RAM's with a new image. It is observed that for a corrected image with size of 1024×1024 and a system clock of 40 MHz, it would take $\left((10^6+88)cycles\times 25\, {}^{ns}\!/_{cycle}\right)+\left(10^6\, write\times 7\, {}^{ns}\!/_{write}\right)$ or about 33 ms to correct one image. Thus, the throughput of the system is about 30 1024×1024 images or 30 Mpixels per second which surpasses the required throughput of 25 Mpixels per second set out at the beginning. This result satisfies the objective set out at the beginning of this research, which is to design an architecture that is capable of correcting barrel distortion of images in real time.

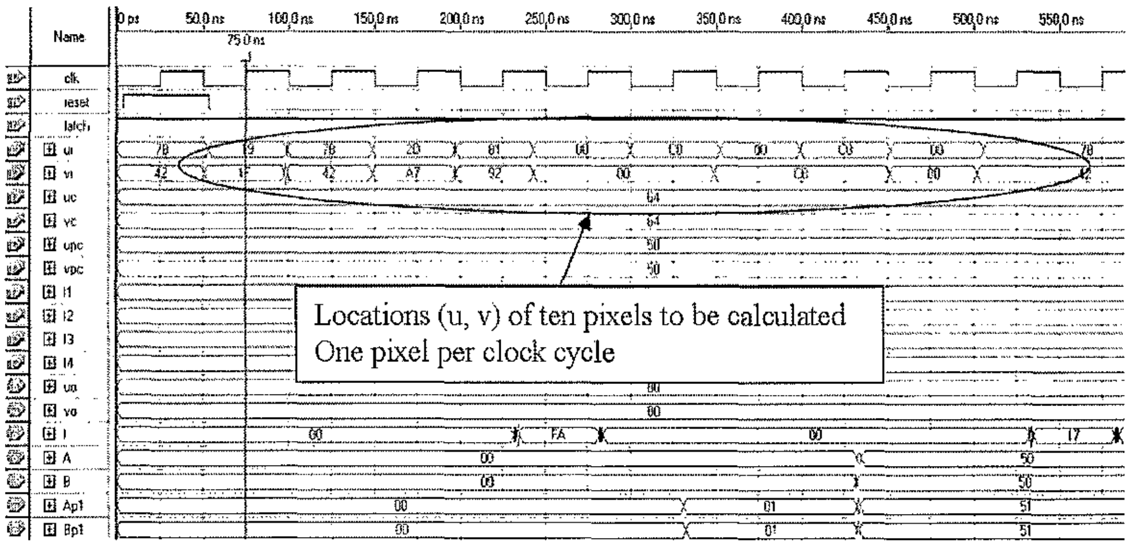
Random values extracted from an experimental image are used in the simulation to verify the results obtained from the distortion correction architecture. Table 5.2 shows the results of the theoretical calculations by software means and the results produced by the hardware architecture design to correct barrel distortion in wide angle camera images. Values of u and v columns in the table are the components of the rectangular coordinate (u,v) of the pixels in CIS. Values of A and B are used to determine the four neighboring pixels (A,B), $(A+1,B)$, $(A,B+1)$ and $(A+1,B+1)$ in DIS in the linear interpolation procedure. Values in the $I(u,v)$ column are the intensities of the pixel that are calculated. The results of A, B and $I(u,v)$ of both software calculation and hardware implementation are listed for comparison purposes. The small differences between the results obtained by software calculation and results obtained by hardware simulation are due to the difference in precision in data representations.

Table 5.2: Intensity values of some random pixels in corrected image obtained from software simulation results and hardware implementation results.

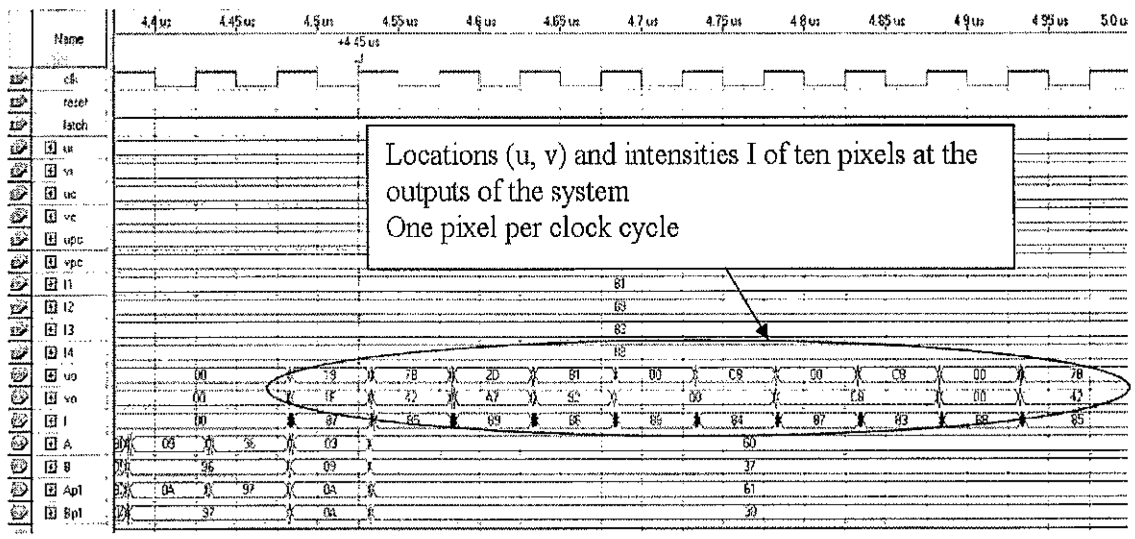
Input		**Output**					
		A		**B**		**I (u, v)**	
u	**v**	**SW**	**HW**	**SW**	**HW**	**SW**	**HW**
200	200	151	150	151	150	132	131
25	31	25	26	30	31	135	135
123	66	96	96	55	55	134	133
45	167	40	40	128	127	138	137
129	146	100	100	113	112	135	134



(a) Simulation results showing total latency of 89 cycles in the barrel distortion correction architecture.


Locations (u, v) of ten pixels to be calculated
One pixel per clock cycle
Name
clk
reset
latch
ui
vi
uc
vc
upc
vpc
I1
I2
I3
I4
uo
vo
I
A
B
Ap1
Bp1

(b) Simulation results showing ten pixel locations being inputted at the rate of one pixel per cycle.


Locations (u, v) and intensities I of ten pixels at the outputs of the system
One pixel per clock cycle

(c) Simulation results showing ten pixel locations and the intensities for each pixel calculated with the rate of one pixel per cycle.

Figure 5.2: Hardware simulation results of the distortion correction architecture.

Figure 5.2 shows the timing waveforms obtained from Quartus II software. As shown in Figure 5.2(a), the first pixel is presented to the system at time 0.725 ns and the calculated intensity for this pixel is available at the output at time 4.45 us, which is 89 cycles, the total latency of the system, where a clock cycle is 50 ns in the simulation. Figure 5.2(b) shows the enlarged section of the simulation where the input locations of the pixels are presented to the input of module 1. The input locations are represented in the form of rectangular coordinates, and the coordinates are shown as u_i and v_i in Figure 5.2(b). The enlarged section where the calculated outputs are available are shown in Figure 5.2(c). Values of u_o and v_o rows represent the coordinate of the pixels whose intensities are calculated, and the intensity values are shown as I row in Figure 5.2(c). It is noted the architecture calculates intensity for one pixel per cycle as expected.

For the improved architecture design with consideration of the symmetry in images, the entire architecture performs 4 times faster but it also consumes more resources because four interpolations sub-modules are needed to compute intensities for four pixels in CIS simultaneously. As shown in Table 5.3, the number of logic elements and the number of flip-lops used to implement the distortion correction architecture in the selected chip (EP20K1000EBC652-1X FPGA) form Altera, increase considerably compared to the earlier design. The improved architecture to correct barrel distortion in images can maintain the peak throughput of 120 1-Mpixel frames or 120 Mpixels per second compared to the throughput of 30 1-Mpixel frames or 30 Mpixels per second of the earlier design.

Table 5.3: Implementation results of the pipelined architecture to correct barrel distortion with consideration of the symmetry in images.

Name	**Result**
Chip Name	EP20K1000EBC652-1X FPGA
Maximum clock frequency	40 MHz
Logic elements	27,170
Flip flops	23,803
I/O power	0.10719 mW
Internal power	559.06 mW
Latency	93 cycles
RAM accesses	~7 ns
DIS RAMs refresh time (L pixels)	<= 7L ns
CIS processing time for 1 image (L pixels)	$\left(\frac{1}{4}L+92\right)$ cycles
Time to correct an 1-Mpixel image with clock frequency of 40 MHz.	~8.25 ms
Throughput of the correction system running with a 40 MHz clock to correct 1-Mpixel images	120 frames or 120 Mpixels per second

In Table 5.4, the results of hardware simulation are listed. Values in (u_1, v_1) rows are the locations of the pixels in quadrant I of the CIS that are the inputs to the systems. Values of (u_2, v_2), (u_3, v_3), and (u_4, v_4) rows are the locations of pixels in quadrant II, quadrant III and quadrant IV respectively that have the same level of distortion as that of pixel at (u_1, v_1). Values in (A_1, B_1), (A_2, B_2), (A_3, B_3), and (A_4, B_4) rows are calculated in the interpolation module, and they are used to determine the neighboring locations of the four pixels in four different quadrants. To simplify the simulation, all values for the intensities of the neighboring pixels are initialized to fixed numbers as:

I(A, B)	= 129
I(A+1, B)	= 137
I(A, B+1)	= 131
I(A+1, B+1)	= 139

Values in $I(u_1, v_1), I(u_2, v_2), I(u_3, v_3)$, and $I(u_4, v_4)$ rows are the intensities calculated for pixels (u_1, v_1), (u_2, v_2), (u_3, v_3), and (u_4, v_4) respectively. These results validate the expectation for the distortion correction system to provide a better throughput, which is 4 times higher than the throughput of the original design.

Table 5.4: Implementation results showing intensities of some random pixels with the consideration of image symmetry.

Input	(u_1, v_1)	(52, 62)	(31, 25)	(95, 15)
Output	(u_2, v_2)	(148, 62)	(169, 25)	(105, 15)
	(u_3, v_3)	(52, 138)	(31, 175)	(95, 185)
	(u_4, v_4)	(148, 138)	(169, 175)	(105, 185)
	(A_1, B_1)	(45, 52)	(30, 25)	(76, 18)
	(A_2, B_2)	(114, 52)	(129, 25)	(83, 18)
	(A_3, B_3)	(45, 107)	(30, 134)	(76, 141)
	(A_4, B_4)	(114, 107)	(129, 134)	(83, 141)
	$I(u_1, v_1)$	132	133	132
	$I(u_2, v_2)$	135	135	134
	$I(u_3, v_3)$	132	134	133
	$I(u_4, v_4)$	135	136	135

CHAPTER 6

CONCLUSION

Images captured by wide-angle camera show barrel type spatial distortion due to wide-angle configuration of the camera lens. Barrel distortion leads to inaccurate quantitative information in images because objects in the outer area of the image are shown in smaller size than they truly are. Thus, correction is needed to acquire more accurate quantitative parameters which are critical in many applications such as clinical endoscopy. Many mathematical models to correct distortion have been presented by researchers. Real time correction of barrel distortion in images requires a fast and efficient dedicated hardware architecture.

In this research, a method to correct barrel distortion in wide-angle camera images based on least squares estimation has been studied. The technique to select expansion coefficients and back-mapping coefficients has been explained and a C++ program was developed to apply this technique in selecting the correction coefficients. The algorithm to correct barrel distortion based on an expansion polynomial, a back mapping polynomial and linear interpolation has been presented in this thesis. A C++ program has been developed to simulate the correction algorithm, and the results were illustrated for the correction of a distorted image of an experimental grid and a typical endoscopic image.

An absolute pipelined architecture has been designed and implemented for the distortion correction algorithm presented in this research. The entire architecture consists of four modules to perform four main tasks in the distortion correction algorithm:

(1) Rectangular to polar coordinate transformation based on CORDIC algorithm

(2) Back mapping procedure based on the back mapping polynomial which maps a position of pixel in the corrected image space onto the corresponding position of the pixel in the distorted image space

(3) Polar to rectangular coordinate transformation based on CORDIC algorithm

(4) Linear interpolation procedure to calculate the intensity of a pixel based on the intensities of its four neighboring pixels

CORDIC algorithm has been used to implement the coordinate transformation modules because it employs only addition, subtraction, and bit shifting to perform the transformation.

Simulation results obtained from hardware implementation shows that the architecture is capable of processing one pixel per cycle while it is in the processing mode which is suitable for correcting barrel distortion in images in real time. At its peak performance, the architecture could sustain a throughput rate of 30 Mpixels per second, which corresponds to 30 1-Mpixel images per second. The performance of the architecture satisfies the required throughput for real-time applications which is 25 1-Mpixel images per second at the minimum. An improved architecture that takes into account the symmetry in images has also been presented. It is capable of processing four pixels in four different quadrants of the image simultaneously. The simulation results also show that the improved architecture could maintain a throughput rate of 120 Mpixels per second, which is four times faster than the original design. The performance achieved from the distortion correction architecture fulfils the objective of this research which is to design an architecture to correct barrel distortion in real time.

The current research for further improvement of the architecture is focused on the power consumption aspect of the architecture. One approach that is being studied is to incorporate the technique of uni-directional CORDIC presented by Ravichandran and Asari [33] into the CORDIC modules in the designed architecture. This technique modifies the traditional CORDIC algorithm so that the vector rotates only in one direction, thus the name uni-directional CORDIC. This approach would reduce the number of iterations considerably because the vector rotates only when it is needed rather than the vector rotates in every iteration as in the traditional CORDIC algorithm. The reduction in the number of rotations would ultimately reduce the amount of dynamic power consumed by the distortion correction architecture, which is desirable for mobile systems.

REFERENCES

[1] N. Greeen, "Environment Mapping and Other Applications of World Projections" *IEEE Comp. Graphics and Application,* vol. 6, no. 11, pp. 21-29, 1986.

[2] G.E. Karass, G. Mountrakis, P. Patias, and E. Patias, "Modeling Distortion of Super-Wide-Angle Lenses for Architectural and Archaeological Applications" *International Archives of Photogrammetry & Remote Sensing*, 32(5):570-573, 1998.

[3] G. Wei, K. Arbter and G. Hirzinger, "Real Time Visual Servoing for Laparoscopic Surgery," *IEEE in Engineering and Biology*, pp. 40-45, Jan. 1997.

[4] A. Sonnenberg, M. Giger, L. Kern, C. Noll, K. Stuby, K. B. Weber, and A. L. Blum, "How Reliable is Determination of Ulcer Size by Endoscopy," *Brit. Med. J.*, vol. 24, pp. 1322-1324, 1979.

[5] C. Margulies, B. Krevsky, and M. F. Catalano, "How Accurate Are Endoscopic Estimates of Size," *Gastrointestinal Endoscopy*, vol. 40, pp. 174-177, 1994.

[6] R. Y. Tsai, "An Efficient and Accurate Camera Calibration Technique for 3D Machine Vision," *Proc. IEEE Comp. Vision Patt. Recogn*, Miami, FL, pp. 364-374, 1986.

[7] Y. Nomura, M. Sagara, H. Naruse, and A. Ide, "Simple Calibration Algorithm for High-Distortion-Lens Camera," *IEEE Trans. Patt. Anal. Machine Intell*, vol. 14, no. 11, pp. 1095-1099, 1992.

[8] J. Weng, P. Cohen, and M. Herniou, "Camera Calibration with Distortion Models and Accuracy Evaluation," *IEEE Trans. Patt. Anal. Machine Intell.*, vol. 14, no. 10, pp. 965-980, 1992.

[9] M. Classen, H. Dancygier, and D. Wurbs, "New Method of Endoscopic Determination of Ulcer Area," *Gut.*, vol. 21, pp. 895, 1980.

[10] H. Okabe, M.Ohida, N. Okada, T. Mitsuhashi, T. Katsumata, K. Saigengi, and K. A. Nakahashi, "New Disc Method for the Endoscopic Determination of Gastric Ulcer Area," *Gastrointest. Endoscopy*, vol. 32, pp. 895, 1980.

[11] W. E. Smith, N. Vakil, and S. A. Maislin, "Correction of Distortion in Endoscopic Images," *IEEE Trans. Med. Imaging*, vol. 11, no. 1, pp. 117-122, 1992.

[12] H. Hideaki, Y. Yagihashi and Y. Miyake, "A New Method for Distortion Correction of Electronic Endoscope Images," *IEEE Trans. Med. Imaging*, vol. 14, no. 3, pp. 548-555, 1995.

[13] K. V. Asari, S. Kumar, and D. Radhakrishnan, "A New Approach for Non-linear Distortion Correction in Endoscopic Images based on Least Squares Estimation," *IEEE Trans. Med. Imaging,* vol. 18, no. 4, pp. 345-354, 1999.

[14] J. P. Helferty, C. Zhang, G. McLennan, and W. E. Higgins, "Videoendoscopic Distortion Correction and Its Application to Virtual Guidance of Endoscopy," *IEEE Trans. Med. Imaging*, vol. 20, no. 7, pp. 605-617, July 2001.

[15] S. D. Haynes, J. Stone, P. Y.K. Cheung, and W. Luk, "Video Image Processing with Sonic Architecture," *IEEE Computer*, pp. 50-57, Apr. 2000.

[16] P. Ranganathan, S. Adve and N. P. Jouppi, "Performance of Image and Video Processing with General-Purpose Processors and Media ISA Extensions," *Proc. of the 26th annual Inter. Symp. on Comp. Arch.*, pp. 124 – 135, May 1999.

[17] D. Pearson, *Image Processing*, United Kingdom: McGraw-Hill Book Company, 1991.

[18] V. L. Do and K. Y. Yun, "A Low-Power VLSI Architecture for Full-Search Block-Matching Motion Estimation" *IEEE Trans. Circuits and Systems for Video Tech.*, vol. 8, pp. 393-398, Aug. 1998

[19] J. E. Eklund, C. Svensson and A. Astrom, "VLSI Implementation of a Focal Plane Image Processor – A Realisation of the Near-Sensor Image Processing Concept," *IEEE Trans. VLSI Systems,* vol. 4, pp. 322-335, 1996.

[20] D.W. Hammerstrom, D.P. Lulich, "Image Processing using One-dimensional Processor Arrays", *Proc. IEEE*, vol. 84, no. 7, pp. 1005-1018, 1996.

[21] N. Hemkumar, K. Kota and J. Cavallaro, "CAPE - VLSI Implementation of a Systolic Processor Array: Architecture, Design and Testing,"

University/Government/Industry Microelect. Symp., Melbourne, FL, pp. 64-69, Jun. 1991.

[22] K. V. Asari and C. Eswaran, ″Systolic Array Implementation of Artificial Neural Networks″, *Journal of Microprocessors and Microsystems*, vol. 18, no. 8, pp. 481-488, 1994.

[23] M. J. Flynn, *Computer Architecture: Pipelined and Parallel Processor Design*, United States: Johns and Bartlett Publishers, Inc., 1995.

[24] S. G. Shiva, *Pipelined and Parallel Computer Architectures,* United States: HapperCollins Publishers, Inc., 1996.

[25] "White Paper for QuantiSpeed Architecture," Advanced Micro Devices, Inc., Sep. 2001.

[26] K. V. Asari, T. Srikanthan, S. Kumar, and D. Radhakrishnan, "A Pipelined Architecture for Image Segmentation by Adaptive Progressive Thresholding," *Journal of Microprocessors and Microsystems*, vol. 23, no. 8-9, pp. 493-499, 1999

[27] J. D. Bruguera, N. Guil, T. Lang, J. Villalba and E. L. Zapata, "CORDIC Based Parallel/Pipelined Architecture for the Hough Transform," *Journal of VLSI Signal Processing*, Jan. 2001.

[28] D. A. Berry and B. W. Lindgren, *Statistic: Theory and Methods.* CA: Brooks/Cole, 1990.

[29] K. V. Asari, "Nonlinear Spatial Warping of Endoscopic Images: an Architectural Perspective fro Real Time Applications," *Journal of Microprocessors and Microsystems*, vol. 26, pp. 161-171, 2002.

[30] J. Volder, "The CORDIC Trigonometric Computing Technique," *IRE Trans. Electronic Computing,* vol. EC-8, pp. 330-334, 1959.

[31] R. Andraka, "A survey of CORDIC algorithm for FPGAs," *Proc. ACM/SIGDA* 98, pp. 191-20, Feb. 1998.

[32] H. T. Ngo and K. V. Asari, "Design of an Efficient VLSI Architecture for Real-Time Correction of Barrel Distortion in Wide-Angle Camera Images," *10th NASA Symp. on VLSI Design*, Albuquerque NM, 8.5.1-8.5.7, Mar. 2002.

[33] S. Ravichandran and K. V. Asari, "Implementation of Unidirectional Cordic Algorithm Using Precomputed Rotation Bits, " *Midwest Symp. on Circuits and Systems - MWSCAS 2002* , Tulsa, OK USA, Aug. 4-7, 2002 .

CURRICULUM VITA
for
Hau Trung Ngo

DEGREES:

Master of Science (Computer Engineering), Old Dominion University, Norfolk, Virginia, May 2003

Bachelor of Science (Computer Engineering), Old Dominion University, Norfolk, Virginia, May 2001

Associate in Science (Engineering), Tidewater Community College, Virginia Beach, Virginia, May 1999

SCHOLARSHIPS, HONORS AND AWARDS

CSEM Scholarship
Kovner Scholarship
Technology Scholarship
Outstanding MS Research Assistant Award, 2003
Dominion Scholar, 2002
National Dean List, 2001
Top Senior Award in Computer Engineering, 2001
Faculty Award in Computer Engineering, 2001

RESEARCH PUBLICATIONS

H. T. Ngo and K. V. Asari, "A Fully Pipelined Architecture for Barrel-Distortion Correction Based on Back Mapping and Linear Interpolation," *The 2003 Intl. Conf. on Embedded Systems and Applications - ESA'03, (Methodologies in Low Power Design - MLPD'03)*, Las Vegas, Nevada, USA, June 23-26, 2003. (accepted for presentation).

M. J. Seow, H. T. Ngo, and K. V. Asari, "Systolic array implementation of block based Hopfield neural network for pattern association," *Proc. of the IEEE Computer Society Annual Symp. on VLSI - ISVLSI 2003*, Tampa, Florida, USA, pp. 213-214, February 20-21, 2003.

H. T. Ngo and K. V. Asari, "Design of an Efficient VLSI Architecture for Real-Time Correction of Barrel Distortion in Wide-Angle Camera Images," *Proc. of 10th NASA Symp. on VLSI Design*, Albuquerque, New Mexico, USA, pp. 8.5.1-8.5.7, March 20-21, 2002.

M.J. Seow, H. T. Ngo, and K. V. Asari, "Systolic implementation of 2-D block based Hopfield neural network for efficient pattern association," *Journal of Microprocessors and Microsystems* (in print).

H. T. Ngo and K. V. Asari, "A Pipelined Architecture for Real-Time Correction of Barrel Distortion in Wide-Angle Camera Images," *IEEE Trans. on Circuit and System for Video Technology* (revised version submitted).