Techniques for detecting camera movements during medical endoscopy

Suchitra Sathyanarayana

2007

Suchitra, S. (2012). Techniques for detecting camera movements during medical endoscopy. Master's thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/35737>

<https://doi.org/10.32657/10356/35737>

9530275

Techniques for Detecting Camera Movements during Medical Endoscopy

Suchitra Sathyanarayana

School of Computer Engineering

A thesis submitted to the Nanyang Technological University
in fulfillment of the requirement for the degree of
Master of Engineering

2007

Contents

Acknowledgments	vii
Abstract	ix
Publications resulting from this work	xi
1 Introduction	1
1.1 Medical video endoscopy	1
1.1.1 A typical video endoscope system	3
1.1.2 Wireless capsule endoscopy	5
1.1.3 Virtual endoscopy	7
1.2 Image processing on endoscopic images	10
1.3 Image processing <i>during</i> endoscopy	14
1.3.1 Revisiting tissues of interest	16
1.3.1.1 Changed working conditions during revisit	17
1.3.2 False negatives in diagnosis	18
1.4 Problem statement: Real-time camera motion tracking	19
1.5 Summary	20
2 Tracking camera motion	22
2.1 Existing methods to track camera motion	22
2.2 Problem scope definition	23
2.3 Outline of envisaged approach	26
2.4 Identifying computational blocks	29
2.5 Experimental Setup	30
2.6 Summary	32
3 High speed image rotation	33
3.1 Exhaustive rotation approach	33
3.1.1 Proof of concept	34
3.1.2 Need for high speed rotation engine	36
3.2 Existing hardware for image rotation	37
3.2.1 LUT-Based Implementation	38
3.2.2 CORDIC-Based Implementation	39
3.2.2.1 Pixel-by-Pixel CORDIC-based Rotation	39

	Pipelined-Parallel Architecture	40
3.3	Proposed Image Rotation Technique	41
3.3.1	Hierarchical Approach for Generating Rotated centers	42
3.3.2	Exploiting Symmetrical Characteristics to Reduce CORDIC Computations	43
3.3.3	Generation of the Pixel Offsets	44
3.3.4	Illustration of proposed method	44
3.4	Rotation system overview	47
3.4.1	Accuracy analysis of the proposed system	49
	Interpolation schemes	49
	Accuracy of rotated pixel position	50
	Choice of data width	52
3.5	Extending to higher resolution images	52
3.6	Image rotation engine: Comparative analysis	54
3.6.1	Computational complexity	55
3.6.2	Latency comparison	55
3.6.3	Area comparison	57
3.6.4	Derivation of optimal hierarchy	58
3.6.5	Latency estimate comparison for varying hierarchies	61
3.6.6	Effect of changing hierarchy on Area	61
3.7	Application to Endoscopy	63
3.8	Summary	64
4	Proposed method for measuring linear motion of camera	66
4.1	Image pre-processing	67
4.1.1	Pre-processing using wavelets	67
4.2	Feature point extraction and tracking	68
4.2.1	Benefits of wavelets for FP tracking	71
4.3	Analysis of FP displacement: Estimation of linear camera motion . .	71
4.3.1	Image unrolling process	73
4.3.2	Computations involved in image unrolling	75
4.3.3	Normalization	76
4.3.4	Verification of distance tracking	79
4.4	Extending method to detect camera rotation	81
4.4.1	Verification of rotation tracking	83
4.5	Results	86
4.5.1	Results: Image pre-processing	86
4.5.2	Results: Feature point extraction	87
4.5.3	Results: Feature point tracking	88
4.5.4	Results: Image unrolling	89
4.5.5	Results: Normalization	90
4.5.6	Results: Combined method for tracking linear motion and ro- tation	91
4.5.7	Effect of radius and FOV parameters	93
4.6	Discussion	95

4.7	Summary	96
5	Automated unrolling	98
5.1	Proposed method for center estimation	98
5.1.1	Relationship between border displacement estimates	99
5.1.2	Conditions for Convergence	102
5.1.3	Algorithm for center estimation	104
5.2	Results on real images	105
5.2.1	Center estimation	106
5.2.2	Overall algorithm on image sequence	107
5.3	Integrated framework for tracking linear motion and rotation	111
5.4	Summary	112
6	Conclusions and future work	114
6.1	Conclusions	114
6.2	Scope for future work	115
	Bibliography	118
	Appendix: Published Papers	i

List of Figures

1.1	Healthy and cancerous esophagus snapshots during endoscopy	3
1.2	A typical endoscope system and camera-based endoscope	4
1.3	A capsule endoscope	5
1.4	Example of graphics based organ rendering: Graphics rendered liver against original image	8
1.5	Virtual endoscopy tool at Harward Medical School	9
1.6	Sample results of image recification	13
1.7	Example of image enhancement in endoscopy using hemoglobin index	14
1.8	Example of image segmentation on an endoscopic image	15
2.1	Degrees of motion of camera through a cylindrical tract	25
2.2	Snapshot of image sequence during endoscopy	27
2.3	Movement of feature points when camera moved upward	28
2.4	Outline of proposed research steps	29
2.5	Experimental setup	31
2.6	Example of laboratory simulated image	32
3.1	Detection of camera rotation	34
3.2	Steps in ‘exhaustive rotation approach’	35
3.3	Example of rotation detection	35
3.4	Correlation versus angle of rotation	36
3.5	Illustration of Rotation Method in [53]	41
3.6	Hierarchical rotation of centers	42
3.7	Inferring rotated pixel value based on image symmetry	44
3.8	Offset computation	45
3.9	Layer centers with hierarchy 2	45
3.10	System overview of proposed image rotation engine	48
3.11	Rotation position accuracy versus operand length	51
3.12	Higher datawidth to hide extra quantization errors	53
3.13	Area, latency, computational complexity comparisons	56
3.14	Latency expression as function of hierarchy	59
3.15	Choice of hierarchy	60
3.16	Optimal hierarchy	60
3.17	Gain of proposed method over existing method	62
3.18	Effect of changing hierarchy on area	62
3.19	Rotation technique applied to endoscopy images	64

4.1	Estimation of camera motion through FP movement tracking	67
4.2	Inconsistency in displacement magnitude	73
4.3	Image Unrolling Concept	74
4.4	Unrolling transformation	75
4.5	Expressing depth as a function of camera angle	77
4.6	Angular view with respect to camera	78
4.7	Image sequence for detection of linear movement	80
4.8	Movement of feature points in different domains	83
4.9	Image sequence for rotation detection	84
4.10	Computational blocks for linear and rotation movement estimation .	86
4.11	Edge extraction using wavelets	87
4.12	Feature point extraction	88
4.13	Feature point tracking using correlation search	89
4.14	Image unrolling experiment	90
4.15	Distance Estimate versus Field of View	94
5.1	Experiment to derive relationship between estimated center and ref- erence center, Q: reference center, A to P: assumed centers (a) initial image (b) final image	99
5.2	(a) image1; (b) image2	103
5.3	Graph showing convergence of border outputs when $C_{assumed} \approx C_{estimate}$	103
5.4	Test image considered for center estimation algorithm	106
5.5	Processing alternate frames	110
5.6	Processing every 3rd frame	110
5.7	Overall research contribution	112

List of Tables

3.1	Illustration of proposed rotation method	46
4.1	Inferring forward distance of camera	81
4.2	Inferring rotation of camera	85
4.3	Estimation of camera motion on image sequence	92
4.4	Tracking Camera Distance for varying FOV and Radius of Cylinder .	93
5.1	Relationship between border outputs	101
5.2	Estimation of center	106
5.3	Results on consecutive frames	108
5.4	Results on alternate frames	109
5.5	Results on every third frame	109

Acknowledgments

I would like to express my heartfelt gratitude to my supervisor, **Professor T. Srikanthan** for his expert guidance, constant support and unceasing encouragement that were vital for research and development of this thesis. The numerable discussions that we have had regarding various related research problems have proved to be invaluable in making the whole experience very fulfilling. His constructive criticism and high expectations have driven me to bring about quality and credibility to the work. I am greatly indebted to him for also providing a conducive research atmosphere and all the latest facilities at Center for High Performance Embedded Systems (CHiPES).

My deep gratitude to **Dr. Christopher Clarke** from the University of Bath, UK, who has been a great source of inspiration for me. I owe a lot to him for his creative inputs which prompted me to explore new ideas and novel methods during the course of this Masters work.

I would also take this opportunity to thank **Dr. Sriparan**, who provided expert suggestions to improve the work, from a medical viewpoint. He also gave invaluable inputs in terms of related papers and documents, which strengthened the applicability of the work for medical endoscopy.

Special thanks to my project-mate and husband, **Ravi**, who has been with me throughout the project. He has given enormous support, confidence and encouragement at every step, apart from helping me fine tuning and proofreading the thesis. It would be incomplete without thanking **Swamiji**, who has been a tremendous source of inspiration, without whom it would not have been possible for me to complete this thesis.

Thanks also to my fellow researchers and colleagues, who have made my stay at CHiPES memorable and enjoyable. I extend my thanks to the lab technicians **Ms. Boh-Nah Kiat Joo** and **Mr. Jeremiah Chua Ngee Tat**, who have supported me in one way or the other towards completion of this thesis.

Finally, I would like to acknowledge School of Computer Engineering, Nanyang Technological University for providing me with this opportunity for pursuing this Masters degree.

Abstract

A medical endoscope is a tool commonly used for medical diagnostic and treatment purposes. A lens-based endoscope consists of a camera mounted on a tube, that is inserted into the human tract. Images captured by endoscopes facilitate examination and treatment of regions inside the patient. These images are commonly subject to image processing to either enhance or segment out a region of interest for better decision making, by employing a wide range of image processing algorithms. However, not much has been done to process the video stream on the fly, during the endoscopic procedure due to computational complexities of these image processing algorithms.

In this thesis, techniques for real-time processing of endoscopic images are proposed for tracking of camera motion under three categories - rotations only, linear-motion only and an integrated rotation and linear motion tracking. To aid in tracking camera rotations, hardware efficient techniques for achieving high throughput image rotation are proposed. These exploit symmetrical characteristics of the image to accelerate the rotation of high-resolution images. The effect of hierarchy on the area-time computational efficiency has also been reported to provide for design flexibility. Investigations based on a 512×512 image show that the proposed methods yield about 20 times speedup at a mere 3% increase in area cost when compared to existing techniques. The proposed techniques are also highly scalable and significant performance gains are evident for very high-resolution images.

To track linear movement of camera through cylindrical tracts, techniques based on the analysis of feature point movements are proposed. The proposed techniques involve feature point extraction, tracking, image unrolling and normalization in order to facilitate the necessary analysis in the unrolled domain. The methods for linear motion detection were further extended to detect rotation, thereby allowing for an integrated tracking of both the linear distance and rotation. The proposed techniques have been evaluated on a sequence of test images to demonstrate that both the linear distance and rotation can be inferred using a small number of feature points, thereby justifying its use in real time systems.

Next, techniques to automate the image unrolling process are presented. The proposed techniques are shown to be capable of performing the unrolling at high speed through an iterative process, which relies on the rapid estimations of the coordinates of the center. Simulations based on real images confirm that only a few iterations are required for convergence, justifying the applicability of this method in high speed realizations. The proposed automated unrolling process has been incorporated into the combined rotation and linear-motion detection system. It was shown that this integrated environment performs well with only a few intermediate frames of real endoscopic images. Finally, an overall framework for the integrated environment to track the linear and rotation movements of the endoscopic camera is proposed to present the various operating scenarios.

Publications resulting from this work

Journals

- S. Suchitra, C. T. Clarke, T. Srikanthan, S. K. Lam, “Accelerating rotation of high-resolution images”, *IEE Proceedings on Vision, Image and Signal Processing*, Vol. 153, No. 6, pp. 815-824, 2006.
- S. Suchitra, T. Srikanthan, C. T. Clarke, “Techniques for detecting camera movements during medical endoscopy”, *IEEE Transactions on Image Processing*, 2006. (Submitted)

International Conferences

- S. Suchitra, S. K. Lam, T. Srikanthan, “Novel Schemes for High-Throughput Image Rotation” , *Proc. of 38th Annual Asilomar Conference on Signals, Systems, and Computers*, pp.1884-1888, 2004.
- S. Suchitra, S.K. Lam and T. Srikanthan, “High-Throughput Image Rotation using Sign-Prediction based Redundant CORDIC Algorithm”, *Proc. of IEEE International Conference on Image Processing*, pp. 2833-2836, 2004.
- S. Suchitra, C. S. Lim and T. Srikanthan, “Array based Architecture for EZW Image Encoding on FPGA using Handel-C”, *Proc. of 38th Annual Asilomar Conference on Signals, Systems, and Computers*, pp.447-450, 2004.

- S. Suchitra, T. Srikanthan and C. T. Clarke, "Real time tracking of camera motion through cylindrical passages", *Proc. of 15th International Conference on Digital Signal Processing*, pp. 455 - 458, 2007.

Chapter 1

Introduction

Motivation

Medical endoscopy is widely used in the medical arena to facilitate examination and treatment of parts of the human body, not otherwise easily accessible. The videos captured by the endoscope are analyzed for decision making by the surgeons. Digital image processing has not only made objective interpretations of the endoscopic videos possible, but has also facilitated automation of the analysis. Most of the processing of the captured videos is currently done off line after the medical procedure. However, there is a need for processing the video on the fly to aid the surgeon in real time.

A survey of the existing state of the art in the context of medical endoscopy and image processing on endoscopic images has shown that there is a need to track the motion of the endoscopic camera in real time. In this thesis, efficient techniques to estimate the linear and rotation movements of an endoscopic camera are explored.

1.1 Medical video endoscopy

The word endoscopy is derived from the Greek words ‘endo’ meaning ‘inside’ and ‘skopeein’ meaning ‘to see’ [1]. It is a word used in medicine to describe the pro-

cedure that is used to see various parts inside the body. Endoscopy has become an indispensable tool for various applications in medicine, particularly for the purpose of minimally invasive diagnostic and therapeutic procedures. An endoscope is intended to facilitate both the examination and treatment of regions inside the human body by inserting a small tube containing a viewing channel, an illumination channel and eventually one or more ancillary channels which cause minimal trauma.

There are broadly two classes of optical endoscopes in clinical use. The first, called optic-fiber endoscope, consists of a long flexible fiber optic coupling between the remote site to be investigated and the user. These give adequate diagnostic information, although the image quality may be compromised by the number of elements within the bundle. An alternative class of endoscope uses a thin tube enclosing an appropriate lens system. These instruments are optically superior to their fiber bundle counterparts and this approach is called video endoscopy. [2] (1985) gives a description of video endoscopy as follows: "The video system represents a new endoscopic technique with major advantages, some of which point the way into the future. This system permits a large number of persons to participate directly in the examination. Documentation is more comprehensive and more reliable, and pathological processes can be observed with the aid of video tape recordings." In [3], a comparative evaluation of a video endoscope is made against the conventional fiber optic method and details the benefits of the former. Video endoscopy gave better endoscopic images for determination of the limits and of the size of cancer invasion.

Medical endoscopy is further divided into categories based on the organ under consideration. For example, *colonoscopy* involves examination of the insides of the colon and large intestine to detect polyps, tumors, ulceration, inflammation, and discovery and removal of foreign bodies [4]. Similarly, *bronchoscopy* refers to examination

of the trachea and lung's bronchial trees to reveal abscesses, bronchitis, carcinoma, tumors, tuberculosis, alveolitis, infection and inflammation. Typical images as captured by the endoscopic camera during an *esophageal endoscopy* procedure are shown in Figure 1.1.

Figure 1.1: Healthy and cancerous esophagus snapshots during endoscopy

1.1.1 A typical video endoscope system

Typically, in a video endoscopic procedure, a video camera and strobe light are utilized to capture images of internal organs. The procedure allows the physician to perform careful visual inspection on the entire lining surface of the organ being examined. If a suspicious image is detected, e.g. suspicious bleeding or tumor, a little piece of the corresponding tissue is taken for biopsy. Further, endoscopy is frequently used to remove polyps, or small growths within the tract or for general treatment purposes. A typical endoscope system used in present clinical procedures is shown in Figure 1.2 [5].

Many interactive endoscopes have been devised to give the surgeon some form of feedback during navigation. Endoscopes have external control knobs that control the movement of the scope inside the tract. [6] describes this control mechanism as

Figure 1.2: A typical endoscope system and camera-based endoscope

follows: Modern endoscopes include means for deflecting the distal tip of the scope to follow the pathway of the structure under examination, with minimum deflection or friction force upon the surrounding tissue. In a conventional endoscope design, mechanical control of the deflectable tip is exerted via control cables similar to bicycle brake cables that are carried within the endoscope body in order to connect a flexible portion of the distal end to a set of control knobs at the proximal endoscope handle. The examiner mechanically steers the distal tip of the endoscope to a region of interest by manipulating the control knobs. The control knobs can be locked in place once a desired position is gained. While manually turning the control knobs, the examiner receives direct feedback regarding the force required to change the position of the tip.

On the aspect of the computer-controlled endoscope development, Buckingham *et Al.* demonstrated that a computer controlled endoscope is both useful, feasible and can be designed to be safe [7]. The proposed software task can be broken down into the identification of a safe trajectory within a complex environment and the control of a highly redundant device to follow this path. Dogramadzi developed a new computer-controlled endoscope, which uses air actuation of a multi-link mechanism with sensory feedback provided by a 3D magnetic sensor placed in the biopsy channel

of a gastroenterological endoscope [8]. Hoeg *et Al.* developed a prototype robotic endoscope for small intestine diagnosis and therapy [9].

1.1.2 Wireless capsule endoscopy

In 1981, an Israeli physician, Dr. Gavriel Iddan, began development of a video camera that would fit inside a pill. Technology was not ready then and the idea was put on hold. It took 20 years for technology to catch up with Dr. Iddan. In 2001, the FDA approved the Given Diagnostic Imaging System. This 11 x 26 mm capsule weighs only 4 gms (about 1/7th of an ounce) and contains a color video camera and wireless radiofrequency transmission, 4 LED lights, and enough battery power to take 50,000 color images during an 8-hour journey through the digestive tract. A picture of a capsule endoscope is shown in Figure 1.3. About the size of a vitamin capsule, the capsule is made of specially sealed biocompatible material that is resistant to stomach acid and powerful digestive enzymes. Another name for this new technique is Wireless Capsule Endoscopy [10].

Figure 1.3: A capsule endoscope

Patients report that the video capsule is easier to swallow than an aspirin. It seems that the most important factor in the ease of swallowing is the lack of friction. The capsule is very smooth, enabling it to slip down the throat with just a sip of water [10]. After the video capsule is swallowed, it moves through the digestive track naturally

with the aid of the peristaltic activity of the intestinal muscles. The patient comfortably continues with regular activities throughout the examination without feeling sensations resulting from the capsule's passage. During the 8 hour exam, the images are continuously transmitted to special antenna pads placed on the body and captured on a recording device which is worn about the patient's waist. After the exam, the recording device is removed. The stored images are transferred to a computer PC workstation where they are transformed into a digital movie which the doctor can later examine on the computer monitor. Patients are not required to retrieve and return the video capsule to the physician. It is disposable and expelled normally and effortlessly with the next bowel movement.

The wireless capsule endoscope has been successfully utilized to diagnose diseases of the small intestine and alleviate the discomfort and pain of patients [13]. This technique is relatively new, but numerous reports have already demonstrated cases in which the given video capsule was able to make a diagnosis not seen by conventional studies. Although capsule endoscopy has been shown to be superior to conventional endoscopy [11], it cannot achieve 100% accuracy in diagnosis [10] [12]. Possible reasons for this are discussed in [11], which include inability to definitively localize or treat small lesions and the inherent inaccuracies in the method for approximating capsule location due to variations in anatomy. Also, the resolution of the images is still low, and some interesting spots may be unintentionally omitted [11] [13]. The images will be severely distorted especially when physicians zoom images in for detailed diagnosis.

Increasing resolution may cause significant power consumption in RF transmission; hence, image compression is necessary for saving the power dissipation of RF transmission. To overcome this drawback, Lin *et Al.* have been developing a new capsule

endoscope, called GICam [13]. They developed an ultra-low-power image compression processor for capsule endoscope or swallowable imaging capsules. In applications of capsule endoscopy, it is imperative to consider battery life/performance trade-offs. Applying state-of-the-art video compression techniques may significantly reduce the image bit rate by their high compression ratio, but they all require intensive computation and consume much battery power. There are many fast compression algorithms for reducing computation load; however, they may result in distortion of the original image, which is not good for use in the medical care. [13] shows how to first simplify traditional video compression algorithms and it proposes a scalable compression architecture. The developed video compressor on FPGA reduces power consumption from 33.5 mW to 14.92 mW by reducing the video size by at least 75%.

1.1.3 Virtual endoscopy

Virtual endoscopy is the navigation of a virtual camera through a 3D reconstruction of a patient's anatomy enabling the exploration of the internal structures to assist in surgical planning. According to [14], the principal aim of virtual reality technology is to present virtual objects or complete scenes to all human senses in a way identical to their natural counterpart. Simulating three dimensional reconstruction of organs from radiological cross sections is an important diagnostic tool that provides clinicians with a more natural view of a patients anatomy. Also, preoperative planning with the use of a computer, including realistic prediction of the outcome, has become an integral part of the intervention in certain disciplines, such as radiation therapy, craniofacial surgery, or neurosurgery [14].

Computerized three dimensional atlases of human anatomy, physiology, and pathology are used as teaching aids for these subjects. Several virtual reality systems have been developed and tested for the physical or mental rehabilitation of patients

and for supporting mental health therapy [14]. Virtual reality technology also plays an important role in telemedicine from remote diagnosis to complex teleinterventions.

Algorithms for graphical rendering of anatomy have been used to provide support for three dimensional organ reconstruction from radiological cross sections [14]. For the clinician this method of visualization provided a more natural view of a patients anatomy without losing the see through capability of the radiologist. Virtual endoscopy techniques (such as virtual colonoscopy or bronchoscopy) based on the virtual reconstruction and visualization of individual patient anatomy are rapidly developing. For example Figure 1.4 shows graphics rendering of liver using virtual endoscopy. The most impressive development has been demonstrated in virtual colonoscopy as a screening tool for colon polyps and cancer and which is currently in the clinical validation phase. Owing to the potential benefits of patient comfort and cost effectiveness, it sometimes appears that virtual endoscopic procedures could replace real endoscopic investigations in the foreseeable future in some areas of diagnosis. But virtual endoscopy carries with it, its own set of disadvantages. It is usually an off-line process and not suitable for treatment. Also, it is usually accompanied by high computational costs.

Figure 1.4: Example of graphics based organ rendering: Graphics rendered liver against original image

There are many virtual endoscopy tools based on virtual reality and computer graphics. For example, a group at Harvard Medical School [15] has been developing a virtual endoscopy tool with which the surgeon can visualize the 3D model of an anatomical structure and define a trajectory path inside the model in order to perform a virtual exploration (shown in Figure 1.5). When the virtual camera flies through the model, the surgeon can track the position of the virtual camera inside the model on one screen, view what the endoscopic camera sees on another screen and track the position of the camera on gray scale 2D slices. Virtual exploration through patient-specific data can help the surgeon perform diagnosis in a non-invasive manner once the model has been built.

Figure 1.5: Virtual endoscopy tool at Harvard Medical School

1.2 Image processing on endoscopic images

In the past, a lot of analysis and decision making during endoscopy was subjective and depended on the experience of the surgeon. For example, as pointed out in [5], the commonly used diagnostic procedure to detect colorectal cancer was endoscopy performed by an expert physician and the images examined visually and qualitatively. However, the human eye is particularly poor at discriminating color or brightness features in images. Thus, not only is the analysis subject to individual interpretation, but, the absence of quantitative measurements obviates the strict activity of comparative evaluation, quantitative assessment and computer-aided diagnosis. The experience and knowledge of the endoscopist were also important. Thus new inexperienced physicians used be at a disadvantage.

With the use of digital image processing and analysis on endoscopic images, more objective interpretations, together with quantitative measurements has become possible. Therefore, computer-assisted image processing has started to become essential to assist decision making after endoscopic procedures. It provides essential critical features for confirmation, comparison and evaluation of suspicious signs during diagnosis and enables easy finding of these areas of interest during treatment. Also, computer-assisted analysis procedures have the capability of detecting abnormalities from endoscopic images with high accuracy and this has helped non-experts.

Image processing is considered to hold great potential for future applications like detection of very early stages of lesion [5]. As the image obtained is in the form of electronic signals, it can be manipulated, enhanced, transmitted, and merged with an entire array of other electronic media and databases using modern technology. One of the technical advances made possible by video-endoscopes is the application of software processing of digital images and hardware improvement.

Image processing techniques have been applied in many areas in the medical field and extended to clinical diagnosis and therapy [16]. Imaging technologies such as magnetic resonance imaging (MRI) and computer tomography [17] [18] [19] are very popular today in the context of medical imaging. In the area of endoscopic image processing and analysis, it is found that a lot of research work has been focused on the detection of colon lumen based on chromatic and spatial feature characterization. Lumen-related parameters are not only significant in the application of navigation of the endoscope in human colon, but also in the detection of abnormalities. Colon lumen detection includes lumen boundary extraction, lumen center orientation, etc. Fuzzy logic algorithms based on the assumption that the lumen is a large dark region has been adopted to extract the lumen boundary and detect the lumen center [20].

A method based on frequency domain to detect the lumen was developed in [21], which avoids the above assumption but employs a computationally intensive operation of discrete Fourier transform. A method by using Canny edge detection algorithm was proposed to identify the colon lumen and extracting the largest darkest region by thresholding method and to find the lumen center [23]. Krishnan presented Adaptive Progress Thresholding (APT) technique for lumen detection [22]. A minimum variance region-growing algorithm based on image histogram analysis was proposed for lumen boundary extraction in [24]. Tian *et Al.* proposed a technique by repeating an Otsu-Iris filter combination in an iterative manner, which arrived at rapid identification of the lumen region accurately in [25].

Color images can provide additional information about the physiological and anatomical status of the organ. Utilization of multiple color components in the endoscopic images can yield better results for segmentation than those obtained by the ap-

proaches using only intensity information. The most frequently used color spaces for color segmentation include HSI, HCI, HSV, OHTA etc. However, they are normally more difficult to be processed and analyzed than gray-level images. Some research work has utilized color information for abnormality detection like bleeding, inflammation etc, which are main features to convey the appearance of abnormality in endoscopic images. Yang developed a fuzzy logic based segmentation scheme in HSI (Hue, Saturation and Intensity) color space [26]. Xue developed two stages segmentation scheme in OHTA color space using a Markov Random Field (MRF) model based on multi-channel Bayesian segmentation technique. Tjoa considered an approach to segment endoscopic images based on homogeneity and color feature hue [27].

Segmented images by color information can be analyzed with texture information to obtain additional clinically useful parameters. Texture features characterize the statistical or structural relationship between pixels, and provide measures of properties such as contrast, smoothness, coarseness, structural complexity, etc. Affected regions, such as those that are bleeding or cancerous, may have different texture features. Karkanis tried three well-known textural descriptors as well as one based on wavelets to interpret tissue regions in endoscopic images [28]. A new metric feature, log-likelihood-ratio, was developed based on local binary pattern (LBP) combined with average intensity to describe the spatial structure of the local texture of an endoscopic image [29]. Marta *et Al.* has developed an approach of obtaining quantitative parameters from the texture spectra both in chromatic and achromatic domains of the endoscopic image [30].

In [31], the issue of poor visual feedback in video-endoscopy is tackled, which normally prolongs the operation time, increases the risk to the patient, and drives up

the cost of health care. [31] designs and validates novel image analysis and rectification algorithms on real images to enhance the visual feedback to the surgeon in video-endoscopy. A method was presented for estimating the amount of rotation necessary to rectify images forming a video stream to help alleviate the dis-orientation problem in endoscopy, as shown in Figure 1.6. An efficient method of computing 2-D

Figure 1.6: Sample results of image rectification

feature tracking was presented. By using a set of epipolar constraints, they obtain the fundamental matrix, which is then factored into a particular form. One of the factors is the rotation matrix that determines the camera’s orientation. From this, they update the ‘up’ direction and perform the rectification. By expressing the 2-D tracking objective function as a convolution, they were able to obtain a large speedup over straightforward evaluation.

Digital image pre-processing techniques can be used to improve the visual clarity

of an image without generating any new information about it prior to diagnosis. It includes various techniques such as edge enhancement, color enhancement, filtering, noise reduction etc. Sometimes image processing stops at the pre-processing stage. The enhanced images are displayed for further action by the surgeon. [32] discusses development of real-time endoscopic image pre-processing technology making use of adaptive index of hemoglobin color enhancement processing. Figure 1.7 shows the original and color enhanced images using a similar hemoglobin index based approach.

Figure 1.7: Example of image enhancement in endoscopy using hemoglobin index

Following image pre-processing, image segmentation is used to segment an endoscopic image to extract specific regions of interest. In [33], a parallel system for medical analysis is proposed that does image segmentation based on color and texture. A sample of processed endoscopic images is shown in Figure 1.8, which also includes lumen region segmentation, color-based segmentation and texture-based segmentation.

1.3 Image processing *during* endoscopy

So far, most of the image processing techniques that were discussed were used for the purpose of diagnosis. In such cases, the actual endoscopic procedure is followed by an off-line processing and analysis of the captured video at a later stage before

Figure 1.8: Example of image segmentation on an endoscopic image

deciding on the outcome. Most of the image processing methods employed today, as pointed in Section 1.2 are applied after the actual endoscopic procedure and hence can afford to be computationally intensive. Usually in such scenarios, the processing requirements are not real time in nature. Examples of this are [34] [35] [31] etc. As of today, a lot of image processing on endoscopic images is performed on workstations and personal computers with high processing power.

However, endoscopy can also be used for treatment purposes. In this case, not only does the surgeon require to identify infected tissues or regions calling for medical attention, but also to take spontaneous decisions and necessary steps for treating them while the endoscopic procedure is in progress. For example, [36] discusses treating gastro-intestinal bleeding. Such cases require the surgeons to perform very efficiently and need systems to aid them in real time. This is because the surgeons are under the pressure to finish the procedure as fast as possible in order not to inconvenience the patient.

If image processing can be achieved in real time, it opens up a new set of applications where it can be applied to. Combining the knowledge of image processing techniques on endoscopic images with real time requirements, there is potential of further enhancing the benefits of image processing technology for endoscopy. The

section below explains possible scenarios where real time image processing would prove beneficial.

1.3.1 Revisiting tissues of interest

During medical endoscopy, especially for treatment procedures, it is a common requirement for surgeons to revisit the area of interest in the tract and a possible application of real time processing of endoscopic images is to guide the surgeon in this re-traversal. This could be for locating infections, ulcers, growths, constrictions, polyps etc. or any other patch of tissue under observation or treatment. [37] gives a possible reason for a doctor to revisit the exact area of ‘biopsy’, at a later stage. In this paper, Shiba *et Al.* conducted a study, where 297 patients who underwent an endoscopic procedure called endoscopic mucosal resection (EMR), enrolled and 57 of them were confirmed to have bleeding after the procedure. In cases of persistent bleeding, the surgeon has to go through a field of blood to find the exact spot of biopsy and then take measures to control it.

Other common reasons for revisiting areas of interest are disturbances during the procedure, like a spurge of blood or some secretion that might disrupt the process. Then the doctor needs to get the endoscope out, clean it and reinsert it into the patient’s body to relocate the area he was working on. Also, follow-up endoscopic sessions are often performed for monitoring a patient’s recovery from tumors and growths. This again necessitates revisiting the infected tissue that is under treatment. In [39], follow-up endoscopic scans are performed to monitor the patient’s recovery from colonic lipomas, benign adipose tumors that occasionally cause symptoms.

The need to revisit specific points in the tract also arises for study and evalua-

tion of specific kinds of treatment procedures. In [40], intestinal mucosal lesions in Crohns disease were endoscopically evaluated before and after giving infliximab, and the usefulness of this treatment was investigated. [41] compares the area of a tumor before and after administering radiofrequency ablation therapy. A minimum of 6 months of clinical follow-up were reviewed. Treatment outcomes with respect to local tumor recurrence between two groups of test cases were then compared and alternate ablation techniques were recommended.

Changed working conditions during revisit

There is no guarantee that the working conditions within the tract would remain consistent during endoscopic traversals at two different time instances, making it all the more challenging for the surgeon to relocate the exact area under observation. For example, [37] shows the case of bleeding during and after the process of an EMR, an endoscopic procedure. As pointed earlier, the surgeon has to go through a field of blood to locate the exact spot of biopsy to take suitable control measures, if the bleeding is excessive and persistent. The passage may now look very different due to the bleeding and the surgeon has to use his discretion in order to decide on the location of the biopsy spot.

It is also possible that the point of interest itself undergoes a drastic change between the two traversals of the endoscope making it difficult for the surgeon to discern the area of interest. For example, the effect of medication on a tumor or growth may cause it to shrink in size or even vanish altogether [41]. On the other hand, the spread of an infection might cause the area to look very different making it a challenging task for the surgeon to spot the area that he was previously working on. Other secretions or contractions of the tract might also result in similar change of working environment that the surgeon has to cope with.

In all the cases explained above, the revisiting of tissues is done live and the surgeon needs to be efficient to relocate the point of interest unlike the conventional cases where an endoscopic pass is followed by off-line analysis for diagnosis, or unlike cases of virtual endoscopy where analysis is done on the virtually reconstructed graphics image, which again poses no such time constraints. Image processing is a possible approach to aid the surgeon in the re-traversal process in real time.

1.3.2 False negatives in diagnosis

Another possible scenario that is thought to benefit from this real time image processing is to keep track of camera positions already covered in order to ensure a thorough diagnosis. If the scanning has not been thorough, it is easy to miss out tumors and other points of interest. [42] evaluates the diagnostic accuracy of gastroscopy that detect gastric tumours and reports that many cancer cases have been missed during the first round of diagnostic gastroscopy. Patient files were examined to identify those who underwent gastroscopy less than 3.5 years before a cancer diagnosis. Of the 284 gastric cancer patients, it was found that 13 (4.6%) had undergone gastroscopy earlier and the procedure had missed the diagnosis at that time. Failure to accurately scan all parts of the organ could also mean a delay in the overall diagnosis procedure. [42] reports that the median delay in cancer diagnosis was 11.5 months. Therefore, it is also a challenge for surgeons to remember areas within the tract that have already been visited during endoscopy.

In this case, if the live diagnosis is thorough, it is possible to avoid the ‘misses’. In other words, the surgeon would need to decide in real time whether all tissues have been thoroughly scanned or not before taking a final decision.

1.4 Problem statement: Real-time camera motion tracking

It was shown in Section 1.3 that there is a need for surgeons to revisit points of interest, either for follow-up treatment or for study and evaluation of treatment methods. Furthermore, the working conditions are dynamic, making it all the more challenging for the surgeon to relocate the area he was previously working on. As explained previously, there could be blood discharges and other growths in the tissue that would make the actual region of interest look very different and difficult to discern. Doctors many a time rely on their memory while performing the endoscopic procedure. In [43], Heibert says “Endoscopy is all but a memory” and questions the worthwhileness of such follow-up sessions. When the surgeon relies completely on his memory of previous visits in order to revisit the point of interest, the reliability and efficiency of the approach are at stake.

Hence it becomes very useful to have a recording of the position of the camera (in terms of depth traversed and orientation of camera) at the region of interest during the previous endoscopic session. This information could be put to use in real time during the subsequent traversals to guide the surgeon to the same point more easily. Being able to track the motion of the camera as it traverses through the human cavity would throw light on the final camera position. Recording these camera position parameters could also prove useful for ensuring there are no ‘miss’es in diagnosis, as described in Section 1.3.2. In other words, if a computerized process keeps note of all areas that have already been visited, by recording camera position parameters, then it makes it easy for the surgeon to concentrate on those that have been missed, for ensuring a thorough scan.

For the cases pointed above, the motion guidance system has to function in real time to give feedback to the surgeon for helping him locate the point of interest. Information about the previous camera position are to be interpreted meaningfully and the doctor could be provided with directions for getting to the point as quickly as possible.

In this thesis, we explore image processing based approaches to track camera motion through the human tract and guide the surgeon in real time. Image processing offers benefits over other mechanical methods like using control cables and knobs, which are used for maneuvering the endoscope (Section 1.1.1). Most importantly, through image processing, the whole motion estimation process can be automated, without the need for manually translating the movement of the control knobs to the movement of the endoscope tube. With the knowledge of the already existing image processing approaches used in the context of medical endoscopy (Section 1.2), a solution for tracking camera motion in real time is attempted in this research work.

1.5 Summary

In this chapter, the various advances in the field of medical endoscopy were discussed, starting from fiber optics and video endoscopy to capsule endoscopy and virtual endoscopy. Image processing on endoscopic images was specifically studied in detail, exploring all the existing approaches. Popular image processing approaches like feature extraction, lumen detection and image enhancement were explored.

The chapter further goes on to identify potential benefits of processing captured images in real time. Tracking and recording camera motion has potential benefits in aiding a surgeon at a later time to revisit a point of interest. Information on

camera motion also potentially helps in ensuring a thorough diagnosis by guiding the surgeon towards areas not visited. To summarize, this chapter explains the need for an image processing based approach to infer the motion of the camera during an endoscopic procedure in real time.

The following chapter delves into tracking of camera motion through efficient processing of endoscopic images. After first studying the existing methods and identifying their limitations for real time usage, it draws an outline of the envisaged solution. It also identifies the various computational blocks to be tackled for arriving at the solution.

Chapter 2

Tracking camera motion

The previous chapter explains the usefulness of image processing for tracking camera motion during endoscopy to guide the surgeon for subsequent traversals. In this chapter, the existing methods for camera motion tracking are first surveyed and their limitations for generating real time solutions are identified. The outline of the envisaged approach is presented after laying down the scope of the problem. Finally, the broad computational blocks are identified, each of which will be tackled in the chapters to follow.

2.1 Existing methods to track camera motion

There have been attempts in the past to track the motion of a camera using image processing. In [44], Thorsten et al. present a robust estimation technique to compute camera motion for 3D reconstruction of a scene from an endoscopic image sequence. Based on conventional feature detection and correspondence analysis, the robustness of the approach is achieved by the use of an efficient outlier detection of misaligned correspondences. This kind of robustness is required for the application of 3D reconstruction. However, it is evident that in the attempt to make it robust, this approach is compute-intensive and not very suitable for real time applications.

In [44], 930 feature points are detected in the first image and 898 in the second.

In another research work [45], Mori et al. attempt to track motion of a bronchoscope using epipolar geometry analysis and intensity-based image registration of real and virtual endoscopic images. The complete method consists of two steps: (a) rough estimation using epipolar geometry analysis and (b) precise estimation using intensity-based image registration. In the rough registration process, the method computes camera motion from optical flow patterns between two consecutive real video image frames using epipolar geometry analysis. In the image registration stage, they search for the virtual endoscopic system viewing parameters that generate the virtual image that is most similar to the current real image. The method required about 20 seconds to process one frame because of the intensive computations and is hence not very suitable for a direct porting into real time systems.

It is evident that the existing methods are compute intensive and are not feasible for deployment in real time solutions. In this thesis it is aimed to estimate the motion of the camera by processing in real time, a stream of images captured during endoscopy. By doing so, one can infer the final position of the camera at the desired feature point. This aids in relocating areas that were previously being worked on, which helps in a thorough scan during diagnosis, as explained in Chapter 1, Section 1.4. It is attempted in this research work to come up with simple yet elegant methods to provide a fast means of guidance to the surgeon during endoscopy.

2.2 Problem scope definition

Cylindrical tracts are common in the context of endoscopy, e.g. esophagus, bronchus etc. The scope of this work is restricted to track the motion of the camera through

cylindrical tracts, where the camera faces the axis of the cylinder. For initial proof of concept, experiments are carried out on geometrical cylinders made of paper imitating the human tract and the methods are later demonstrated on real images. Since the motion of the camera is restricted as it is moving through a tunnel-like passage, certain degrees of motion are more likely than others because of the restrictions imposed by the walls of the tract. In order to simplify the problem at this stage, the various degrees of motion should be studied carefully and only the most important ones selected.

This research work attempts to track the *linear* motion of the camera within the cylindrical tract as well as the *rotation* about the main axis. Let us first analyze the various ways in which the camera can be maneuvered in the cylindrical tract. Figure 2.1 shows the possible degrees of motion. There can be either a rotation or a translation along the X, Y and the Z axes, totaling to six possibilities. T1 gives the depth of traversal of the camera into the tract and R1 gives the orientation of the view of the tract. The sideward translations correspond to T2 and T3 in 2.1. Finally, R2 and R3 provide the tilt for the camera to face a particular side wall of the tract in closer proximity.

Although Figure. 2.1 shows the motion for all six degrees of movement, the camera motion is restricted as it is moving through a tunnel-like passage. The walls of the cylinder impose restrictions on sideward movements (R2, T2, R3 and T3). But a free movement is allowed along axis 1 (T1) and rotation about it (R1). In other words, T2 and T3 are limited by the diameter of the cylindrical tract whereas T1 can be as much as the length of the tract. Similarly R2 and R3 are restricted by the walls, but R1 can take values up to 360° or more. Because of this, a large component of the overall camera motion will be contributed by T1 and R1. This has also been veri-

Figure 2.1: Degrees of motion of camera through a cylindrical tract

fied through subjective observations of a wide range of endoscopy video footage [61] .

In this thesis, a detailed mathematical analysis is provided mainly for tracking R1 and T1 (or a combination of R1 and T1), as these are the most probable movements that the camera can produce, considering the constraints imposed by the walls of the cylindrical tract for other movements. It is assumed that the motion of the camera is smooth in between image frames considered. There are no jerky movements and every two image frames have a reasonable link in terms of image content.

2.3 Outline of envisaged approach

An image sequence consists of a stack of image frames that have been captured in consecutive time instances. When two images are captured with a very small time interval, the majority of the contents of the scene would remain the same. However, portions of the scene may show different kinds of transformations like shrinking, expanding, skew, rotation etc. The nature and degree of these transformations could potentially throw light on the camera motion during that elapsed time.

Instead of studying every pixel of the scene under consideration, we can selectively identify certain distinct portions of it and study their corresponding transformations to extrapolate to the rest of the scene. Let us call the anchor points as Feature Points (or FP for short). These could refer to tumors, polyps, lesions etc. which show distinct properties in the image and are easy to be isolated and studied. By analyzing the properties of certain selected FPs, one can make crude estimations about the camera movement. In order to facilitate a proof of concept, it is assumed that there are adequate feature points on the walls of the tract and that there are no glaring patches that could lead to false FPs.

During medical endoscopy, cylindrical passages are usually photographed from the camera moving through the passage itself. It is aimed to estimate the motion of the camera during endoscopy by studying and processing the image sequence obtained (as a video) during the first endoscopic traversal through the tract. A typical snapshot of an image sequence captured during endoscopy looks like Figure 2.2.

As pointed out in Section 2.2, rotation and linear motion of the camera (and a combination of these) are aimed to be estimated by this research work. If it is known that there has been just a pure rotation of camera, then in order to infer the angle

Figure 2.2: Snapshot of image sequence during endoscopy

of rotation, a possible approach is to rotate the first image exhaustively by varying angles and performing a match of rotated image against the second one. The angle corresponding to the highest match value would be the possible relative angle between the two images. However, to achieve this we need to have a high speed image rotator to exhaustively rotate the image by varying angles.

In order to detect linear camera motion, the *coordinate positions* of feature points are studied. The walls of the cylinder are projected as circular rings on the image (Figure 2.2) and this gives a template for analyzing the feature points. With regards to the coordinate position of the feature points, the following observation can be made. If the camera moves downward in a linear fashion, then the feature points move radially outwards. That is, they are projected on rings of bigger radii. Similarly, when the camera moves upward linearly, the positions of feature points shift toward the center of the rings. This suggests working in a transformed domain in order to better study these projections of the cylinder as circular rings in the image.

Figure 2.3 shows an image of a test cylinder captured with a camera moving through it, with feature points marked as black patterns. For this, the laboratory setup that will be explained in Section 2.5 was used. After capturing Figure 2.3 (a), the cam-

era is moved upward before capturing Figure 2.3 (b). This is reflected by the two marked feature points (arrowhead and tip of cross) that have moved inward toward the center of the cylinder.

Figure 2.3: Movement of feature points when camera moved upward

It is aimed to mathematically map this displacement of feature points to the linear camera movement along the axis of the cylinder. Given two images $image_1$ and $image_2$, captured with a time interval t , the method attempts to track the linear motion of the camera that has taken place during t . Feature points are first extracted from $image_1$ and their corresponding new positions are to be located in $image_2$. It is envisaged that a thorough study of positions of feature points and their displacements would throw light on the downward/upward motion of the camera. The displacements of these FPs are studied in a transformed domain to understand more clearly the projections of the cylinder walls as concentric rings on the image.

In the following section, the computational blocks employed to tackle this problem are presented.

Figure 2.4: Outline of proposed research steps

2.4 Identifying computational blocks

A block diagram of the broad steps to be taken are summarized below in Figure 2.4. As explained in Section 2.3, rotation and linear motions of the camera as well as a combination of these are aimed to be estimated. Attempts to detect pure camera rotation through exhaustive image rotations necessitates a high speed rotation engine. This will be elaborated in Chapter 3. Novel schemes are proposed for performing high throughput rotation of images, especially of high resolution. After each rotation, an image correlation operation can be used to obtain a measure of match between the rotated first image and the second image. If this value is high, then it is likely that this is the angle by which the camera was rotated.

It was illustrated that the change in properties of feature points present in consecutive image frames of a video sequence can help make crude estimations about the linear camera movement. In order to quantify the movement of these feature points, the following broad computations are seen as necessary. Firstly, there is a need to extract the FPs and locate them in the next image. Image processing techniques are employed for feature point extraction and tracking. Image pre-processing

enhances feature points, thereby making it easier for their extraction. It also helps to reduce the data being processed, thereby helping in achieving compute efficiency. After noting the initial and final coordinate positions, the displacements of FPs are to be analysed. For analysis of circular projections of the cylinder, a transformed domain is employed. For this, there is a need to perform an unrolling of the image. This standardizes the recordings of different FPs by eliminating differences that arise due to varying depths and positions of the FPs, as will be explained later. Each of these steps are detailed in Chapter 4. It will be shown here that although FP study was targeted for tracking only linear motion of the camera, the same method can be used to detecting rotations also.

For the proposed method for tracking linear camera motion, any of the existing feature point extraction and tracking methods can be used. The method focuses on analysis of FP position displacements in a transformed domain through a process of image unrolling in a compute efficient manner. Selecting feature points in medical images has been a separate research study altogether because of the numerous challenges accompanying it [59] [35]. For example, the surface of the human passage could be lined with mucus and other shiny tissues which could cause glaring effects and misleading feature points. Also in cases of smooth walls, feature points are more subtle and all the more difficult to locate. In this research work, novel feature point extraction methods are not proposed. Instead, efficient means to study and analyze the properties of feature points are devised.

2.5 Experimental Setup

For initial proof of concept and verification at each step in the project, images captured in a simulated environment are used. Subsequently, results are demonstrated

Figure 2.5: Experimental setup

on real endoscopic images.

In order to simulate the movement of the endoscope through the human tract, a laboratory set up as shown in Figure 2.5 was used. A sheet of paper with feature points marked is rolled into a cylinder and placed below the camera. The camera is mounted on a stand such that it can be lowered into the paper cylinder in a controlled manner. A light source is attached to the camera such that it moves in one entity with it, similar to the medical endoscopic setup. Images are captured after simulating various rotation and linear motions of the camera, for further processing. An example of an image captured using this setup is shown in Figure 2.6. Markings are deliberately introduced on a sheet of paper to simulate feature points (Figure 2.6 (a)). It is then rolled into a cylinder and its captured image is shown in Figure 2.6 (b).

Figure 2.6: Example of laboratory simulated image

2.6 Summary

In this chapter, methods to track camera motion during endoscopy have been discussed. First the existing methods were briefly described followed by their limitations for deployment in real time systems. Then the scope of the problem tackled in this research work was defined, according to which only cylindrical tracts are considered. Out of the several possible degrees of motion, rotation about the main cylindrical axis and linear motion along it will be tackled. The broad computational blocks like image rotation, feature point extraction, feature point tracking, image unrolling and analysis were laid down.

In the next chapter, means to detect pure rotations of the camera will be explored as the first step towards camera motion tracking (Figure 2.4). The use of a high speed image rotation engine to achieve this will be explained. Further, the existing real time approaches will be explored by studying image rotation hardware. Novel techniques to achieve high throughput rotation of images will be presented. A detailed comparative study will show the benefits of the proposed method in comparison with existing ones.

Chapter 3

High speed image rotation

It was briefly discussed in Chapter 2 that a possible way to detect pure camera rotations between two images is by exhaustively rotating the first image and correlating against the second (Figure 3.1). In this chapter, this approach is first explained and verified. Next, the use of a high speed image rotation engine for detection of camera rotation is elaborated. Since specialized hardware offer faster performance as compared to its software counterparts, the various hardware approaches that exist for performing rotation of images are studied. The limitations of these methods in achieving a high performance rotation, especially for high resolution images, are then understood. Novel strategies for high throughput image rotation are proposed. Area time benefits, accuracy and scalability of the proposed method are evaluated against existing methods.

3.1 Exhaustive rotation approach

A flowchart in Figure 3.2 explains this method of angle detection. The inputs to this system are two images, let us call them $image_1$ and $image_2$. Let us assume that the camera has been rotated by a small angle θ after capturing $image_1$ and before capturing $image_2$. The first step in this process is to rotate $image_1$ by a

Figure 3.1: Detection of camera rotation

random angle α to get rot_image_1 . Then, an image correlation operation between rot_image_1 and $image_2$ is performed to quantize the degree of match between them. Let us call this correlation value as cor . This correlation value is stored and the entire process is repeated for a new value of angle α . The maximum of these correlations is computed and the angle giving this maximum correlation is also noted. The negative of this angle that gives the maximum value of cor is taken as the angular displacement between $image_1$ and $image_2$. The negation is performed because if the camera is rotated in the clockwise direction, then $image_2$ would be in an anti-clockwise orientation compared to $image_1$ and vice versa. This is better illustrated in the example in Section 3.1.1.

3.1.1 Proof of concept

The entire method was functionally verified using MATLAB using the images captured from the experimental setup, as explained in Chapter 2, Section 2.5. For the correlation step, input images need to be of the same size. But due to rotation, rot_image_1 is usually larger with many padded white pixels (as seen in $image_2$ of Figure 3.2). In order to circumvent this problem, rot_image_1 is cropped to the same size as $image_1$ and the correlation is performed only on the central portion of the two images.

Figure 3.2: Steps in 'exhaustive rotation approach'

Figure 3.3: Example of rotation detection

An example is shown in Figure 3.3. $image_1$ and $image_2$ are snapshots of a rolled paper cylinder. After capturing $image_1$, the camera has been rotated by about -30°

Figure 3.4: Correlation versus angle of rotation

to capture $image_2$. In order to detect this angle through exhaustive rotation and correlation, $image_1$ was rotated by 15° , 30° , 45° , 60° , 75° and 90° . The rectangular area marks out the regions that are being correlated. Figure 3.4 shows a plot of the correlation value against each of these rotations. It can be seen that 30° rotation produces the maximum correlation value of 0.95. It is to be noted that for the sake of this experiment, only pure camera rotations were simulated (without any other kind of movement), which justifies the high correlation output that is obtained.

3.1.2 Need for high speed rotation engine

To make the method truly exhaustive, it is desirable to rotate by more number of angles before concluding the angle based on the highest correlation value. In other words, the resolution of angles considered for rotation should be increased. However, for this to be feasible in real time, the rotation and image correlation should be done efficiently. This necessitates a high speed rotation engine, which would be specially beneficial for processing endoscopic images of high resolution, an increasing trend in the medical field. In this chapter, methods for high speed image rotation engine are explored. The next section explores the various options that are currently avail-

able for performing image rotation. Hardware architecture for rotation of images are studied closely as they offer benefits for high performance.

Image correlation is performed by directly using the MATLAB functions in order to verify the results. There is scope for future work in speeding up the image correlation process as well, which is not tackled here. The rest of the chapter focuses on image rotation techniques and architectures.

3.2 Existing hardware for image rotation

Conventional rotation of images involves performing trigonometric computations on each image pixel. Previously reported hardware implementations for image rotation rely on LUT (Look-Up Table) or CORDIC (Coordinate Rotation Digital Computer) based methods to simplify the complex trigonometric operations. CORDIC [46] is capable of performing a series of micro-rotations on a vector lying on the X-Y plane over a desired input angle using simple add-shift operations. However, the iterative nature of CORDIC can lead to performance degradation, especially for images of high resolution. An alternative technique to overcome the inefficiency of the trigonometric computations is by employing a LUT to store all the possible sine and cosine values. However, since the size of the LUT is governed by the resolution of angles that can be rotated, the hardware cost incurred becomes undesirable for applications that require high rotation resolutions.

Till date, there has been very limited reported work on hardware implementations for image rotation. Two notable contributions are found in [47] and [48], which employ LUT-based and CORDIC-based methods respectively.

3.2.1 LUT-Based Implementation

Bhandarkar and Yu [48] proposed an image rotation engine that is based on mapping of pixels along a skew raster scan line in the source image to a horizontal scan line in the rotated image space. The skew lines are parallel to each other in the source image and inclined at an angle $-\theta$ to the horizontal where θ is the specified rotation angle. The design makes use of an LUT to store the sine and cosine values, which are required for computation of the initial rotated position. Due to the backward mapping of the pixels onto the rotated image, the holes or measles that are normally introduced in the rotated image have been eliminated.

However, the throughput is still limited since there is no parallelism exploited in this architecture. Although this technique has scope for parallelism, it would incur a computational overhead and a significant increase in hardware cost. It has been shown how the bulk of the operations can be handled by just additions. However, it becomes inevitable to perform at least one set of multiplication operations to obtain the address of the first pixel in the first scan line. This is because only sine and cosine values are stored and computations resembling equations 3.2 are mandatory to obtain the starting address of the first skew line, before generating the remaining addresses by simple additions. This overhead becomes more prominent when the architecture is parallelized. Moreover, it does not scale well for applications requiring high rotation resolutions. For example, if the resolution of the rotation is made finer (e.g. 0.1° instead of 1°), the memory required to store the sine/cosine values will increase by ten times.

3.2.2 CORDIC-Based Implementation

Pixel-by-Pixel CORDIC-based Rotation

The CORDIC engine can be employed to improve the hardware efficiency of image rotation systems that are based on computing transformations on each image pixel [49]. Let (x', y') be the coordinates of a pixel, which has been rotated by an angle of ϕ where (x_0, y_0) is the original position of the pixel. Equations 3.1 show the relationship between (x', y') and (x_0, y_0) , where $\delta = +1/-1$ when ϕ is $+ve/-ve$.

$$\begin{aligned} x' &= \cos\phi(x_0 - \delta y_0 \tan\phi) \\ y' &= \cos\phi(y_0 + \delta x_0 \tan\phi) \end{aligned} \quad (3.1)$$

The CORDIC algorithm, which was developed by Volder [46], can be used to reduce the complexity of the trigonometric computations in Equation 3.1. The CORDIC computation performs a series of micro-rotations on a vector lying on the X-Y plane over a desired input angle using simple add-shift operations. The algorithm is based on the principle that any angle can be approximated as a summation of n micro angles of the form $\arctan(2^{-1})$ (i.e. $\phi \approx \sum_{i=1}^n \pm \arctan(2^{-1})$). The equations for the CORDIC micro-rotations shown in Equation 3.2 can be derived from Equation 3.1 by replacing the multiplication with simple shift operations by i positions.

$$\begin{aligned} x_{i+1} &= k(x_i - \delta_i y_i 2^{-i}) \\ y_{i+1} &= k(y_i + \delta_i x_i 2^{-i}) \end{aligned} \quad (3.2)$$

Image rotation architectures that rely on the pixel-by-pixel transformations are intensive in CORDIC computations. In the recent past, there have been many attempts

to improve the performance of the CORDIC engine, which include use of redundant number systems [50] [51] and architectural improvements such as pipelining and parallelism [52]. Although these advancements have shown tremendous improvement in the CORDIC algorithm, the number of CORDIC operations still remains as the bottleneck that restrict the throughput of the architecture. This is exacerbated in systems that process high resolution images, as the number of CORDIC computations increases proportionally with the number of image pixels.

Pipelined-Parallel Architecture

Ghosh and Majumdar [53] [47] proposed schemes to reduce the number of CORDIC computations in an image rotation system. In [53], a pipelined-parallel architecture for image rotation using CORDIC was proposed where a 512×512 image is first divided into an 8×8 window grid. The rotated positions of the centers of the 64 window grids are first calculated using the CORDIC engine and stored in the initialisation stage as shown in Figure 3.5. In Pipeline Stage A, the relative rotated offset of each pixel with respect to its window center is computed using the CORDIC engine. Only the pixels of a single window are considered for the offset computation, as the relative offsets of the corresponding pixels in each window are the same. In Pipeline Stage B, this offset is added in parallel to the 64 rotated window centers using local adders to generate the rotated positions of the 64 corresponding pixels simultaneously. The CORDIC engine employed for a 512×512 image in [53] performs 12 CORDIC iterations with 20-bit data path width.

The bottleneck of this approach arises from the iterative CORDIC micro-rotations. Firstly, the approach has high initialisation latency, as 64 CORDIC computations are required to obtain the rotated window centers. In addition, the CORDIC computations are required in Pipeline Stage A for each pixel in a single window. This

incurs an additional $64 \times 64 = 4096$ CORDIC calculations, which contributes to high overall latency.

Figure 3.5: Illustration of Rotation Method in [53]

3.3 Proposed Image Rotation Technique

The image rotation scheme proposed in this thesis addresses the computational bottleneck of the approach in [53], without the need for large LUT. The following schemes have been proposed to increase the throughput of the rotation process:

1. Employing a hierarchical approach to generate the rotated centers in order to significantly reduce the initialisation latency
2. Exploiting the symmetrical characteristics of the image coordinate system to

further reduce the number of CORDIC computations for calculating the rotated centers

3. Replacing the CORDIC computations in the offset calculation with additions by exploiting the order of the pixel coordinates.

3.3.1 Hierarchical Approach for Generating Rotated centers

Figure 3.6: Hierarchical rotation of centers

This method aims to reduce the number of CORDIC computations in the initialisation stage by recursively partitioning the image into hierarchical quadrant layers. To explain this method, we take the example of an 8×8 window grid and its 3 hierarchical quadrant layers as shown in Figure 3.6. A window center lies in a single quadrant in each of the 3 quadrant layers. In the example shown, center P has A, B, C as its set of identifying quadrants in Layers 1, 2 and 3 respectively. The rotated position of any center P (with respect to the image center) is equivalent to the sum of the rotated positions of the centers of the identifying quadrants with respect to the Layer centers. Thus, the number of CORDIC operations to obtain the 8×8 window centers can be reduced by computing only the rotated positions of the 4 centers of each layer, and subsequently generating the rotated positions of all the centers through various addition combinations of the 12 centers. This has effectively reduced the number of

CORDIC computations from 64 to 12.

As described in [53], having a larger dimension of window grid can lead to increased performance, as there will be lesser number of offset computations in Pipeline Stage A (see Figure 3.5). This implies that the proposed method can employ larger number of hierarchical layers to improve the performance. However, increasing the number of hierarchical layers necessitates a larger number of additions to obtain all the rotated window centers in the initialisation stage. Hence, the choice of the number of hierarchical layers affects the overall performance. This will be discussed in more detail in Section 3.6.4.

3.3.2 Exploiting Symmetrical Characteristics to Reduce CORDIC Computations

We can further reduce the number of CORDIC computations in the initialisation stage by exploiting the fact that the coordinates of the window centers in the hierarchical layers are symmetrical about the coordinate axes. Hence, in hierarchical layers with the same number of horizontal and vertical pixels, we only need to perform the CORDIC computations on a single representative center of the layer. The remaining 3 image centers can be inferred by exploiting the symmetrical characteristics of the coordinate system. In the case of hierarchical layers with different number of horizontal and vertical pixels (rectangular images), 2 CORDIC computations are required. It is noteworthy that the inference of the remaining window centers requires at most a simple negation. Figure 3.7 illustrates this notion of inferring the rotated window centers based on the symmetrical characteristics of the image. For example, by computing (x', y') , the rotated position of (x, y) , we can easily infer the rotated positions of $(x, -y)$, $(-x, -y)$ and $(-x, y)$. This has led to further reduction in the number of CORDIC computations from 12 to 3.

Figure 3.7: Inferring rotated pixel value based on image symmetry

3.3.3 Generation of the Pixel Offsets

In this section, we propose a hardware efficient technique to reduce the computational complexity of Pipeline Stage A (see Figure 3.5). In [53], the offsets of each pixel in the 64×64 window are computed using the iterative CORDIC computations. However, if the pixel coordinates are provided in a consecutive order for the offset computations, it is sufficient to calculate the offset of just one pixel, and increment the result with a regular value (of $\sin\phi$ or $\cos\phi$) to obtain the offsets of the neighbouring pixels. The explanation to this can be obtained by analysing Equation 3.2 for input values (x, y) , $(x+1, y)$ and $(x, y+1)$, since neighbouring pixels differ by exactly 1 unit. The flowchart shown in Figure 3.8 elucidates the process. This reduces the 64^2 CORDIC computations to mere additions.

3.3.4 Illustration of proposed method

In order to illustrate the proposed method for generating the rotated positions of the centers, we refer to Figure 3.9. Figure 3.9 (a) shows a 512×512 image divided

Figure 3.8: Offset computation

Figure 3.9: Layer centers with hierarchy 2

into 4×4 window grids and hierarchy is taken as 2. The centers of each of the 16 window grids is shown. Figure 3.9 (b) and (c) show the quadrant layers, with the

Table 3.1: Illustration of proposed rotation method

Angle = 30, Hierarchy = 2			
Center	Method of Generation	Rotated X Coordinate	Rotated Y Coordinate
Step 1: Generation of A'1 and A'2 using CORDIC			
A'1	using CORDIC	146.2	-106.7
A'2	using CORDIC	73.1	-53.4
Step 2: Generation of B', C' and D' by Symmetry			
B'1	$-A'1y, A'1x$	106.7	146.2
B'2	$-A'2y, A'2x$	53.4	73.1
C'1	$-A'1x, -A'1y$	-146.2	106.7
C'2	$-A'2x, -A'2y$	-73.1	53.4
D'1	$A'1y, -A'1x$	-106.7	-146.2
D'2	$A'2y, -A'2x$	-53.4	-73.1
Step 3: Generation of O'			
O'_{A1A2}	$A'1 + A'2$	219.3	-160.1
O'_{A1B2}	$A'1 + B'2$	199.6	-33.6
O'_{A1C2}	$A'1 + C'2$	73.1	-53.4
O'_{A1D2}	$A'1 + D'2$	92.9	-179.8
O'_{B1A2}	$B'1 + A'2$	179.8	92.9
O'_{B1B2}	$B'1 + B'2$	160.1	219.3
O'_{B1C2}	$B'1 + C'2$	33.6	199.6
O'_{B1D2}	$B'1 + D'2$	53.4	73.1
O'_{C1A2}	$C'1 + A'2$	-73.1	53.4
O'_{C1B2}	$C'1 + B'2$	-92.9	179.8
O'_{C1C2}	$C'1 + C'2$	-219.3	160.1
O'_{C1D2}	$C'1 + D'2$	-199.6	33.6
O'_{D1A2}	$D'1 + A'2$	-33.6	-199.6
O'_{D1B2}	$D'1 + B'2$	-33.6	-199.6
O'_{D1C2}	$D'1 + C'2$	-179.8	-92.9
O'_{D1D2}	$D'1 + D'2$	-160.1	-219.3

corresponding representative centers denoted by A1 and A2.

Table 3.1 shows each of the steps in generating the rotated positions of all the sixteen window centers, for an angle of 30° . The first step is to compute the rotated positions of A1 and A2, the representative centers of the two layers. This is computed using the CORDIC engine and the rotated positions are denoted by A'1 and A'2 respectively. In the second step, exploiting the symmetry characteristics as explained in Section 3.3.2, we compute the rotations of the remaining layer centers,

$B'1$, $B'2$, $C'1$, $C'2$, $D'1$ and $D'2$. The third step shows the generation of the rotated positions of the 16 window grids using the results obtained in steps 1 and 2. This is shown in Table 3.1. The centers are labeled based on the layer centers that are added to generate them. For example, O'_{A1D2} is generated by addition of $A'1$ and $D'2$, and so on. This illustration shows that the number of CORDIC operations for 2 hierarchical layers is reduced from 16 to 2.

Once the rotated positions of these centers are obtained, they are added with the pixel offsets generated using Figure 3.8 are added in parallel to corresponding points using the local adder array to obtain the final rotated pixel-coordinates.

3.4 Rotation system overview

The overall rotation process can be broadly divided into the generation of rotated centers and generation of offsets, running on similar lines as [53]. For illustration purpose, we fix the number of hierarchical layers as 3, and we assume that the hierarchical layers have equal number of horizontal and vertical pixels. Figure 3.10 depicts the main computations involved. The rotated position of a single window center from each hierarchical layer is first computed using CORDIC. Subsequently, the rotated values of the remaining window centers in the hierarchical layers are inferred through the 'Symmetry-based Inference Unit'. As discussed earlier, this requires at most a simple negation operation. The outputs of this unit are added in different combinations to generate the 8×8 rotated-positions of the window grid centers. The various addition combinations are generated using a control signal and the values are routed to the adder through multiplexers (MUX). The outputs of the adder correspond to the rotated positions of all the 64 window grid centers, which are stored in the memory to be used in the next stage.

Figure 3.10: System overview of proposed image rotation engine

To generate the pixel offsets, the sine and cosine value of the angle are obtained using the CORDIC engine. Then rotation is performed on the first pixel of the window, $(1, 1)$, using the CORDIC engine to generate (x', y') . Subsequent offsets are obtained by incrementally adding the sine and cosine values to (x', y') as described in Figure 3.8. As these offsets are generated, they are simultaneously added to all centers using the local adder array to obtain the final rotated pixel-coordinates.

3.4.1 Accuracy analysis of the proposed system

The accuracy of the proposed image rotation system is governed by two factors:

1. interpolation scheme employed in the image rotation system
2. error of the rotated pixel position, which are elaborated in the sections to follow.

A certain degree of accuracy of the final rotated position is often needed in order to choose certain interpolation schemes. It is worth mentioning that the proposed system can employ various interpolation schemes based on the application's requirement by utilizing a CORDIC implementation with sufficient accuracy.

Interpolation schemes

Conceptually, an image rotation requires two steps: a coordinate system transformation followed by pixel interpolation. Interpolation is necessary when the transformed pixel positions do not coincide with the original pixel positions. New pixel values are obtained by interpolating the original pixels in the neighbourhood of the transformed pixel position. There are many popular interpolation functions, which include nearest neighbour, bilinear, bicubic polynomial, cubic spline and Gaussian [54]. The proposed system can easily accommodate any of these interpolation schemes. The

choice of interpolation scheme involves three main considerations, which are discussed below:

1. computational cost: There is a direct cost consideration when choosing among these interpolation kernels. For example the nearest neighbour requires the least number of computations as no arithmetic operations are needed. The interpolated output data point is assigned the value of the nearest sample point from the original data. As mentioned in [55], cubic interpolation requires 40% fewer multiplies than bicubic interpolation.
2. required quality of the final image: The other consideration when choosing the interpolation methods is the desired accuracy of the final image. Each method has a capacity to reconstruct the sampled image with a certain degree of exactness. It can be seen in [55] that of all the interpolation methods considered, the nearest neighbour introduces noticeable noise and jagged edges.
3. accuracy of the rotated pixel position: Another important issue that accompanies the choice of the interpolation method is the accuracy of the rotated pixel position, which has to match the size of the interpolation kernel. In other words, if the rotated coordinate is of 1-pixel accuracy, then only the nearest neighbour method would be feasible. The other two methods require sub-pixel accuracies of the coordinate position. For example, in [54], to implement the Gaussian interpolation each destination pixel is treated as an array of 65×65 subpixels.

Accuracy of rotated pixel position

In this section, we describe the quantization errors that affect the accuracy of the final rotated coordinate position in our system. As described in [56], the CORDIC algorithm inherently suffers from 2 types of quantisation errors. The first is due to

approximation of the angle arising from finite number of micro- rotations and the second is because of usage of finite length registers in the CORDIC data path.

The first type of error is inherent in the CORDIC algorithm and is dependent on the number of iterations that are chosen. However, the second type is purely a design issue since it is affected by the size of the finite-length registers. In the proposed approach, error would be introduced due to the large number of additions during the offset computations. The proposed technique provides a means for design exploration based on accuracy and area trade-offs. Simulations for register lengths varying from 12 to 32 were performed for a 512×512 image. The maximum error is defined as the average of the maximum errors along the x and y directions, $\frac{max(error_x)+max(error_y)}{2}$, where $error_x$ and $error_y$ are the maximum errors in the x and y coordinates respectively. For each data-path length, this error is recorded for all angles, and the maximum of these errors is plotted on a logarithmic scale as shown in Figure 3.11.

Figure 3.11: Rotation position accuracy versus operand length

It can be observed that the error decreases exponentially with increasing operand

length and becomes negligible for very high operand lengths. As mentioned earlier, the choice of interpolation method is application specific and the rotation schemes proposed in this thesis are independent of it. The various interpolation schemes can be adopted by altering the register length to provide for sufficient accuracy. In the remaining sections, we adopt the same interpolation scheme as [53], which is the nearest neighbour interpolation method, in order to compare between the two systems.

Choice of data width

We will show that any additional error that is induced due to fixed register sizes can be offset completely by extending the data width. To illustrate this, we choose the image size as 512×512 , the interpolation scheme as nearest neighbour and to keep the error comparable to that in [53], we replace the registers and adders of length 20 (as in [53]) with those of length 25. As mentioned earlier, the number of hierarchical layers was chosen to be 3 to keep the window sizes comparable. Figure 3.12 shows the plot of average error per pixel for the proposed method with the register length of 20 and 25. The error is obtained by comparing the results with the conventional rotation method that performs the CORDIC operations on each pixel in the 512×512 image. The outputs (rotated x and y coordinates) were compared with pixel-by-pixel rotation method for all angles from 0° to 45° with 1° resolution.

3.5 Extending to higher resolution images

In this section, a new parameter called Hierarchy (or h) is introduced that enables us to study the proposed method across a wide range of area-time measures. The parameter h refers to the number of layers that the image is broken into before the proposed techniques are applied. The extreme cases are $h = 0$ and $h = \log_2 m$ where

Figure 3.12: Higher datawidth to hide extra quantization errors

m refers to the length on the square image.

(i) $h = 0$: With no hierarchy, the initialization phase is eliminated since the image center is the only center to be rotated (with respect to itself and hence redundant) and the entire image becomes the window for offset generation. Using the proposed engine, the computations involved would be only additions with 2 CORDIC operations. One of the main limitations of this set up is that it cannot be parallelized since pixels need to be fed in successive order (Figure 3.8). In addition, larger window sizes will lead to higher quantization error.

(ii) Maximum possible h : With the maximum hierarchy of $\log_2 m$, each pixel becomes a center and this implies the entire computation is done in the initialization phase. Pipeline stages A and B (Figure 3.5) now become redundant.

The choice of this parameter exerts a push-pull effect on the initialization latency and the offset generation. For example, lower levels of hierarchy will lead to larger window sizes, and this increases the latency of the offset computation. Alternatively, higher levels of hierarchy lead to more additions of the layer centers to obtain the

rotated positions of the centers in the initialization phase. It is evident that the optimal hierarchy lies somewhere in between these two extremes.

High resolution images can accommodate higher hierarchies because the large size of the image allows breaking into many smaller layers. The results of a detailed mathematical analysis of how the area and time of the engine vary with varying hierarchies are included in the next section. Expressions for latency and area of the rotation engines in terms of h are derived that would help in making an informed choice of the hierarchy. It will be shown that with higher hierarchies (higher values of h), the proposed method starts to show more obvious benefits in terms of performance.

3.6 Image rotation engine: Comparative analysis

In this section, the proposed rotation engine is compared with the design proposed in [53] in terms of design considerations, computational complexity, latency and area of overall chip for a 512×512 image. A 12-iteration CORDIC engine is used. The data width of the adders/registers used in the CORDIC engine for the method in [53] and the proposed method are 20-bit and 25-bit respectively. The outputs of CORDIC in Pipeline Stage-A (Figure 3.5) are truncated to 10 bits prior to the additions in Pipeline Stage B. So, the data widths of the local-adders in both methods are 10-bits wide. An 8×8 window grid is employed, which gives 64 window grid centers and a window size of 64×64 . To obtain this configuration, a hierarchy of 3 was chosen for the proposed method. The latency estimates are obtained from the Passport 0.35 μ standard-cell library [57].

3.6.1 Computational complexity

The entire rotation process essentially comprises of additions and CORDIC functions and these will be analysed for the reference method and the proposed method using their system block diagrams shown in Figure 3.5 and Figure 3.10 respectively. In [53], 64 centers are rotated during the initialisation phase and this accounts to 64 CORDIC operations. In the proposed method, 3 centers are rotated and subsequently 2 additions are performed for every rotated center. Hence, a total of 3 CORDIC computations and 128 additions are performed in the initialisation stage. For the generation of the offsets, the technique in [53] performs a rotation for each of the 64×64 pixels in the offset-window, accounting to $64^2 = 4096$ CORDIC operations. In the proposed method, one CORDIC computation is performed to rotate the first pixel of the offset-window and another CORDIC computation to obtain the sine and cosine values of the input angle.

Subsequently, the remaining 4095 offsets are obtained through additions. Once the offsets and the rotated centers are generated, the final phase is similar for both the methods, where each offset is added to all the rotated centers using local-adders to generate 64 rotated pixels simultaneously. The number of different computations involved in each stage are summarised in Figure 3.13 and it can be seen that the proposed method requires only 5 CORDIC operations as against $64^2 + 64$ in the method in [53].

3.6.2 Latency comparison

In this section, we estimate the latency of the CORDIC computations for the method in [53] and the proposed method. Here, *latency* is referred to the total computational time. It is assumed that all additions are implemented using CLA adders and the latency estimates are in terms of Full Adder Latency (T_{FA}). The estimated latency for

	Method in Ghosh and Majumdar [7]		Proposed method	
Computational latency (T_{FA})	399 360		19 199	
Area (A_{FA})	3817		3951	
Computational complexity	CORDIC	Addition	CORDIC	Addition
Initialisation	64	—	3	64×2
Pipeline A	64^2	—	2	$64^2 - 1$
Pipeline B	—	64^3	—	64^3

Figure 3.13: Area, latency, computational complexity comparisons

an n -bit CLA addition is $(\log_2 n) \times T_{FA}$. Hence, a 10-bit add-operation is estimated to take $3.32T_{FA}$, 20-bit add about $4.32T_{FA}$ and a 25-bit addition takes $4.64T_{FA}$. A CORDIC engine comprises of addition and shift operations. Assuming that the shift operations are carried out through MUX-based (multiplexer-based) barrel shifters, a maximum shift of 12-positions warrants 4 levels of MUX. So the latency attributed to the shift operation is $4 T_{MUX}$, where T_{MUX} is the latency of a 2-1 multiplexer. A 12-iteration CORDIC operation involves 12 additions and 12 shifts and delay due to use of registers. We assume that T_{F-F} is the latency of a register. We can also assume from [57] that $T_{F-F} \approx T_{FA}$ and $T_{MUX} \approx 0.65T_{FA}$. Based on these values, the CORDIC computational latency for the method in [53] is $96 T_{FA}$ (i.e. $T_{CORDIC-ref} = [12 \times (4.32T_{FA} + T_{F-F} + 4T_{MUX})]$) while that of the proposed method is $99 T_{FA}$ ($T_{CORDIC-NEW} = [12 \times (4.64T_{FA} + T_{F-F} + 4T_{MUX})]$).

Hence, the initialisation latency for the method in [53] is $I_{ref} = 64 \times 96 = 6144T_{FA}$ while that of the proposed method is $I_{new} = 3 \times 99 + (64 \times 2 \times 4.64) = 891T_{FA}$. The initialisation stage of the proposed method is about 7 times faster now since the number of CORDIC operations is reduced. Referring back to the pipeline shown in Figure 3.5, the overall latency for [53] is $I_{ref} + (64^2 T_{CORDIC-ref}) = 399360T_{FA}$. For the proposed method, the overall latency is given by $I_{new} + ((64^2 - 1) \times 4.64) + (2 \times 99) = 19199T_{FA}$.

The latency estimates take into the account the pipelined processing of the two methods. For method [53], the CORDIC computations in Pipeline Stage A are done in parallel with the additions in Pipeline Stage B (see Figure 3.5). The latency of CORDIC in Pipeline Stage A is more time-consuming than the latency of the additions in Pipeline Stage B. Hence, the throughput of the pipelined implementation in [53] is governed by the latency of CORDIC, and this is used for the pipeline latency estimates. Similarly, for the proposed method, the addition with 25 bit operands to generate offsets is done in parallel with the addition of offsets and rotated centers using 10-bit operands. The former latency is therefore considered in the performance evaluations.

Figure 3.13 summarizes the performance findings and it is evident that an overall speed up of about 20 times is possible over the technique proposed in [53].

3.6.3 Area comparison

The architecture in [53] primarily comprises of a CORDIC engine, and 64^2 pairs of local-adders. The area of the CORDIC engine used in [53], which has three 20-bit adders and 20-bit registers (for each X, Y and Z datapath), a pair of 20-bit wide barrel shifters and a memory table is estimated to be $(20 \times 3 \times (2A_{FA} + A_{F-F})) + (20 \times 2 \times 4 \times A_{MUX}) + (20 \times 12 \times A_{F-F})$, where A_{FA} , A_{F-F} and A_{MUX} correspond to the area of a 1-bit Full Adder, Flip Flop and 2-1 MUX respectively. A 1-bit CLA is estimated to occupy twice the area of a 1-bit serial adder. The memory is assumed to be implemented as flip-flops and for a 12 iteration CORDIC, 12 arctan values are stored. This requires an area of $12 \times 20 A_{F-F}$. It can also be inferred from [57] that $A_{F-F} \approx 1.13A_{FA}$ and $A_{MUX} \approx 0.47A_{FA}$. Hence the CORDIC engine in [53] is estimated to occupy about $534 A_{FA}$.

Similarly, the area of the CORDIC engine used in the proposed method is about $668 A_{FA}$. In the proposed method, there is a CORDIC engine, a pair of adder-units to generate the rotated centers and the offsets, and the 64^2 sets of local-adders. The memory requirements to store the rotated centers is estimated to be about $64 \times 10 A_{F-F} = 723 A_{FA}$. All the local-adders in both methods are implemented as 10-bit CLAs. Assuming there are 64 pairs of them (one for each center's x and y coordinates), the area is estimated to be $64 \times 10 \times 2 \times 2 A_{FA}$. The adder-unit for generating the offsets in the proposed technique comprises of 25-bit CLAs (area = $100 A_{FA}$).

The area estimates are also summarized in Figure 3.13. It can be seen that the areas of the two architectures are comparable with the proposed method showing a marginal increase of about 3%.

3.6.4 Derivation of optimal hierarchy

The choice of the parameter h exerts a push-pull effect on the initialization latency and the offset generation. For example, lower levels of hierarchy will lead to larger window sizes, and this increases the latency of the offset computation. Alternatively, higher levels of hierarchy lead to more additions of the layer centers to obtain the rotated positions of the centers in the initialization phase. It is evident that the optimal hierarchy lies somewhere in between these two extremes.

In order to obtain the optimal choice of hierarchy given an image of size $m \times m$, we first represent the overall rotation latency estimate in terms of number of additions for various hierarchies. In order to derive these figures, first the number of CORDIC and addition computations contributing to the latency of the engine are listed. Then, the latency of the CORDIC is estimated to be k times that of an add

operation.

Derivation of k : To derive k , we assume the requirement of 1-pixel accuracy of the rotated position after truncation, since mapping is done onto a single pixel and no interpolation is performed. For this, $\log_2 m$ iterations would suffice within a CORDIC operation, each of which mainly comprise of one add one shift operation, apart from the register delays. Assuming that barrel shifters are used for shifting, which are implemented as a tree of multiplexers and by obtaining the relative delay of a 2-1 MUX with respect to a 1-bit Full Adder, we can estimate the latency of the CORDIC engine to be $1.6\log_2 m$ times that of an add operation. To get this value, we also assume that the size of registers used for addition is $n + \log_2 n$, where n is the number of iterations.

Figure 3.14 shows the expression for latency of rotation (in number of adds) for the proposed method in terms of hierarchy h and image size $m \times m$. It also shows the latency for the method in [53] with fixed window size $a \times a$.

Figure 3.15 shows a plot of the latency (in number of adds) in logarithmic scale plotted against varying hierarchies. It illustrates that extreme hierarchies do not lead to the most time-efficient implementation. Figure 3.16 shows the optimal hierarchy for images of varying image resolution. This corresponds to the hierarchy with the minimum latency for various resolutions in Figure 3.15.

	Method in Ghosh and Majumdar [7]		Proposed method	
	CORDIC	Addition	CORDIC	Addition
Initialisation	$(m/a)^2$	—	$h + 2$	$4^h(h - 1)$
max (Pipeline Stage A, Pipeline Stage B)	a^2	—	—	$(m/2^h)^2$
Total latency (number of adds)	$[(m/a)^2 + a^2]k$		$(h + 2)k + 4^h(h - 1) + (m/2^h)^2$	

$m \times m$: image size; k : $1.6 \log_2 m$; $a \times a$: window size; h : hierarchy

Figure 3.14: Latency expression as function of hierarchy

Figure 3.15: Choice of hierarchy

Image resolution, <i>m</i>	Optimal hierarchy
128	3
256	4
512	4
1024	4
2048	5
4096	5
8192	6
16 384	6
32 768	7
65 526	7
131 072	8

Figure 3.16: Optimal hierarchy

3.6.5 Latency estimate comparison for varying hierarchies

‘Gain’ is defined as the ratio of the latency of the method in [53] to the optimal latency of the proposed method. The latency measures are derived by employing the optimal hierarchy referring to Figure 3.16. Figure 3.17 shows a graph of gain plotted against varying image resolutions. It can be observed that there is a definite gain for all resolutions. The gain becomes more pronounced for images of higher resolutions as it increases in an exponential fashion. It is worth noting that in the initialisation phase, the additions for obtaining the rotated centers are assumed to be sequential. In order to further increase the gain, the local adders can be employed in the initialisation stage to perform the additions for obtaining the rotated centers in parallel. Also, the proposed system can employ improved CORDIC implementations to further increase the performance gain. However, since the majority of the operations in the proposed system consist of additions, the improvements in CORDIC will not lead to notable performance gain.

3.6.6 Effect of changing hierarchy on Area

The memory area to store the centers in the initialisation stage increases with the number of hierarchies. Furthermore, since the proposed architecture requires every center to have a local adder for addition of the offset in Pipeline Stage B (Figure 3.5), the adder area also grows proportionally. The number of such adders is $(m/a)^2$ and $(m/2h)^2$ for the method in [53] and the proposed method respectively, which is the same as the number of memory elements to store the centers. Treating the area of one memory element with its local adder together as a unit α , we plot the area in α for both methods in Figure 3.18. For the proposed method, the area corresponding to the configuration giving optimal performance is considered. Both methods use one CORDIC engine each and this area is negligible compared to the overall area consumed by the memory and the local adders.

Figure 3.17: Gain of proposed method over existing method

Figure 3.18: Effect of changing hierarchy on area

As can be observed in Figure 3.18, the proposed engine is more favorable in terms of area and latency for high image resolutions. It is noteworthy that this graph takes into account the optimal hierarchy and hence the areas look unattractive for lower resolutions. For cost sensitive solutions, a different hierarchy that leads to a lesser optimal solution can be chosen to satisfy the area constraint and at the same time sustain a considerable gain in performance. For example, the rotation engine described in Section 3.6 is 20 times faster with almost no increase in area. Further reduction of the area can be achieved by reusing a smaller set of adders for the computations.

3.7 Application to Endoscopy

In this section, the application of the proposed method for endoscopic images are discussed. The proposed image rotation method deals with only the coordinates of the image and not its gray levels. Hence, the performance and area costs of the technique is independent of the image content and only depends on the size of the image. A suitable hierarchy can be chosen based on the resolution of the camera used in the endoscope, using Figure 3.16.

An example is shown in Figure 3.19. Two consecutive image frames (a) and (b) are chosen from an endoscopy sequence such that the nature of relative displacement is visibly only a rotation. The size of images (a) and (b) is 352×240 . A hierarchy of 4 was applied, and rotation of (a) was performed for all angles ranging from 0° to 180° . The best match on applying correlation was found to be at 24° .

Figure 3.19: Rotation technique applied to endoscopy images

3.8 Summary

In this chapter, a novel method for detecting pure rotations of the camera by processing pairs of images from the sequence was proposed. Novel schemes to notably minimize the number of CORDIC operations to accelerate the process of image rotation were proposed. Hierarchical techniques to better manage the required number of additions and CORDIC operations were employed, thereby providing for area-time flexibility in the hardware realization. It has been demonstrated that this approach provides for substantial speed up with acceptable increase in area cost when compared with existing techniques. The effect of hierarchy on the computational efficiency has been analyzed so as to identify the optimal hierarchy for a given image resolution. In the case of 512×512 image, the hierarchy of 4 was shown to be optimal although the maximum possible hierarchy is 9. It has been shown that the increase in the data width to accommodate the total number of additions incurs marginal increase in the overall hardware cost. The proposed technique is highly scalable and lends well for high-speed rotation of high-resolution images.

After having tackled the problem of detecting pure camera rotations, the next important step in estimating camera motion is linear motion tracking (Figure 2.4). In the next chapter, techniques for tracking of linear motion of camera will be explored.

This is based on the study of the movement of feature points in an image sequence captured by the endoscope. The proposed method will be demonstrated on laboratory captured images. It will also be shown that the same method can be extended to detect rotations and a combination of linear and rotation movements as well.

Chapter 4

Proposed method for measuring linear motion of camera

In the previous chapter, we have seen methods to detect pure rotations of the camera in real time. However, in an endoscopic scenario, it is important to detect linear camera motions as well. This chapter focuses on tracking linear motion of camera through cylindrical endoscopic passages (Figure 4.1), by studying movement of feature points (FP). The outline of this method was presented earlier in Chapter 2. The component image processing steps such as image pre-processing, feature point extraction, feature point tracking, image unrolling and analysis in the unrolled domain are explained in this chapter. Novel methods to analyze and study the displacement of feature points in an unrolled domain in a compute efficient manner are proposed. It will also be shown how this method can be extended to detect rotations as well as a combination of linear and rotation movements. The proposed steps and overall method are verified using images captured in a simulated environment.

Figure 4.1: Estimation of camera motion through FP movement tracking

4.1 Image pre-processing

Prior to extracting the feature points, the image is subject to a pre-processing step to make the FP extraction easier. Since an edge-profile of an image would capture the major distinct areas of the image, it is envisioned to prove useful to work on this edge-profile as against working with the entire image. This not only helps to reduce the data set, but also makes it easy for FP extraction, since it retains only the important information. Regions showing high edge content usually indicate a feature point that can be easily recognized and tracked. Smooth and even areas of the surface would show no features and hence less information is available in the edge profile. FPs are then more easily spotted from this edge profile than from the original image.

4.1.1 Pre-processing using wavelets

To achieve this preprocessing prior to feature extraction, the images captured in the simulated environment are subject to wavelet decomposition. This not only helps to generate edges easily, but also helps to further reduce the data set by generating thumbnails of varying sizes. This is explained below:

1. Wavelets enable working on thumbnails of increasing size rather than directly on the entire image. A mother wavelet is used to construct wavelets at different scales (resolutions). A discrete wavelet transform (DWT) can be implemented as a filter bank, which decomposes the original image into horizontal (H), vertical (V), diagonal (D), and baseband (B) subbands, each being one-fourth the size of the original image.
2. Within the thumbnail, wavelets further facilitate working only on edge-information. The H , V , D and B subbands obtained from wavelet decomposition correspond to the high frequency components (or edges) in the horizontal, vertical and diagonal directions respectively. This not only helps in making clear the feature points which need to be extracted, but also helps in reducing the image data for processing.
3. Wavelets offer benefits for feature point tracking, which will be explained in Section 4.2.1.

After performing a wavelet decomposition, the horizontal, vertical and diagonal edges are combined by simply adding and normalizing the values to generate a consolidated edge image for further processing.

4.2 Feature point extraction and tracking

To facilitate analysis of feature point movement, any existing FP extraction and tracking methods can be employed. Several feature point extraction and tracking methods have been developed based on the nature of images and the application. In [58], Thomas *et al.* use local features based on color information, gray-scale histograms, texture and shape properties for locating feature points for the sake of medical image retrieval. In [44], Thorsten *et al.* use the Harris feature point detec-

tor [59] for FP extraction and subsequently the normalized cross-correlation method for correspondence analysis in order to perform 3D reconstruction of a scene from an endoscopic image sequence.

Feature extraction produces quantifiable diagnostic data that would be useful in detection of abnormalities such as presence of tumors etc. For example, [35] presents a colorectal lesions detector, an innovative detection system to support colorectal cancer diagnosis and detection of pre-cancerous polyps, by processing endoscopy images or video frame sequences acquired during colonoscopy. It utilizes second-order statistical features that are calculated on the wavelet transformation of each image to discriminate amongst regions of normal or abnormal tissue. An artificial neural network performs the classification of the features.

In another example [34], features are extracted from endoscopic images of the colon containing rich texture and color information. Novel schemes are developed to extract new texture features from the texture spectra in the chromatic and achromatic domains, and color features for a selected region of interest from each color component histogram of the colonoscopic images. These features are reduced in size using Principal Component Analysis (PCA) and are evaluated using Backpropagation Neural Network (BPNN). Based on the features extracted, image classification algorithms are used to analyze and classify images into categories like healthy and unhealthy regions. In [34], features extracted from endoscopic images were tested to classify the colon status as either normal or abnormal. The classification results show the features' capability for classifying the colon's status. Usually, this step is followed by a clinical decision in terms of diagnosis of abnormalities or administering treatment to the specific regions identified.

FP extraction method used: In our method, feature points are to be extracted from images that are reduced to edge profiles after pre-processing. The edge image should now be further processed to segment out regions containing unique feature points. Many criteria can be applied for this like selecting areas with highest entropy etc. In the case of images captured in the simulated environment, the pre-processed image mainly consists of white edges against a black background. Hence, a simple thresholding operation was first applied to make the pixels 1 bit valued (1 for white and 0 for black). Subsequently, occurrences of 1 and 0 are counted within each image sub-block and those giving the highest count for 1 are selected. This is because more edge information correspond to greater content of unique information compared to the black background.

FP tracking method used: Consider two consecutive images, $image_1$ and $image_2$ from the video, that are considered for processing. Any movement in camera causes the feature points selected in $image_1$ to be displaced in $image_2$. In order to locate the new positions of the FP, $image_2$ is searched for similar image content as the FP in $image_1$. Location of the FPs would give information about the displacement in the positions of the FPs, which can further be used for estimating the linear distance traversed by the camera. Image correlation was used as a measure to select the portions within $image_2$ that match closely with the content of FP in $image_1$. To elaborate, each feature point is represented as a small image of size $p \times q$, a subset of $image_1$. This sub-image is swept across $image_2$ of size $m \times n$ and at each step the correlation output between the FP and the overlapping area of $image_2$ is recorded. The correlation is a measure of the degree of match between the two images considered. The position of the sweep when a high correlation value is noted indicates a high probability of that being the new FP position.

However, this may not lead to the correct tracking because of the following reason: When the FP's move closer or further away, they are scaled up or down resulting in low correlation peaks in spite of the search coinciding with the correct position. We introduce a new parameter called s , denoting the scaling factor. $image_1$ is scaled by various values before performing the correlation. This correlation search is performed for a range of values s centered around 1. A value of 1 indicates no scaling, which typically follows zero camera movement.

4.2.1 Benefits of wavelets for FP tracking

Wavelets not only help in making the FP extraction easy, they also facilitate easy tracking of FPs. The reason is that the search can be done in an incremental fashion by working first with the low frequency layers and iteratively refining the process with higher frequency layers. A similar incremental algorithm called 'Iterative refinement algorithm' using wavelets is explained in [60]. Also, since only important information from the whole image is extracted, it enables working on a reduced data set, thereby making it more compute-efficient.

4.3 Analysis of FP displacement: Estimation of linear camera motion

After extracting feature points from the first image and matching their position in the second, we have the initial and final coordinate values of the FPs. Now, these displacements of FP positions are studied for estimating the linear motion and rotation of the camera. It was shown in Chapter 2 that a linear motion of the camera into the cylindrical tract causes feature points to move radially outward or inward. It is observed that the magnitude of displacement of feature points seen when the camera is moved upward/downward along the main axis, is dependent on the relative

distances of these points with respect to the camera. The points that lie physically closer to the camera show a greater apparent displacement when compared to points further away, for the same magnitude of movement of the camera. Points that are closer to the camera lie on circles with a larger radius compared to points that are further deep.

To verify that points at different depths show different apparent movements, an experiment was conducted using the laboratory setup explained in Section 2.5. Figure 4.2 (a) shows a sheet of paper with markings that simulate feature points. This is rolled into a cylinder and the snapshot of it is captured using the camera placed above it (shown in Figure 4.2 (b)). The camera is now moved downward along the axis by 4 cm and the cylinder is recaptured as shown in Figure 4.2 (c).

We now choose two points P and Q that are placed at different distances (depths) from the camera. Point P is closer to the camera and is seen more toward the periphery of the image as compared to point Q . P_i , P_f , Q_i and Q_f represent the initial and final coordinates of P and Q , in the image coordinate system. Initial coordinates are obtained from Figure 4.2 (b) and final coordinates from Figure 4.2 (c), using MATLAB. ΔP and ΔQ represent the displacements of P and Q (Euclidean distance from P_i to P_f and Q_i to Q_f). It can be seen from Figure 4.2 that the same movement of the camera by 4 cm causes ΔP to be almost three times the value as ΔQ . This shows that the apparent movement of P is higher than Q .

A consolidated value for camera motion is to be inferred from the various displacements indicated by the FP-pairs. We see that FPs situated at different depths undergo different apparent displacements, for the same movement in camera. Hence, to determine the actual movement of the camera using the displacements of FPs, these have to be standardized to even out the differences arising due to varying depths. To

Figure 4.2: Inconsistency in displacement magnitude

standardize the readings that each FP-pair gives us, we perform an operation called *image unrolling* followed by a *tan-theta normalization*. The process of unrolling sorts points based on depth information and creates a depth profile. Tan-theta normalization is applied on this depth profile to even out differences due to varying depths that were discussed above. These processes are explained in detail below and we discuss how they enable camera motion estimation.

4.3.1 Image unrolling process

To explain the concept of unrolling, we make use of Figure 4.3. Figure 4.3 (a) shows a cylinder and Figure 4.3 (b) shows the image as captured by a camera that is placed at the opening of the cylinder facing its axis. The unrolling process is equivalent to

cutting the cylinder along the dotted edge and flattening it into a 2-D surface. The unrolled image is illustrated in Figure 4.3 (c).

Figure 4.3: Image Unrolling Concept

Images captured by the camera along the axis of the cylinder reveal concentric circles, as seen in Figure 4.3 (b). Mathematically, unrolling refers to the process of mapping these circles of varying radii to straight lines of equal lengths, and then stacking these straight lines together to provide a flattened view of the cylinder. The length of the unrolled rectangle spans the angular range of 0° to 360° corresponding to the mapped circles. The breadth of the rectangle is proportional to the number of circles unrolled and stacked. The black band on the cylinder is revealed as a black patch between two concentric circles when captured by the camera. On unrolling, this band is displayed as a 2-D rectangle.

4.3.2 Computations involved in image unrolling

Consider a point P with coordinates (x, y) lying on a circle with radius r and origin (x_0, y_0) , as shown in Figure 4.4.

Figure 4.4: Unrolling transformation

If P makes an angle θ with the horizontal axis, then:

$$\begin{aligned} r &= \sqrt{(x - x_0)^2 + (y - y_0)^2} \\ \theta &= \tan^{-1} \left(\frac{y - y_0}{x - x_0} \right) \end{aligned} \quad (4.1)$$

Let the dimensions of the unrolled canvas be $L \times B$. Point P would map to a point P' with coordinates (x', y') after the unrolling transformation. Each line in the transformed domain has a length L and corresponds to 2π radians. This leads us to equation 4.2:

$$\begin{aligned} x' &= \frac{L\theta}{2\pi} \\ y' &= B - r \end{aligned} \quad (4.2)$$

Clubbing equations 4.1 and 4.2, we get the transformation equations as below:

$$x' = \frac{L \tan^{-1} \left(\frac{y-y_0}{x-x_0} \right)}{2\pi} \quad (4.3)$$

$$y' = B - \sqrt{(x-x_0)^2 + (y-y_0)^2} \quad (4.4)$$

4.3.3 Normalization

As explained earlier, unrolling sorts the points based on depth information. In other words, each row in the unrolled domain corresponds to a ring of tissue at a particular depth in the passage. The location of a FP in the unrolled image gives information about its depth in the cylinder. The differences in apparent displacements of FPs (refer to Figure 4.2) in the image domain are still carried forward in the unrolled domain. Although unrolling helps to generate a depth profile by arranging the FPs according to their depth in the cylinder, we still have not resolved the differences in the displacements of FPs, which arise due to their location at different depths. We now subject the depth profile obtained through unrolling to a normalization process to offset these differences arising due to varying depths.

The first step employed is to express the depth of a point in the cylinder in terms of its angle of view with respect to the camera. This is done because the mathematical expression for the normalization is derived more easily using this angular representation of the depth information. To explain better, we make use of Figure 4.5, where C refers to the camera capturing the image of the cylinder with radius R . Here, α refers to the maximum field of view of the camera. When the camera is placed facing the cylinder axis, the closest points on the cylinder captured by the camera (eg. point P) lie at an angle of view of α . Point Q refers to a point lying at a greater angular depth of θ_1 . When the camera is moved forward by a distance d , the same point Q (let's call this Q') now has a different angular depth of θ_2 .

Figure 4.5: Expressing depth as a function of camera angle

In order to derive the angular depth of a selected feature point, the y coordinate of that FP in the unrolled domain is studied. This is because the circular rings in the image are stacked up vertically in the unrolled domain, which means the y coordinate represents the count of the circle that is unrolled. The largest circle in the image is captured at the maximum field of view of the camera, α and has a radius of R_{im} . The depth of this circle in the unrolled domain is also given by R_{im} . A value of 0 in the y axis of the unrolled domain means an angular depth of 0° . The angular depths of the rest of the positions are calculated proportionally within this range of 0° and α . θ_1 is calculated using the y coordinate of the initial position of FP (U_{init_y}) while θ_2 is calculated using the y coordinate of the final position of FP (U_{final_y}), as shown below in Equation 4.5.

$$\begin{aligned}\theta_1 &= U_{init_y} \frac{\alpha}{R_{im}} \\ \theta_2 &= U_{final_y} \frac{\alpha}{R_{im}}\end{aligned}\tag{4.5}$$

Figure 4.6: Angular view with respect to camera

P , Q and Q' of Figure 4.5 are shown in terms of α , θ_1 and θ_2 in Figure 4.6. The plane where P , Q and Q' are projected is taken as the approximate image plane. δ_Q in Figure 4.6 refers to the angular displacement of point Q when the camera is moved by d . This displacement δ_Q illustrates that a linear movement of the camera into the cylinder along the axis makes the feature points move radially outwards, as shown in Chapter 2. In the unrolled domain, this is reflected by a vertical displacement of the points. For different depths of Q , the displacement δ_Q and the corresponding displacement in the unrolled domain are also of different magnitudes. We need to derive an expression for linear distance traversed by camera (d) in terms of the initial depth of FP (θ_1) and final depth of FP (θ_2), so that the depth information is factored. Referring to Figure 4.5, we can deduce the following equations:

$$\begin{aligned} \tan\theta_1 &= \frac{R}{D} \\ \tan\theta_2 &= \frac{R}{D-d} \\ d &= \frac{R}{\tan\theta_2} \left(1 - \frac{\tan\theta_2}{\tan\theta_1}\right) \end{aligned} \tag{4.6}$$

From Equation 4.6, it is seen that information on depth of feature point is factored into the final expression of distance moved by camera. As seen in Figure 4.2, FPs at different depths in the image show different displacements for the same movement of the camera. Through this expression from normalization, the actual displacement of the camera ' d ' is independent of the location of the FP chosen in the image. To reiterate, unrolling facilitates deriving of θ_1 and θ_2 , which can further be used in Equation 4.6 to deduce d . The distance d obtained above is the distance moved downward by the camera in absolute units (in this case cm , since radius was also measured in cm).

4.3.4 Verification of distance tracking

This analysis in the unrolled domain to detect linear distance traversed is verified by using test images created using Microsoft Visio that mimics a sequence of images of a cylinder captured with the camera placed at the axis, moving down linearly along the axis. Figure 4.7 shows a set of images with the feature points moving radially outward. To facilitate easy analysis of the FP displacements, FPs here are obtained through manual mouseclicks at the markings and the center for unrolling is fed into the program. The algorithm is coded in MATLAB to perform unrolling, normalization and analysis.

Table 4.1 shows the coordinates of the feature points before and after unrolling. C_{init} and C_{final} refer to the initial and final coordinates of FPs respectively in the cylindrical domain. Similarly, U_{init} and U_{final} refer to the initial and final coordinates respectively of the FP in the unrolled domain. θ_1 and θ_2 are the angular depths calculated using Equation 4.5. d is derived using Equation 4.6. The size of the image is taken to be 256×256 and the radius of the cylinder is taken as 4 cm. The diagonal of the image is taken as the diameter of the biggest circle captured in

Figure 4.7: Image sequence for detection of linear movement

the image, at the maximum field of view of α (22.2° in this case).

It is seen that all the 16 FPs considered indicate that the distance estimated is about 1.74 cm when the first image pair (*image1* and *image2*) is considered, which means the motion of the camera is in a downward direction into the cylinder. The second image pair also indicates a movement of the camera downward (FPs moving radially outward) recording around 2.17 cm. The effect of normalization can be seen by comparing the first and second image pairs. Although in the image, the second image pair shows double the amount of FP displacement than the first, the distances recorded do not follow the same ratio. As explained earlier in this section, the movement of points in the periphery of the image corresponds to a lesser movement of the camera when compared to the same movement of points located at the image center.

Table 4.1: Inferring forward distance of camera

Cylindrical Domain		Unrolled Domain		$\theta_1 = U_{init_y} \frac{\pi}{Y}$	$\theta_2 = U_{final_y} \frac{\pi}{Y}$	$d = \frac{R}{\tan \theta_2} (1 - \frac{\tan \theta_2}{\tan \theta_1})$
C_{init}	C_{final}	U_{init}	U_{final}			
Image 1 & Image 2						
(198, 251)	(174, 233)	(222, 176)	(220, 209)	0.38	0.45	1.77
(175, 313)	(141, 305)	(72, 176)	(71, 208)	0.38	0.45	1.72
(174, 378)	(141, 384)	(2329, 175)	(2329, 207)	0.38	0.45	1.74
(493, 250)	(518, 232)	(979, 175)	(977, 207)	0.38	0.45	1.74
(520, 316)	(553, 308)	(1135, 177)	(1134, 209)	0.38	0.45	1.70
(518, 378)	(553, 387)	(1270, 179)	(1272, 210)	0.39	0.45	1.62
(250, 199)	(233, 173)	(379, 176)	(377, 206)	0.38	0.44	1.63
(318, 176)	(311, 142)	(537, 173)	(535, 208)	0.37	0.45	1.91
(381, 177)	(388, 143)	(678, 176)	(677, 208)	0.38	0.45	1.72
(251, 494)	(233, 520)	(2017, 176)	(2017, 210)	0.38	0.45	1.81
(313, 517)	(306, 551)	(1872, 177)	(1871, 210)	0.38	0.45	1.75
(378, 519)	(383, 552)	(1730, 176)	(1730, 210)	0.38	0.45	1.81
						1.74
Image 1 & Image 2						
(172, 233)	(123, 199)	(219, 207)	(222, 267)	0.45	0.57	2.20
(141, 307)	(82, 295)	(71, 208)	(72, 269)	0.45	0.58	2.21
(141, 382)	(83, 394)	(2332, 209)	(2329, 267)	0.45	0.57	2.11
(520, 230)	(569, 196)	(976, 209)	(974, 270)	0.45	0.58	2.19
(553, 308)	(612, 298)	(1132, 210)	(1132, 271)	0.45	0.58	2.18
(553, 384)	(610, 397)	(1269, 210)	(1273, 270)	0.45	0.58	2.15
(232, 173)	(199, 122)	(377, 208)	(378, 268)	0.45	0.58	2.18
(311, 142)	(301, 81)	(535, 207)	(535, 269)	0.45	0.58	2.26
(388, 142)	(400, 83)	(677, 209)	(677, 269)	0.45	0.58	2.16
(233, 522)	(199, 572)	(2018, 210)	(2020, 270)	0.45	0.58	2.15
(307, 552)	(295, 610)	(1871, 210)	(1872, 269)	0.45	0.58	2.12
(384, 553)	(394, 612)	(1730, 210)	(1731, 270)	0.45	0.58	2.15
						2.17
Image 1 & Image 2						
(121, 200)	(148, 217)	(219, 268)	(220, 236)	0.58	0.51	-1.04
(83, 296)	(112, 301)	(70, 267)	(72, 238)	0.57	0.51	-0.94
(80, 394)	(111, 390)	(2330, 270)	(2329, 240)	0.58	0.52	-0.95
(569, 196)	(544, 215)	(974, 270)	(977, 238)	0.58	0.51	-1.02
(610, 298)	(580, 302)	(1131, 269)	(1130, 239)	0.58	0.51	-0.96
(612, 396)	(581, 391)	(1271, 270)	(1272, 241)	0.58	0.52	-0.92
(198, 123)	(217, 148)	(376, 268)	(379, 237)	0.58	0.51	-1.00
(300, 82)	(307, 112)	(534, 268)	(536, 238)	0.58	0.51	-0.97
(400, 82)	(394, 114)	(677, 270)	(677, 237)	0.58	0.51	-1.06
(201, 572)	(217, 547)	(2017, 269)	(2017, 239)	0.58	0.51	-0.96
(294, 611)	(301, 580)	(1874, 270)	(1872, 239)	0.58	0.51	-0.99
(396, 611)	(389, 582)	(1728, 270)	(1731, 240)	0.58	0.52	-0.95
						-0.98

The third image pair shows an upward motion of the camera since the FPs move radially inward this time. This is reflected by a negative value (-0.98 cm) estimated as the distance moved by the camera.

4.4 Extending method to detect camera rotation

Analysis of the image in the unrolled domain can be extended to study rotations of the camera. Any movement of the camera in the cylinder causes feature points to move in various directions. Unrolling standardizes the direction of motion and provides valuable information about whether the camera has been rotated or not during the elapsed time interval. If the direction of displacement of all FPs is downward in

the unrolled domain, that means there has been no rotation but a pure downward motion (Figure 4.8 (a)).

For the case of just a pure rotation of the camera about the axis, the displacements of all FPs would be horizontal in the unrolled domain. (Figure 4.8 (b)). A combination of rotation and linear motion of the camera, as seen in the unrolled domain is illustrated in Figure 4.8 (c). Unrolling can be used not just for detecting rotations, but also to monitor the *amount* of rotation. The greater the horizontal component, greater is the rotation and vice versa. This is verified in Section 4.4.1.

For estimating rotations, there is no need for calculating angular depths of FPs and normalization. This is because irrespective of how deep the points are in the cylinder, the apparent rotations remain the same. Let us call the horizontal FP displacement as ΔX . The ratio of ΔX to the length of the unrolled image directly translates into the rotation of the camera. If ΔX tends to the length of the unrolled image, then this reports the maximum rotation detectable by the method of 360° . Depending on the location of the FPs, the system either reports a positive value or an equivalent negative value for the rotation. This is verified by an experiment discussed in Section 4.4.1.

Figure 4.8: Movement of feature points in different domains

4.4.1 Verification of rotation tracking

In section 4.3.4, the analysis of vertical components of the FP displacements in the unrolled domain was verified for estimating linear motion of camera. Similarly, in this section the analysis of horizontal components of FP displacement in the un-

Figure 4.9: Image sequence for rotation detection

rolled domain is verified using test images for estimating rotation. Figure 4.9 shows a sequence of images mimicking a scenario where the camera undergoes a series of rotation movements. To facilitate easy analysis of the FP displacements, FPs here are obtained through manual mouseclicks at the markings and the center for unrolling is fed into the program. The algorithm is coded in MATLAB to perform unrolling.

Table 4.2 shows the initial and final coordinate values of the 16 FPs chosen, in both the cylindrical and unrolled domains (represented by C_{init} , C_{final} , U_{init} , and U_{final}). The horizontal displacement of the FPs is indicated by ΔX , which is given by the difference between the x coordinates of the FPs in the unrolled domain ($U_{final_x} - U_{init_x}$). The length of the unrolled rectangle, L , is taken as 2400 units. The ratio of ΔX to L provides the angle of rotation. It is seen that the magnitude of rotation is estimated as the expected values of $+22.5^\circ$, 45° (same as -315° reported for some FPs) and -22.5° .

Table 4.2: Inferring rotation of camera

Cylindrical Domain		Unrolled Domain		ΔX	$rot = \frac{\Delta X}{L} \times 360^\circ$ ($L = 2400pixels$)
C_{init}	C_{final}	U_{init}	U_{final}		
Image 1 & Image 2					
(174, 463)	(142, 382)	(2178, 1291)	(2328, 1293)	150	22.50
(148, 480)	(111, 390)	(2174, 1261)	(2325, 1260)	151	22.60
(121, 495)	(82, 393)	(2178, 1230)	(2328, 1232)	150	22.50
(520, 230)	(551, 308)	(979, 1291)	(1127, 1290)	148	22.26
(545, 212)	(580, 304)	(978, 1259)	(1128, 1261)	150	22.64
(571, 196)	(612, 298)	(978, 1229)	(1128, 1229)	150	22.46
(232, 174)	(311, 142)	(379, 1293)	(531, 1293)	152	22.80
(216, 148)	(306, 111)	(381, 1263)	(531, 1261)	150	22.62
(199, 122)	(301, 81)	(381, 1232)	(531, 1231)	150	22.63
(463, 520)	(383, 552)	(1576, 1290)	(1728, 1290)	152	22.73
(479, 545)	(390, 580)	(1577, 1260)	(1725, 1261)	148	22.10
(497, 573)	(394, 611)	(1578, 1227)	(1727, 1230)	149	22.36
					22.51
Image 2 & Image 3					
(142, 382)	(172, 233)	(2328, 1293)	(225, 1293)	-2103	-315.40
(111, 390)	(147, 217)	(2325, 1260)	(223, 1263)	-2102	-315.30
(82, 393)	(121, 202)	(2328, 1232)	(224, 1233)	-2104	-315.40
(551, 308)	(523, 459)	(1127, 1290)	(1427, 1289)	300	45.00
(580, 304)	(547, 474)	(1128, 1261)	(1426, 1261)	298	44.70
(612, 298)	(574, 490)	(1128, 1229)	(1425, 1229)	297	44.55
(311, 142)	(461, 171)	(531, 1293)	(832, 1290)	301	45.15
(306, 111)	(477, 148)	(531, 1261)	(833, 1262)	302	45.30
(301, 81)	(494, 122)	(531, 1231)	(833, 1231)	302	45.30
(383, 552)	(233, 522)	(1728, 1290)	(2028, 1290)	300	45.00
(390, 580)	(216, 547)	(1725, 1261)	(2029, 1260)	304	45.60
(394, 611)	(200, 572)	(1727, 1230)	(2029, 1230)	302	45.30
					44.97
Image 3 & Image 4					
(121, 199)	(80, 295)	(221, 1231)	(72, 1230)	-149	-22.35
(145, 217)	(112, 301)	(222, 1261)	(72, 1262)	-150	-22.50
(172, 233)	(142, 305)	(219, 1293)	(74, 1292)	-145	-21.75
(574, 490)	(612, 397)	(1420, 1229)	(1273, 1228)	-147	-22.05
(547, 475)	(581, 391)	(1422, 1260)	(1272, 1259)	-150	-22.50
(521, 459)	(553, 385)	(1428, 1290)	(1272, 1288)	-156	-23.40
(461, 173)	(388, 142)	(824, 1292)	(677, 1291)	-147	-22.05
(478, 149)	(394, 113)	(825, 1262)	(677, 1262)	-148	-22.20
(494, 123)	(400, 84)	(826, 1232)	(677, 1232)	-149	-22.35
(233, 522)	(307, 552)	(2018, 1290)	(1871, 1290)	-147	-22.05
(217, 546)	(301, 581)	(2022, 1262)	(1872, 1260)	-150	-22.50
(199, 572)	(294, 611)	(2021, 1230)	(1874, 1229)	-147	-22.05
					-22.31

It has been shown how the motion vectors in the unrolled domain give information about the magnitude of linear movement of the camera as well as rotation. The vertical component is directly related to the linear motion and the horizontal component corresponds to the rotation of the camera. This combined approach to tracking rotation and linear camera motions is shown in Figure 4.10 which shows the flow between the various computational blocks for estimation of linear motion and rotation of camera.

Figure 4.10: Computational blocks for linear and rotation movement estimation

4.5 Results

For initial proof of concept and verification at each processing step in the project, images in a simulated environment are used with a set up shown in Figure 2.5 in Chapter 2. Subsequently, results are demonstrated on real endoscopic images in Chapter 5.

4.5.1 Results: Image pre-processing

It was explained in Section 4.1 that wavelets are used for pre-processing the images before extracting feature points. Figure 4.11 shows a sample result after applying wavelet decomposition. The original sheet of paper (with markings simulating feature points) that was rolled as a cylinder and its captured image are shown in Figure 4.11 (a) and Figure 4.11 (b) respectively. Figure 4.11 (c) shows a snapshot of the image

decomposed into the three edge layers (H , V and D). Figure 4.11 (d) shows the combined edge image after addition. Clearly, it is seen that the image after pre-processing now has been reduced to essential information about the unique patterns, from which feature points can be extracted easily.

Figure 4.11: Edge extraction using wavelets

4.5.2 Results: Feature point extraction

As seen in Figure 4.11 (d), the pre-processed image mainly consists of white edges against a black background. As explained in Section 4.2, this edge profile is subject to a thresholding operation to convert it into a black and white image, thereby further reducing the data to be processed for FP extraction. For FP extraction, regions showing high edge content (reflected by white areas) are selected. Figure 4.12 shows

Figure 4.12: Feature point extraction

extraction of four such feature points corresponding to areas having highest white content. When real images are considered, this becomes a more serious problem since the wavelet decomposed edges would not offer such a clear distinction between background and foreground.

4.5.3 Results: Feature point tracking

A correlation search is done to track the feature points just extracted. This is done for a range of values for the scaling factor (Section 4.2), s , centered around 1.0. A value of 1.0 indicates no scaling, which typically follows zero camera movement. In the program used s values of 0.7 to 1.6 were chosen with an interval of 0.1. Figures 4.13 (a) and (b) show the edge profiles of $image_1$ and $image_2$ after the wavelet pre-processing. Figure 4.13 (c) shows the four FPs selected from $image_1$ based on highest edge content. The correlation search is applied to locate these FPs in $image_2$, and the results obtained are shown in Figure 4.13 (d). Figure 4.13 (e) shows a plot of the correlation values for different values of s and reveals a peak at $s = 1.5$, which implies that the FPs have been scaled up roughly one and a half times its original size.

Figure 4.13: Feature point tracking using correlation search

4.5.4 Results: Image unrolling

The unrolling method explained in Section 4.3.1 helps to transform the image into an unrolled domain for analysis. In the overall method, this unrolling process is applied only to the FPs selected to save computations. However, to present the results of this step better in this section, unrolling is applied to the entire image, captured using the experimental setup. All codes were written in MATLAB. Figure 4.14 shows a snapshot of the cylinder captured using the camera. Applying the unrolling equations 4.3, we obtain Figure 4.14 (b).

Figure 4.14: Image unrolling experiment

4.5.5 Results: Normalization

In Section 4.3.3, it was explained that through the process of normalization, any discrepancies arising due to variations in depth of the FPs are evened out. In the overall method, this process of normalization is reflected in the computation of d , the distance traversed by the camera (Equation 4.6). In this section, to present results better, normalization is applied to the entire unrolled image. This is done by applying the same ‘tan-theta’ factor to the y coordinate of the unrolled domain. It is expected that the circles lying deeper in the cylinder are represented as thicker lines compared to those lying on the periphery of the image.

Figure 4.14 (b) shows the unrolled representation of the image in Figure 4.14 (a). This is without any normalization and it is evident that it is not in proportion. Figure 4.14 (c) shows the same image after subjecting it to normalization. Now, the deeper regions of the cylinder stretch out, making the transformed image resemble the sheet of paper before rolling (Figure 4.14 (d)). Some noise is inevitable due to imperfections in the geometry of the test image.

4.5.6 Results: Combined method for tracking linear motion and rotation

To illustrate the overall combined method for tracking both the distance and rotation of the camera, we consider a sequence of images captured by the laboratory setup. Images of a rolled cylinder are captured from various heights of the camera. *Frame1* was captured at a height of 30 cm from the ground and *Frame7* at 24 cm from ground level, with no rotations of the camera. The rest of the intermediate frames were captured with approximately 1 cm change in camera height. So the net distance traversed by the camera is 6 cm and the rotation is nil, with a measurable accuracy of 0.5 cm and 1° respectively. The size of the images are 640×480 , captured using a National Semiconductor LM9628 camera. It's maximum field of view is 22.5 degrees and the radius of the paper cylinder used was 4.5 cm. Since it is a rectangular image, it is assumed that the diagonal of the image subtends the maximum field of view (Figure 4.6). Feature points are extracted, tracked and subject to unrolling using MATLAB. The initial and final coordinate positions in the unrolled domain are now analyzed.

The proposed method processes images in pairs where FPs are first located in the first image and their corresponding locations are tracked in the second. These coordinates are further analysed to estimate the linear distance and rotation motions

Table 4.3: Estimation of camera motion on image sequence

<i>Frame1</i>	<i>Frame2</i>	(x_i, y_i)	(x_f, y_f)	(ux_i, uy_i)	(ux_f, uy_f)	dist (cm)	rot (deg)
(118,253) (498,339) (316,130) (166,443)	(106,249) (500,342) (313,124) (156,449)	(76,1253) (1300,1357) (504,1322) (2159,1260)	(78,1240) (1306,1354) (501,1315) (2161,1248)	0.91 0.90 0.93 0.89	-0.33 -0.93 0.44 -0.23
						0.91	-0.27
(106,249) (503,302) (312,124) (156,449)	(97,250) (510,301) (313,116) (148,460)	(108,1288) (1188,1307) (604,1316) (2117,1291)	(102,1279) (1187,1300) (606,1308) (2112,1278)	0.88 0.86 1.05 1.34	0.90 0.22 -0.27 0.70
						1.03	0.39
(159,271) (544,166) (349,61) (301,450)	(150,268) (551,157) (347,50) (298,453)	(75,1290) (939,1269) (576,1248) (1996,1348)	(77,1281) (935,1258) (574,1237) (1969,1344)	0.96 0.93 0.80 0.80	-0.31 0.60 0.28 -0.54
						0.87	0.01
(151,269) (487,262) (309,108) (301,451)	(142,265) (490,260) (307,100) (298,454)	(86,1280) (1036,1366) (500,1283) (1971,1352)	(89,1271) (1034,1362) (501,1274) (1975,1348)	0.92 0.90 0.87 0.88	-0.47 0.24 -0.03 -0.53
						0.89	-0.20
(71,242) (541,292) (344,34) (195,453)	(56,236) (550,289) (342,17) (188,459)	(92,1228) (1172,1293) (613,1227) (2092,1298)	(95,1212) (1168,1284) (609,1210) (2094,1289)	0.95 0.96 0.99 1.00	-0.45 0.62 0.51 -0.24
						0.98	0.11
(133,258) (550,287) (380,31) (297,457)	(122,253) (556,281) (378,14) (288,457)	(109,1261) (1124,1308) (623,1205) (1975,1353)	(111,1249) (1115,1301) (620,1189) (1995,1349)	0.93 0.87 0.90 0.87	-0.39 1.35 0.56 -3.06
						0.89	-0.38
						5.57	-0.35

of the camera. The coordinates of the feature points in the first image of the pair are referred to as (x_i, y_i) and the corresponding coordinates in the second are referred to as (x_f, y_f) . (ux_i, uy_i) and (ux_f, uy_f) are used to denote the initial and final positions in the unrolled domain. The displacements are calculated and normalized using Equation 4.6. Rotations are inferred from the horizontal components of FP displacements, as shown in Section 4.4.

Table 4.3 shows the initial and final coordinate values of the feature points in the image as well as in the unrolled domain for every image pair. It can be seen that the net distance moved by the camera as estimated by the system is about 5.57 cm. The distance recorded experimentally was 6 cm, indicating a 7% error in the distance measured using the proposed method. The rotation was detected to be a negligible value of -0.35° , which coincides with the experimental value since no camera rotation was performed while capturing the images. The error noted in rotation is 0.09%.

4.5.7 Effect of radius and FOV parameters

Table 4.4: Tracking Camera Distance for varying FOV and Radius of Cylinder

Varying Radii of Cylinder				Varying FOV of Camera		
Field of View (degrees) = 22.2				Radius = 2 units		
Radius (units)	Dist (units)	Dist (Normalized)		FOV (degrees)	Dist (units)	Dist (Normalized)
1.5	FP1	0.320	1.428	15.0	1.405	1.418
	FP2	0.224	1.000		0.991	1.000
	FP3	0.423	1.889		1.868	1.885
	FP4	0.442	1.974		1.943	1.961
2.0	FP1	0.426	1.428	20.0	1.061	1.424
	FP2	0.299	1.000		0.745	1.000
	FP3	0.564	1.889		1.406	1.888
	FP4	0.589	1.974		1.467	1.969
2.5	FP1	0.533	1.428	25.0	0.857	1.433
	FP2	0.373	1.000		0.598	1.000
	FP3	0.705	1.889		1.130	1.891
	FP4	0.737	1.974		1.184	1.981
3.0	FP1	0.640	1.428	30.0	0.722	1.444
	FP2	0.448	1.000		0.500	1.000
	FP3	0.846	1.889		0.947	1.895
	FP4	0.884	1.974		0.997	1.994
3.5	FP1	0.746	1.428	35.0	0.627	1.457
	FP2	0.523	1.000		0.430	1.000
	FP3	0.987	1.889		0.818	1.900
	FP4	1.032	1.974		0.865	2.011
4.0	FP1	0.853	1.428	44.4	0.510	1.488
	FP2	0.597	1.000		0.343	1.000
	FP3	1.128	1.889		0.655	1.911
	FP4	1.179	1.974		0.702	2.050

To calibrate the estimates derived in Table 4.3, the field of view of the camera and radius of paper cylinder were set to be 22.2° and 4.5 units respectively. In this section we study the effect of offsets in these two parameters: field of view and radius of cylinder. This analysis is useful in cases when these parameters are not easily known, as in real endoscopy videos (Section 5.2). For this study, distance estimates are computed for different radii of cylinder and for different fields of view of camera,

for a given set of feature point locations. Since estimation of rotation is primarily governed by the ratio of the lengths of horizontal motion vectors and the length of the unrolled image, it not affected by these settings. Hence, only linear distance estimates are considered.

Table 4.4 shows the linear distance estimates for four different camera movements, normalized by dividing with the lowest distance obtained in that set. It can be seen that the normalized distances remain the same irrespective of the value of radius chosen. This is because the radius scales all outputs uniformly by a certain value (4.6), resulting in the same amount of relative movement. In general, increasing the radius causes larger unnormalized displacement values. However, changing the angle of view results in small differences in distance estimates. This is plotted in Fig. 4.15. It was found that a 100% offset in the field of view ($FOV = 44.4^\circ$) leads to a difference of a mere 4.0% in the normalized estimate, for a displacement of 2 normalized units, showing that the estimates largely remain the same for large variations in the calibration parameters.

Figure 4.15: Distance Estimate versus Field of View

4.6 Discussion

The results for the combined tracking of camera rotations and linear movements are obtained using 4 feature points in Section 4.5.6. It is noteworthy that using 4 FPs per image pair is drastically lesser than the existing methods. For example, method in [44] uses 930 feature points for tracking camera motion for a virtual endoscopy application.

The proposed method presented in this chapter provides a quick estimate of the rotation and linear motions of camera. The results reported a difference of 7% in distance tracking and 0.09%, well within the measurable accuracy of the system. Some of the potential reasons for inaccuracies in the system include:

- Imperfect geometry of cylinder: The mathematics behind the proposed unrolling method is for perfect cylindrical tracts, that project as circles in the image. In the laboratory simulated images, sheets of paper were first rolled into cylinders and any imperfections in this would cause errors in the analysis.
- Center for unrolling: Differences in the position of the center for unrolling due to manual approximation could also cause variations in the output. In order to have a good center estimation, an automated algorithm is proposed in the next chapter.
- Barrel distortion effects of wide angle lens: In all the cases considered, there is no visible distortion due to the camera lens. Hence the proposed method remains largely unaffected by these changes. However, if the use of wide angle lenses causes severe distortions, then it would introduce errors in analysis.
- 3D features: The feature points in a real endoscopy set up consist of three dimensional polyps and other structures. If 3D features that extend into the

tract significantly are chosen as feature points, then they may give misleading information about their depth. In other words, the tip of a polyp present close to the camera but extending into the cavity would give the same depth information as another point that is present much deeper. To overcome these differences, a possible approach is to process a larger number of feature points and then take the median of the outputs.

It is also noteworthy that the proposed method works for dynamic conditions where the feature points can change. If this system is used to aid in endoscopic re-traversal, then it is not necessary that the conditions within the tract remain the same. In other words, the distance and rotation tracked do not depend on which feature points are chosen. Differences arising due to varying depths of FPs are nullified by the normalization process. Overall, the motion estimated adopts a quick approach using only a handful of feature points and is very suited to meet real time requirements.

4.7 Summary

In this chapter, the proposed method for estimating linear camera motion through cylindrical passages is explained. Initial and final coordinate positions of the feature points are obtained by employing image processing based FP extraction and tracking. Novel methods for analysis of feature point displacements in an unrolled domain are proposed. An unrolling operation is performed followed by normalization to even out discrepancies arising because of varying FP locations. It was shown that this method can be extended to determine rotations as well. A combination of rotations and linear motions can be detected by analyzing the horizontal and vertical components of FP displacements. All the proposed individual steps have been verified using images in a simulated environment. Finally, the overall method for combined tracking of rotations and linear camera motions is demonstrated on a sequence of

image and the net distance and rotation estimated are verified against experimental data.

So far, the unrolling operation takes the coordinate of the center as a manual input. This is not desirable for an automated real time solution. In the next chapter, an improved version of this method is proposed where the unrolling process is automated to remove dependency on user input of the center.

Chapter 5

Automated unrolling

In the previous chapter, the proposed method for tracking linear camera motion was presented. Image unrolling forms an integral part of the proposed overall algorithm as explained previously in Chapter 4. The suggested unrolling process in equation 4.3 takes the coordinates of the center as an input. Circles of varying radii with respect to this center are mapped to straight lines in the unrolled domain. Manually feeding the center coordinates into the unrolling program implies intervention of the user each time unrolling had to be done. This is undesirable for a real time automated system. In this chapter, methods to eliminate dependence on an external input will be explored. An algorithm is proposed that automates the entire unrolling process by estimating the center coordinate through processing borders of the image. The overall method, with automated unrolling and combined linear/rotation tracking is then demonstrated on sequences of real images using this improved proposed technique.

5.1 Proposed method for center estimation

It is proposed here that instead of working with the entire image, only the left, right, top and bottom borders of the image are chosen for processing. The center for

unrolling is first assumed to be a random coordinate value (like the image center) and then iteratively fine-tuned by combining the results obtained on independently processing each of the sub-images (left, right, top and bottom borders), as explained below.

The basic concept used here is that when the center assumed for unrolling does not match with the correct center, then this error is reflected in the outputs generated by processing the borders. This error is fed back into the system for adjusting the assumption iteratively till the outputs agree with each other. The top and bottom borders work as a pair together to give information about the Y coordinate of the center. Similarly the left and right borders act as handles that help us infer the X coordinate of the center. As the center estimate tends to the correct value, the outputs of the four sub-images agree with each other and convergence is achieved. Firstly, we discuss how left-right and top-bottom borders work as pairs to give information on the X and Y coordinates of the center respectively. Secondly, we analyze the condition for convergence of the estimate with the actual center, where all border outputs match (Section 5.1.2).

5.1.1 Relationship between border displacement estimates

Figure 5.1: Experiment to derive relationship between estimated center and reference center, Q: reference center, A to P: assumed centers (a) initial image (b) final image

In order to derive a relationship between the border displacement estimates and the assumed center for unrolling, the following experiment was conducted. A pair of

consecutive images were taken from an image sequence, $image_1$ (Fig. 5.1 (a)) and $image_2$ (Fig. 5.1 (b)). They are used as input images for estimating the distance traversed by the camera. Let us select the left, right, top and bottom borders of a particular width from both the images and call them $border_L$, $border_R$, $border_T$ and $border_B$ respectively, as illustrated on $image_2$. Initial coordinates of feature points are extracted from the borders of $image_1$ and the corresponding final coordinates are extracted from the borders of $image_2$. Then, a center is assumed for unrolling and the resulting outputs, i.e. the border displacement estimates are studied in relation to the assumed center used for unrolling.

To study the effect of location of centers on these estimates, 16 different centers are assumed for unrolling at 16 random positions, A to P as shown on $image_1$ in Fig. 5.1. These are referred to as assumed centers or $C_{assumed}$. One feature point was selected from each border, which was subject to unrolling using each of the assumed centers A to P . Based on the unrolled coordinates, normalization was performed to infer linear distance traversed by the camera. Table 5.1 shows the camera distances (in cm) recorded by the left, right, top and bottom image borders, given by $dist_l$, $dist_r$, $dist_t$ and $dist_b$ respectively.

Now, for the sake of the study, a reference center, Q Fig. (5.1) is manually obtained by visual inspection. Let us call this center $C_{reference}$ as the coordinates of Q will act as a reference. It can be seen from the Table 5.1 that for all the assumed centers that lie above the $C_{reference}$, i.e. A , B , C , D , E , F , G and H , the distance recorded by the top border ($dist_t$) is greater than that of the bottom border ($dist_b$). Working backward, this imbalance in distance can be used to infer that the y coordinate of the assumed center should be adjusted to move it downward in the next iteration. For the rest of the center positions, $dist_b$ is higher than $dist_t$ which means the center should be moved upward. In a similar manner, the centers to the left of the reference center (A , B , E , F , I , J , M and N) show that left border displacement

Table 5.1: Relationship between border outputs

Assumed center	$dist_l$	$dist_r$	$dist_t$	$dist_b$	Adjustment	
A	2.06	0.40	1.33	0.62	→	↓
B	1.10	0.62	1.88	0.59	→	↓
C	0.69	1.04	1.96	0.48	←	↓
D	0.49	1.75	1.56	0.38	←	↓
E	2.20	0.45	1.04	0.95	→	↓
F	1.12	0.77	1.24	0.85	→	↓
G	0.72	1.43	1.27	0.62	←	↓
H	0.50	3.41	1.10	0.44	←	↓
I	1.78	0.47	0.69	2.18	→	↑
J	1.02	0.83	0.84	1.31	→	↑
K	0.67	1.63	0.78	0.88	←	↑
L	0.48	3.98	0.78	0.81	←	↑
M	1.29	0.45	0.53	5.15	→	↑
N	0.84	0.77	0.62	1.80	→	↑
O	0.59	1.35	0.63	0.89	←	↑
P	0.45	2.50	0.59	0.64	←	↑

estimate ($dist_l$) is greater than that of the right border ($dist_r$), indicating that the center should be moved rightward.

This observation was made on a number of test images leading to the following conclusion. When $dist_l$ is greater than $dist_r$, the assumed center is left of the reference center and hence should be moved rightward. Similarly, when $dist_t$ is greater than $dist_b$ the assumed center is above the reference center and should be moved downward. Based on this, the direction of refinement as suggested by the left-right and top-bottom output pairs is also shown in Table 5.1. The above observations are summarized below:

Inferences 1:

- $dist_l > dist_r \Rightarrow C_{reference}$ lies right of $C_{assumed}$
- $dist_r > dist_l \Rightarrow C_{reference}$ lies left of $C_{assumed}$
- $dist_t > dist_b \Rightarrow C_{reference}$ lies below $C_{assumed}$

- $dist_b > dist_t \Rightarrow C_{reference}$ lies above $C_{assumed}$

where $dist_l, dist_r, dist_t$ and $dist_b$ represent the distances of the camera as estimated by the left, right, top and bottom borders.

5.1.2 Conditions for Convergence

From Inferences 1 listed above, we can estimate the approximate position of the center for unrolling in order to further infer the motion of the camera. In this section, we discuss the conditions that are satisfied by the border outputs when the assumed center converges with the estimated center (i.e. $C_{assumed} \approx C_{estimate}$).

Since the camera is assumed to be moving linearly along the main axis of the cylinder in a symmetrical fashion, the distance recorded by the left, right, top and bottom image borders are expected to match. The corollary of this statement is that, if the left-right or top-bottom outputs do not match, the process of computing the distances could be erroneous. Since the unrolling process largely affects the distance measured, use of incorrect centers for unrolling would directly lead to wrong distances being estimated. In other words, when $C_{assumed} \approx C_{estimate}$, the linear and rotation motion recorded by the four borders coincide.

In order to verify this, a series of MATLAB simulations were run where the center estimates were made to vary and outputs were analyzed. Figure 5.2 shows *image2* obtained after rotation of *image1* with the feature points highlighted. The size of the image is 424×424 , with its correct unrolling center at the image center, (212, 212). In Figure 5.3, the X coordinate of the center is fixed at the correct location (212) and the Y coordinate is varied from 0 to 424 (breadth of the image), i.e., $C_{assumed}$ is (212, y), where $y \in [0, 424]$. Figure 5.3 shows the linear motion of the camera, as

Figure 5.2: (a) image1; (b) image2

Figure 5.3: Graph showing convergence of border outputs when $C_{assumed} \approx C_{estimate}$ detected by the four borders. It is seen that when $dist_l = dist_r = dist_t = dist_b$, then the center position is (212, 212), which equals $C_{estimate}$. When the correct center has been used for unrolling, then distance recorded by all four borders coincide.

Similar tests were performed keeping the Y coordinate fixed and varying the X coor-

dinate of the center. Similar graphs were produced where the point of convergence of $dist_l$, $dist_r$, $dist_t$ and $dist_b$ corresponds to the correct center for unrolling. Through these tests, we conclusively show the convergence of all border outputs at the correct unrolling center. In the next section, an algorithm is devised for estimating the center for unrolling using the border outputs.

5.1.3 Algorithm for center estimation

In Section 5.1.2, the conditions for convergence using the left, right, top and bottom border outputs were discussed. The left-right and top-bottom borders are treated as handles for refining the estimated center. The following algorithm has been presented that estimates the approximate center for unrolling purposes, prior to inferring the motion of the camera.

Algorithm 1 illustrates estimation of the center for an image size $m \times n$. We first start with the initial center position to be the midpoint of the image, $\left\{\frac{m}{2}, \frac{n}{2}\right\}$. The distance of linear motion is estimated after first performing an unrolling with this value as the center. If the right border returns greater magnitude of displacement, then it means that the assumed center is right of the actual center, so the x-coordinate is moved leftward by Δ_h for the next iteration; vice versa for rightward refinement. Similarly, if the top border returns a bigger displacement value compared to the bottom border, the center is shifted downward by Δ_v . The rationale for this adjustment of the center is derived from `Inferences:1`.

When the direction for left-right or top-down adjustments change, the value of Δ_v and Δ_h are decreased accordingly to facilitate convergence. When the displacements recorded by the left-right and top-bottom pairs match (i.e, $dist_l = dist_r$ and $dist_t = dist_b$), we obtain a fair estimate not only on the amount of displacement, but

Algorithm 1 Iterative estimation of center

```

1: image size :=  $m \times n$ 
2: estimated center :=  $(c_x, c_y)$ 
3:  $c_y := \frac{n}{2}$  // variable updated iteratively
4:  $c_x := \frac{m}{2}$  // variable updated iteratively
5:  $\epsilon = 0.05$  //Tolerable discrepancy in border displacement estimates
6:  $dir_h$  //Horizontal direction for  $c_x$  movement (left/right)
7:  $dir_v$  //Vertical direction for  $c_y$  movement (up/down)
8:  $\Delta_h := 1$  //Magnitude of horizontal refinement, 1: 20pixels, 2: 10pixels, 3: 5pixels
9:  $\Delta_v := 1$  //Magnitude of vertical refinement, 1: 20pixels, 2: 10pixels, 3: 5pixels
10: Perform unrolling using  $(c_x, c_y)$  to get  $dist_l, dist_r, dist_t, dist_b$ 
11: if  $|(dist_l - dist_r)| \geq \epsilon$  then
12:   if  $(|dist_l| - |dist_r|) \geq \epsilon$  then
13:      $dir_h := right$  //Move rightward
14:     if if previous value of  $dir_h$  was left  $\rightarrow$  right then
15:       set  $\Delta_h$  to lower value
16:     end if
17:      $c_x \leftarrow c_x + \Delta_h$ 
18:     goto 10
19:   else
20:      $dir_h := left$  //Move leftward
21:     if  $dir_h$  has changed direction from right  $\rightarrow$  left then
22:       set  $\Delta_h$  to lower value
23:     end if
24:      $c_x \leftarrow c_x - \Delta_h$ 
25:     goto 10
26:   end if
27: end if
28: if  $(|dist_t - dist_b|) \geq \epsilon$  then
29:   if  $(|dist_t| - |dist_b|) \geq \epsilon$  then
30:      $dir_v := down$  //Move downward
31:     if  $dir_v$  has changed direction from up  $\rightarrow$  down then
32:       set  $\Delta_v$  to lower value
33:     end if
34:      $c_y \leftarrow c_y + \Delta_v$ 
35:     goto 10
36:   else
37:      $dir_v := up$  Move //upward
38:     if  $dir_v$  has changed direction from down  $\rightarrow$  up then
39:       set  $\Delta_v$  to lower value
40:     end if
41:      $c_y \leftarrow c_y - \Delta_v$ 
42:     goto 10
43:   end if
44: end if
45: estimated center is  $c_x, c_y$ 

```

also on the location of the center.

Results of the automated method presented in this chapter are applied to real images in Section 5.2.

5.2 Results on real images

In this section, we present results of the proposed method for center estimation on real images, when applied to medical endoscopic images. Test videos of endoscopy were first obtained from the El Salvador Atlas of Gastrointestinal VideoEndoscopy [61]

and were subject to splitting into a stack of frames. Human esophageal endoscopy was chosen since the esophagus closely resembles a cylinder. Also a video segment where the movement of the camera is straight through the tract without too many jerks and tilts is chosen so that the traceability of FPs is maintained between every two frames. The frames are captured at the rate of 30 frames per second. The radius of the esophagus and field of view of the camera for these videos were not available. Based on the studies in Section 4.5.7 that these parameters do not largely affect the normalized estimates, they were taken to be 2 units and 22.5° respectively.

5.2.1 Center estimation

Table 5.2: Estimation of center

Iteration No	center coordinate	$dist_l$	$dist_r$	$\Leftrightarrow$	amount	$dist_t$	$dist_b$	$\Updownarrow$	amount
<i>Iter#1</i>	(176, 120)	2.10	0.44	$\rightarrow$	1	0.22	1.09	$\uparrow$	1
<i>Iter#2</i>	(186, 110)	1.31	0.56	$\rightarrow$	1	0.29	0.82	$\uparrow$	1
<i>Iter#3</i>	(196, 100)	0.78	0.66	$\rightarrow$	1	0.37	0.59	$\uparrow$	1
<i>Iter#4</i>	(206, 90)	0.47	0.69	$\leftarrow$	2	0.45	0.44	$-$	1
<i>Iter#5</i>	(201, 90)	0.52	0.62	$\leftarrow$	2	0.50	0.45	$\downarrow$	2
<i>Iter#6</i>	(196, 95)	0.68	0.60	$\rightarrow$	3	0.44	0.52	$\uparrow$	3
<i>Iter#7</i>	(198, 93)	0.61	0.61	$-$	3	0.47	0.48	$-$	3
$\epsilon := 0.05$

Figure 5.4: Test image considered for center estimation algorithm

From this image stack, a pair of images is chosen to demonstrate the proposed center

estimation algorithm, as shown in Figure 5.4. The variables obtained at the end of every iteration are presented in Table 5.2. The size of the image is 352×240 , so we start with the assumption that the center is at the mid point, $A(176, 120)$ (Figure 5.4) . It was seen that after 7 iterations the outputs converge and the center is estimated to be at $B(198, 93)$. It can be visually seen from Figure 5.4 that B is the center of the circular tract.

In the process of estimating the center, the distance is also estimated by averaging $dist_l$, $dist_r$, $dist_t$ and $dist_b$. To process the next pair of images in the sequence, we start with the center position estimated in the previous computation and then refine it by applying Algorithm 1. This is because the new center is expected to be in the vicinity of the previous position and gives a better starting point for convergence compared to the midpoint of the image. Similarly, the distances are recorded for every image pair during the first traversal.

5.2.2 Overall algorithm on image sequence

The overall algorithm employing the left, right, top and bottom handles is now demonstrated on a sequence of real endoscopic images. Table 5.3 shows an image sequence (Frame 56 to Frame 68) of an endoscopy procedure of the esophagus, captured at 30 frames per second. Every consecutive pair of images in the sequence are processed and the estimates of linear distance and rotation are shown in Table 5.3. The net distance and rotation, estimated by the proposed algorithm, are highlighted in bold in the tables. It can be seen that the rotation that is detected is almost zero, as can be visually verified by following the image sequence.

In order to save computations, instead of processing every frame in the sequence as image pairs, alternate frames are selected and paired within the same sequence

Table 5.3: Results on consecutive frames

$frame_i$	$frame_{i+1}$	cen		(x_i, y_i)	(x_f, y_f)	(ux_i, uy_i)	(ux_f, uy_f)	dist	rot
56	57	(180, 105)	<i>l</i>	(157, 149)	(157, 150)	(1984, 1450)	(1980, 1449)	0.47	0.01	
			<i>r</i>	(256, 132)	(258, 133)	(1330, 1419)	(1332, 1417)	0.44	0.00	
			<i>t</i>	(204, 37)	(206, 35)	(730, 1428)	(736, 1425)	0.63	-0.02	
			<i>b</i>	(174, 193)	(174, 197)	(1826, 1412)	(1825, 1408)	0.65	0.00	
								0.55	0.00
57	58	(198, 92)	<i>l</i>	(129, 106)	(126, 102)	(2324, 1430)	(2347, 1427)	0.60	-0.07	
			<i>r</i>	(254, 132)	(256, 133)	(1437, 1431)	(1435, 1429)	0.60	0.01	
			<i>t</i>	(202, 38)	(203, 37)	(628, 1446)	(635, 1445)	0.48	-0.02	
			<i>b</i>	(189, 179)	(190, 182)	(1839, 1413)	(1834, 1410)	0.49	0.02	
								0.54	-0.02
58	59	(178, 104)	<i>l</i>	(130, 98)	(129, 99)	(48, 1452)	(39, 1451)	0.49	0.02	
			<i>r</i>	(250, 132)	(252, 133)	(1342, 1423)	(1343, 1421)	0.48	0.00	
			<i>t</i>	(202, 41)	(203, 39)	(739, 1433)	(740, 1430)	0.63	0.00	
			<i>b</i>	(148, 198)	(148, 203)	(1918, 1401)	(1912, 1397)	0.62	0.02	
								0.56	0.01
59	60	(193, 98)	<i>l</i>	(138, 95)	(137, 96)	(21, 1445)	(14, 1444)	0.41	0.02	
			<i>r</i>	(254, 80)	(255, 79)	(1090, 1436)	(1086, 1435)	0.40	0.01	
			<i>t</i>	(201, 42)	(203, 41)	(654, 1443)	(666, 1442)	0.53	-0.03	
			<i>b</i>	(188, 182)	(187, 185)	(1823, 1416)	(1826, 1413)	0.56	-0.01	
								0.48	0.00
60	61	(198, 96)	<i>l</i>	(128, 87)	(126, 88)	(49, 1429)	(42, 1428)	0.49	0.02	
			<i>r</i>	(253, 82)	(254, 81)	(1105, 1443)	(1100, 1442)	0.49	0.01	
			<i>t</i>	(203, 45)	(205, 44)	(637, 1449)	(651, 1448)	0.61	-0.04	
			<i>b</i>	(210, 198)	(210, 203)	(1755, 1397)	(1757, 1392)	0.60	-0.01	
								0.55	0.00
61	62	(224, 94)	<i>l</i>	(137, 97)	(134, 98)	(2387, 1413)	(2383, 1410)	0.52	0.01	
			<i>r</i>	(253, 83)	(253, 82)	(1062, 1469)	(1050, 1469)	0.50	0.03	
			<i>t</i>	(204, 48)	(204, 46)	(443, 1450)	(449, 1448)	0.94	-0.02	
			<i>b</i>	(190, 172)	(188, 177)	(1957, 1415)	(1956, 1410)	0.93	0.00	
								0.72	0.01
62	63	(207, 112)	<i>l</i>	(134, 85)	(130, 84)	(135, 1422)	(133, 1418)	0.86	0.01	
			<i>r</i>	(233, 67)	(233, 65)	(800, 1448)	(793, 1446)	0.87	0.02	
			<i>t</i>	(206, 50)	(202, 47)	(594, 1438)	(571, 1435)	1.05	0.07	
			<i>b</i>	(163, 181)	(158, 184)	(2017, 1418)	(2028, 1413)	0.98	-0.03	
								0.94	0.01
63	64	(205, 114)	<i>l</i>	(119, 54)	(117, 50)	(233, 1395)	(240, 1391)	0.56	-0.02	
			<i>r</i>	(259, 117)	(260, 118)	(1221, 1446)	(1228, 1445)	0.57	-0.02	
			<i>t</i>	(206, 53)	(204, 50)	(606, 1439)	(594, 1436)	1.02	0.03	
			<i>b</i>	(168, 178)	(164, 181)	(2000, 1426)	(2010, 1421)	1.06	-0.03	
								0.81	-0.01
64	65	(210, 113)	<i>l</i>	(126, 57)	(119, 54)	(225, 1399)	(220, 1392)	0.91	0.01	
			<i>r</i>	(263, 116)	(265, 114)	(1222, 1447)	(1207, 1445)	0.92	0.04	
			<i>t</i>	(180, 32)	(178, 25)	(465, 1414)	(467, 1406)	1.20	-0.01	
			<i>b</i>	(174, 172)	(170, 175)	(2009, 1431)	(2019, 1426)	1.22	-0.03	
								1.06	0.01
65	66	(216, 108)	<i>l</i>	(145, 89)	(140, 85)	(100, 1427)	(112, 1421)	1.35	-0.04	
			<i>r</i>	(258, 114)	(260, 113)	(1254, 1458)	(1243, 1456)	1.32	0.03	
			<i>t</i>	(153, 43)	(149, 34)	(306, 1409)	(319, 1400)	1.37	-0.04	
			<i>b</i>	(175, 171)	(170, 175)	(2020, 1425)	(2030, 1419)	1.33	-0.03	
								1.34	-0.02
66	67	(184, 111)	<i>l</i>	(131, 67)	(126, 62)	(265, 1431)	(268, 1424)	1.80	-0.01	
			<i>r</i>	(253, 114)	(260, 113)	(1217, 1431)	(1210, 1424)	1.97	0.02	
			<i>t</i>	(178, 47)	(177, 39)	(564, 1436)	(563, 1428)	2.31	0.00	
			<i>b</i>	(182, 165)	(176, 170)	(1814, 1446)	(1851, 1440)	2.28	-0.11	
								2.09	-0.02
67	68	(216, 90)	<i>l</i>	(139, 70)	(132, 66)	(97, 1420)	(106, 1413)	1.50	-0.03	
			<i>r</i>	(249, 117)	(251, 118)	(1462, 1457)	(1458, 1455)	1.52	0.01	
			<i>t</i>	(150, 47)	(145, 43)	(221, 1421)	(223, 1415)	1.27	-0.01	
			<i>b</i>	(186, 160)	(181, 164)	(1955, 1424)	(1969, 1418)	1.22	-0.04	
								1.38	-0.02	
								net	11.01	-0.05

of Frame 56 to Frame 68 (ie, the image pairs processed are 56-58, 58-60, 60-62, 62-64, 64-66 and 66-68). Table 5.4 shows results for alternate frame selection and

Table 5.4: Results on alternate frames

$frame_i$	$frame_{i+2}$	cen		(x_i, y_i)	(x_f, y_f)	(ux_i, uy_i)	(ux_f, uy_f)	dist	rot
(174, 110)	l	(142, 135)	(141, 136)	(2147, 1459)	(2145, 1458)	1.10	0.00
			r	(264, 90)	(271, 87)	(1116, 1408)	(1111, 1400)	1.09	0.02
			t	(187, 36)	(188, 30)	(666, 1425)	(666, 1419)	1.33	0.00
			b	(172, 190)	(174, 197)	(1810, 1420)	(1800, 1413)	1.34	0.03
								1.21	0.01
(189, 102)	l	(145, 134)	(142, 136)	(2160, 1446)	(2161, 1442)	1.52	0.00
			r	(248, 132)	(253, 134)	(1380, 1434)	(1377, 1428)	1.51	0.01
			t	(202, 42)	(202, 39)	(682, 1439)	(678, 1436)	0.99	0.01
			b	(188, 181)	(187, 186)	(1805, 1421)	(1809, 1416)	1.01	-0.01
								1.26	0.00
(201, 102)	l	(137, 97)	(133, 101)	(30, 1436)	(6, 1432)	1.16	0.07
			r	(248, 128)	(250, 130)	(1393, 1446)	(1398, 1444)	1.20	-0.02
			t	(205, 48)	(202, 44)	(628, 1446)	(607, 1442)	1.64	0.06
			b	(191, 173)	(185, 179)	(1853, 1428)	(1878, 1421)	1.64	-0.07
								1.41	0.01
(208, 107)	l	(145, 96)	(137, 94)	(66, 1436)	(69, 1428)	2.38	-0.01
			r	(232, 69)	(233, 65)	(815, 1455)	(805, 1451)	2.39	0.03
			t	(193, 49)	(191, 43)	(503, 1440)	(501, 1434)	2.12	0.01
			b	(167, 177)	(157, 185)	(2002, 1419)	(2021, 1407)	2.13	-0.05
								2.25	-0.01
(207, 110)	l	(128, 62)	(117, 54)	(209, 1408)	(213, 1394)	1.85	-0.01
			r	(258, 113)	(262, 111)	(1222, 1449)	(1207, 1445)	1.86	0.04
			t	(184, 45)	(178, 33)	(470, 1431)	(462, 1418)	3.13	0.02
			b	(177, 170)	(167, 179)	(1977, 1433)	(2001, 1420)	3.16	-0.07
								2.50	0.00
(198, 101)	l	(140, 69)	(130, 62)	(193, 1434)	(199, 1422)	3.12	-0.02
			r	(251, 113)	(259, 114)	(1285, 1446)	(1280, 1438)	3.16	0.01
			t	(163, 53)	(156, 42)	(359, 1441)	(364, 1428)	4.03	-0.01
			b	(185, 159)	(176, 170)	(1884, 1441)	(1918, 1428)	4.02	-0.10
								3.58	-0.03
							net	12.22	-0.01

Table 5.5: Results on every third frame

$frame_i$	$frame_{i+3}$	cen		(x_i, y_i)	(x_f, y_f)	(ux_i, uy_i)	(ux_f, uy_f)	dist	rot
(178, 110)	l	(135, 66)	(136, 60)	(304, 1438)	(333, 1435)	1.25	-0.08
			r	(251, 132)	(257, 132)	(1312, 1424)	(1304, 1418)	1.23	0.02
			t	(202, 42)	(205, 35)	(730, 1428)	(732, 1420)	1.76	-0.01
			b	(188, 183)	(189, 191)	(1748, 1426)	(1748, 1418)	1.78	0.00
								1.51	-0.02
(204, 108)	l	(133, 88)	(126, 91)	(105, 1426)	(82, 1420)	1.37	0.07
			r	(253, 85)	(254, 80)	(1032, 1446)	(1005, 1443)	1.37	0.08
			t	(203, 48)	(200, 41)	(594, 1440)	(577, 1433)	2.35	0.05
			b	(158, 185)	(149, 199)	(2006, 1410)	(2008, 1394)	2.33	-0.01
								1.85	0.05
(199, 105)	l	(140, 69)	(125, 56)	(214, 1430)	(227, 1411)	4.22	-0.04
			r	(248, 117)	(258, 116)	(1284, 1450)	(1264, 1440)	4.27	0.06
			t	(162, 54)	(150, 36)	(364, 1436)	(367, 1415)	5.29	-0.01
			b	(187, 160)	(173, 173)	(1884, 1445)	(1941, 1428)	5.55	-0.16
								4.83	-0.04
(199, 104)	l	(140, 69)	(125, 56)	(205, 1431)	(220, 1412)	4.32	-0.04
			r	(248, 117)	(258, 116)	(1299, 1449)	(1277, 1440)	4.16	0.06
			t	(162, 54)	(150, 36)	(357, 1438)	(361, 1416)	5.53	-0.01
			b	(187, 160)	(173, 173)	(1881, 1443)	(1938, 1426)	5.20	-0.16
								4.80	-0.04
							net	12.99	-0.05

processing. Similarly every third frame is selected next and a smaller sequence is first created. From this newly constructed sequence, pairs of images are sent into the

proposed system and the distance and rotation estimates are shown in Table 5.5.

Figure 5.5: Processing alternate frames

Figure 5.6: Processing every 3rd frame

It has been shown that processing can be done on smaller sequences obtained by omitting frames, which can drastically reduce computations and yet provide a reasonable estimate of the camera movement. For example, by selecting alternate frames, the

computational effort is halved. Figures 5.5 and 5.6 show the linear distance estimates obtained by processing every alternate frame and every third frame compared to those generated by processing every consecutive frame. From Tables 5.4 and 5.5, the error in the net distance estimated is 11% if alternate frames are processed and 18% when every third frame is processed. These errors could be because of processing fewer frames, apart from the other potential error sources pointed in Section 4.6 in Chapter 4.

It has been thus shown that the overall method has been successfully applied to real images to automatically track both the rotation and linear distance of the camera. In the next section, an integrated framework for tracking both rotations and linear distances of the camera is presented.

5.3 Integrated framework for tracking linear motion and rotation

Figure 5.7 shows a framework for an integrated environment to detect both rotations and linear motion of the camera during endoscopy. This overall solution was achieved in an incremental fashion. First, the rotation engine that was proposed in Chapter 3 can be independently used at very high speeds to track pure rotations. This provides a flexibility in terms of area, speed and desired accuracy and the user can make an informed choice based on the resources available. Secondly, the combined approach for tracking both linear motion and rotation of camera proposed in Chapter 4, can be employed for a quick approximation using a small number of feature points. Automation of the process of unrolling has eliminated any dependencies on external inputs in Chapter 5, thus making it viable for deployment in real time systems. Thus, this overall integrated environment provides surgeon an option of using the

Figure 5.7: Overall research contribution

high throughput rotation engine for detection of pure rotations, or adopting the fast but crude feature point based approach of detecting both rotations and linear movements.

5.4 Summary

In this chapter, an improved solution for camera motion estimation is proposed by an automated process for image unrolling. This is achieved by estimating the center for unrolling instead of relying on manual inputs. This is based on a comparative study of the outputs of the left, right, top and bottom borders of the image. The overall method with automated unrolling through center estimation is then demonstrated on

a real endoscopic image sequence. Next, in order to achieve compute efficiency, the proposed method is applied on smaller sequences obtained by omitting frames. It is shown that this can drastically cut computations, while still providing a reasonable estimate of the camera motion.

It was shown in this chapter that for the proposed method to work, only a few feature points (4 for each image pair in the results above) are needed as against the highly compute intensive existing methods. As explained earlier in Chapter 4, this method is independent of the feature points selected because of unrolling and normalization and can be used in changing working conditions. Also, the method is now fully automated with no manual intervention. It can be used to provide a very quick and crude estimate of the camera movement.

Chapter 6

Conclusions and future work

6.1 Conclusions

In this thesis, we have devised new image processing techniques for the high-speed tracking of a camera during endoscopic procedures. The number of CORDIC operations have been notably reduced in the proposed high throughput image rotation engine. The effect of hierarchy on the computational efficiency has also been analyzed so as to identify the optimal hierarchy for a given image resolution. It was demonstrated that very high throughput computations can be realized with the optimal assignment of the hierarchy. In the case of 512×512 image, the hierarchy of 4 was shown to be optimal although the maximum possible hierarchy is 9. It was evident that the increase in the data width to accommodate the total number of additions incurs only marginal increase in the overall hardware cost. The proposed technique is highly scalable and lends well for high-speed rotation of high-resolution images. It has been demonstrated that this approach provides for substantial speed up with acceptable increase in area cost when compared with existing techniques.

The method proposed for detection of linear movements of camera requires only

a few feature points as compared to the existing methods, indicating its feasibility for use in real time systems. It was also shown that the overall output is independent of the feature points selected due to the proposed unrolling and normalization processes, thus making it highly suited for deployment in dynamic environments. The horizontal and vertical components of displacement values were elegantly utilized to detect rotations as well. This integrated method was successfully applied to a sequence of images captured in a simulated environment to estimate linear distance and rotation within a maximum error bound of 7%.

Further, the proposed algorithm for center estimation to automate unrolling was shown to converge within 10 iterations for the image sequences considered. Although this algorithm is not capable of detecting accurate center coordinates, it is very appropriate for the high-speed detection of the rotation and linear distances. The automated method continued to perform well even when it relied on a substantially reduced set of intermediate frames. Finally, the overall integrated environment presented provides surgeons the option of using the high throughput rotation engine for detection of pure rotations, or the integrated compute engine for the high-speed estimation of both the rotations and linear movements.

6.2 Scope for future work

- Only linear camera motion and rotation are addressed in this thesis as they form the most important degrees of motion of cameras through cylindrical passages. Robustness of the system can be increased by extending similar analysis to camera tilt and sideward movement. This would be a challenging problem as the horizontal and vertical displacement components will need to be represented using multiple degrees of motions.

- The effectiveness of the proposed feature point based method has been thoroughly evaluated for nearly perfect cylinders in order to manage the complexity of the problem. It would be interesting to extend it to irregular cylinders with the help of an appropriate dynamic unrolling function. For example, if the opening of the tract is elliptical instead of circular, then the ellipses should be mapped to straight lines in the process of unrolling. The normalization should also be adjusted accordingly. Also, the cases where the camera is not aligned with the axis of the cylinder can be looked into.
- More work on error analysis is required to ensure that the proposed methods can provide for a more dependable solution. This is a challenge as the overall error in motion estimation can be due to factors such as the imperfections in geometry of tract, inaccuracies in center estimation, and the number of frames selected. Although deriving separate error components resulting from each of these factors is a complex problem, the net error can be studied as a function of them. Effect of barrel distortion effects of wide angle lenses of the camera on the proposed method can be explored in detail.
- A detailed analysis of hardware complexity of the proposed image rotation engine, with regards to area and time has been done. It would be worthwhile to implement the high-speed rotation in hardware by exploiting the inherent parallelisms. Similarly, the various issues pertaining to the hardware portability of camera motion tracking could be investigated.
- In this thesis, study of movement of feature points in the unrolled domain was used for tracking camera motion during endoscopy. It would be interesting to extend this approach to aid other medical investigations. For example, the surgeon can benefit from an unrolled view of the entire endoscopic tract, as though it were slit lengthwise and then flattened out as a 2-dimensional

canvass to reveal its inner tissues. This may lead to the development of suitable techniques for real time unrolling, segmentation and stitching.

Bibliography

- [1] "What is endoscopy?", <http://www.arachnis.com/drsastry/endoscop.htm>
- [2] L.Demling, and H.J.Hagel, "Video endoscopy. Fundamentals and problems", *Endoscopy*, vol. 17, no. 5, pp. 167-9, 1985.
- [3] A.Chak, G.Isenberg, S.Mallery, J.V.Dam, G.S.Cooper, and M.V.Sivak Jr., "Prospective comparative evaluation of video US endoscope", *Gastrointest Endosc.*, vol. 49, no. 6, pp. 695-9, 1999.
- [4] "Imaginis - Endoscopy", <http://www.imaginis.com/endoscopy/>
- [5] Z. Miaomiao, "An intelligent decision support system by fusion for clinical endoscopic diagnosis", *PhD Thesis*, Nanyang Technological University, 2003.
- [6] "Programmable brake control system for use in a medical device", <http://www.freshpatents.com/Programmable-brake-control-system-for-use-in-a-medical-device-dt20060330ptan20060069310.php>
- [7] R.O.Buckingham, and A.C.Graham, "Computer controlled redundant endoscopy, *Proc. of IEEE International Conference on Systems, Man, and Cybernetics*, vol. 3, no. 14-17, pp. 1779-1783, 1996.
- [8] S.Dogramadzi, C.R.Allen, and G.D.Bell, "Computer controlled colonoscopy", *Proc. of IEEE Instrumentation and measurement technology conference*, vol. 1, pp. 210-213, 1998.

- [9] H.D.Hoeg, A.B.Slatkin, J.W.Burdick, and W.S.Grundfest, "Biomechanical modeling of the small intestine as required for the design and operation of a robotic endoscope", Proc. of IEEE International conference on Robotics and Automation, vol. 2, pp. 1599-1606, 2000.
- [10] "Wireless capsule endoscopy - the camera in a pill", <http://www.gihealth.com/html/test/given.html>
- [11] A.K.Hara, J.A.Leighton, V.K.Sharma, R.I.Heigh, and D.E.Fleischer, "Imaging of Small Bowel Disease: Comparison of Capsule Endoscopy, Standard Endoscopy, Barium Examination, and CT", *RadioGraphics*, vol. 25, pp. 697-711, 2005.
- [12] "Given (TM) Capsule Endoscopy - Instructions", <http://www.riverviewendo.com.au/capsuleinstruction.htm>
- [13] M.C.Lin, L.R.Dung and P.K.Weng, "An ultra-low-power image compressor for capsule endoscope", *Biomedical Engineering Online*, pp. 5-14, Feb. 2006.
- [14] G. Szkely and R.M.Satava, "Virtual reality in medicine", *British Medical Journal*, vol. 319, no. 1305, pp. 1-4, 1999.
- [15] "Virtual endoscopy", <http://spl.bwh.harvard.edu:8000/pages/projects/vizualization/virtual-endoscopy/index.html>
- [16] R.C.Gonzalez, and R.E.Woods, "Digital Image Processing", *Addison-Wesley Pub Co.*, 1992.
- [17] J.I.Hasegawa, K.Mori, J.I.Toriwaki, H.Anno, and K.Katada, "Automated extraction of lung cancer lesions from multiscale chest CT images by using three-dimensional image processing", *Systems & Computers in Japan*, vol. 25, no. 11, pp. 68-77, 1994.

- [18] W.M.Jeng, and D.J.Yang, "Portable PET image reconstruction system for parallel machines", *Proc. of IEEE Symposium on Computer-based Medical Systems*, pp.137-142, 1998.
- [19] Y.M.Kim, J.H.Kim, C.Basoglu, T.C.Winter, "Programmable ultra-sound imaging using multimedia technologies: a next-generation ultrasound machine", *IEEE Trans. Information Technologies in Biomedicine*, vol.11, pp. 19-29, 1997.
- [20] C.Tan, S.Krishnan, and C.Chan, "Endoscopic image processing with fuzzy reasoning for identification of intestinal lumen contour and abnormalities", *World Congress on Medical Physics and Biomedical Engineering*, pp. 616, 1994.
- [21] C.Kwoh, and D.F.Gillies, "Using fourier information for the detection of the lumen in endoscopic images", *Proc. of Frontiers of Computer Technology IEEE Region 10's Annual International Conference*, pp. 981-985, 1995.
- [22] S.M.Krishnan, K.V.Asari, C.J.Yap, and P.M.Y.Goh, "Application of computer vision techniques for intestinal lumen estimation", *Proc. of the Fifth International Conference on Control, Automation, Robotics and Vision*, 1998.
- [23] X.Yang, S.M.Krishnan, K.L.Chan, and P.M.Y.Goh, "Quantitative characterization of colon by fuzzy based image analysis", *Proc. of ISIRS 98*, pp. 53-55, 1998.
- [24] Z.Y.Xue, S.M.Krishnan, and X.Y.Lin, "Detection of oesophageal lumen contour and abnormality from endoscopic images", *Proc. of First Joint BMES-EMBS Conference*, pp. 13-16, 1999.
- [25] H.Tian, T.Srikanthan, and K.V.Asari, "A recursive Otsu-Iris filter technique for high-speed detection of lumen region from endoscopic images", *Applied Imagery Pattern Recognition Workshop*, pp. 182-186, 2001.

- [26] X.Yang, S.M.Krishnan, and K.L.Chan "Color image segmentation based on fuzzy rule-base reasoning applied to colonoscopic images", *Critical Reviews TM in Biomedical Engineering*, vol. 26, pp.346-347, 1998.
- [27] M.P.Tjoa, S.M.Krishnan, C.Kugean, and R.Doraiswami "Clinical endoscopic image segmentation based on homogeneity and hue", *Proc. of International Conference on Mathematical and Engineering Techniques in Medicine and Biological Sciences*, 2001.
- [28] S.A.Karkanis, G.D.Magoulas, D.K.Iakovidis, D.A.Karras, and D.E.Maroulis, "Evaluation of textural feature extraction schemes for neural network-based interpretation of regions in medical images", *Proc. of Intl. Conference on Image Processing*, vol. 1, pp. 281-284, 2001.
- [29] P.Wang, S.M.Krishnan, Z.Lin, N.Vikram, and Z.Xue, "Development of texture analysis method for medical endoscopic color iamges", *Proc. of IEEE-EMBS Asia-Pacific Conference on Biomedical Engineering*, 2000.
- [30] M.P.Tjoa, and S.M.Krishnan, "Texture-based quantitative characterization and analysis of colonoscopic images", *Proc. of Second Joint Meeting of IEEE Engineering in Medicine and Biology Society and the Biomedical Engineering Society*, 2002.
- [31] D.Koppel, Y.Wang, and H.Lee, "Viewing Enhancement in Video-Endoscopy", *Proceedings of the Sixth IEEE Workshop on Applications of Computer Vision* pp. 304, 2002.
- [32] K.Nakamura, "Development of real-time endoscopic image processing technology: Adaptive index of hemoglobin color enhancement processing", *Digestive Endoscopy*, vol. 14, pp. S40, July 2002.

- [33] T.C.S.Sean, and B.Schmidt, "Combining Message-passing and Inter-process communication for Efficient Parallel Medical Image Analysis", SPIE 04.
- [34] M.P.Tjoa, and S.M.Krishnan, "Feature extraction for the analysis of colon status from the endoscopic images", *BioMedical Engineering OnLine*, 2003.
- [35] D.E.Maroulis, D.K.Iakovidisa, S.A.Karkanisa, and D.A.Karras, "CoLD: a versatile detection system for colorectal lesions in endoscopy video-frames", *Computer Methods and Programs in Biomedicine*, vol. 70, no. 2, pp. 151-166, Feb. 2003.
- [36] K.F.Binmoeller, F.Thonke, and N.Soehendra, "Endoscopic Hemoclip Treatment for Gastrointestinal Bleeding", *Endoscopy*, vol. 25, pp. 167-170, 1993.
- [37] M.Shiba, K.Higuchi, K.Kadouchi, A.Montani, K.Yamamori, H.Okazaki, M.Taguchi, T.Wada, A.Itani, T.Watanabe, K.Tominaga, Y.Fujiwara, T.Hayashi, K.Tsumura, and T.Arakawa, "Risk factors for bleeding after endoscopic mucosal resection", *World J Gastroenterol*, vol. 11, no. 46, pp. 7335-7339, 2005.
- [38] S.K.Lu, S.S. Raman, P.Limanond, D.Aziz, J.Economou, R.Busuttil, and J.Sayre, "Influence of Large Peritumoral Vessels on Outcome of Radiofrequency Ablation of Liver Tumors", *Journal of Vascular and Interventional Radiology*, vol. 14, pp. 1267-1274, 2003.
- [39] J. Koo, and A. Kaffes, "Endoscopic resection of large colonic lipomas assisted by a prototype single-use Endoloop device", *Endoscopy*, vol. 38, no. 6, pp. 644-647, 2006.
- [40] S.Sou, T.Matsui, T.Yao, M.Yorioka, S.Tsuda, Y.Kikuchi, Y.Wada, T.Nagahama, K.Yao, "Clinical and endoscopic healing after infliximab treatment in patients with Crohn's disease", *Digestive Endoscopy*, vol. 18, pp. 2933, 2006.

- [41] D.S.K. Lu, and S.S. Raman, P.Limanond, D.Aziz, J.Economou, R.Busuttil and J.Sayre, "Influence of Large Peritumoral Vessels on Outcome of Radiofrequency Ablation of Liver Tumors", *Journal of Vascular and Interventional Radiology*, vol. 14, pp. 1267-1274, 2003.
- [42] M.E.Voutilainen and M.T.Juhola, "Evaluation of the diagnostic accuracy of gastroscopy to detect gastric tumours: Clinicopathological features and prognosis of patients with gastric cancer missed on endoscopy", *Eur J Gastroenterol Hepatol*, vol. 17, no. 12, pp.1345-1349, 2005.
- [43] Hiebert, "The 'Cured' Lung Cancer Patient: Is Follow-up by the Surgeon Worth-while?", *Annals of Thoracic Surgery*, vol. 60, pp. 1557-1558, 1995.
- [44] T.Thormaehlen, H.Broszio, and P.Meier, "Three-Dimensional Endoscopy", *Technical paper*, citeseer.ist.psu.edu/thormaehlen02threedimensional.html, 2002.
- [45] K.Mori, D.Deguchi, J.Sugiyama, Y.Suenaga, J.Toriwaki, C.R.Maurer Jr, H.Takabatake, and H.Natori, "Tracking of a bronchoscope using epipolar geometry analysis and intensity-based image registration of real and virtual endoscopic images", *Med Image Analysis*, vol. 6(3), pp. 321-36, 2002.
- [46] J.E.Volder, "The CORDIC Trigonometric Computing Technique", *IRE Trans. Elecron. Comput.*, vol. no. 8, pp. 330-334, 1959.
- [47] I.Ghosh, and B.Majumdar, "Design of an application specific VLSI chip for image rotation", *Seventh International Conference on VLSI Design*, pp. 275 - 278, 1994.
- [48] S.M.Bhandarkar, and H.Yu, "A VLSI Implementation of Real-Time Image Rotation", *Proc. Intl. Conf. Image Processing*, vol. 2, pp. 1015-1018, 1996.

- [49] D.D.Eggers, and E.Ackerman, "High Speed Image Rotation in Embedded Systems", *Science Direct - Computer Vision and Image Understanding*, vol. 61, pp. 270-277, 1995.
- [50] N.Takagi, T.Asada, and S.Yajima, "Redundant CORDIC Methods with a Constant Scaling Factor for Sine and Cosine Computation", *IEEE Transactions on Computers*, vol. 40, pp 989-995, 1991.
- [51] J.A.Lee and T.Lang, "Constant Factor Redundant CORDIC for Angle Calculation and Rotation", *IEEE Transactions on Computers*, vol. 41, pp.1016-1025, 1992.
- [52] S.Wang, V.Piuri, and E.E.Schwartzlander, Jr., "Granularly-pipelined CORDIC processors for sine and cosine generators", *IEEE Transactions on Computers* vol. 46, pp. 1202-1207, 1997.
- [53] I.Ghosh, and B.Majumdar, "VLSI Implementation of An Efficient ASIC Architecture for Real-Time Rotation of Digital Images", *International Journal of Pattern Recognition and Artificial Intelligence*, pp. 449-462, 1995.
- [54] J.W.Wallis, and T.R.Miller, "An optimal rotator for iterative reconstruction" *IEEE Transactions on Medical Imaging*, Vol. 16, pp. 118 - 123, 1997.
- [55] E.V.R.Di Bella, A.B.Barclay, R.L.Eisner, and R.W.Schafer, "A comparison of rotation-based methods for iterative reconstruction algorithms", *IEEE Trans on Nuclear Science* Vol. 43, pp. 3370 - 3376, Dec 1996.
- [56] Y.H.Yu, "The Quantization Effects of the CORDIC Algorithm", *IEEE Transactions on Signal Processing*, pp. 834-844, 1992.
- [57] "Avant! Passport 0.35 micron, 3.3 Volt SC Library CB35OS142", March 2000.

- [58] T.M.Lehmann, B.B.Wein, J.Dahmen, J.Bredno, F.Vogelsang, and M. Kohnen, "Content-based image retrieval in medical applications: a novel multistep approach", *Proceedings of SPIE: Storage and Retrieval for Media Databases 2000*, vol. 3972, pp. 312–320, 2000.
- [59] C.Harris, and M.Stephens, "A combined corner and edge detector", *Proceedings of 4th Alvey Vision Conference*, pp. 147-51; 1988.
- [60] J.Le Moigne, W.J.Campbell, and R.F.Crompt, "An automated parallel image registration technique based on the correlation of wavelet features", *IEEE Transactions on Geoscience and Remote Sensing*, vol. 40, no. 8, pp. 1849-1864, 2002.
- [61] "El Salvador Atlas of Gastrointestinal VideoEndoscopy", <http://www.gastrointestinalatlas.com/>

Appendix

Published Papers

NOVEL SCHEMES FOR HIGH-THROUGHPUT IMAGE ROTATION

S. Suchitra

S.K. Lam

T. Srikanthan

Centre for High Performance Embedded Systems
Nanyang Technological University, Singapore- 637553

Abstract—Real-time image rotation forms a core operation in many applications such as medical image processing, computer graphics and pattern recognition. Conventional rotation engines usually comprise of compute-intensive CORDIC operations that become a severe bottleneck in high-throughput applications. In this paper, we present a novel scheme that employs hierarchical computation strategies, exploits the symmetrical characteristics of the image, and performs successive pixel computations at high-speed to achieve high-throughput image rotation at low cost. We will show that the proposed method leads to a significant performance gain of about 20 times over an existing CORDIC-based rotation engine with similar hardware cost.

I. INTRODUCTION

Real-time image rotation is becoming an essential operation in many application areas such as image processing, computer graphics and pattern recognition. Conventional image rotation performs trigonometric operations on each pixel location. For example the final coordinates x' and y' of a pixel, which has been rotated by an angle of θ , can be computed as shown in (1) and (2). The original position of the pixel is (x_0, y_0) , and $\delta = +1/-1$ when θ is +ve /-ve.

$$x' = x_0 \cos \theta - \delta y_0 \sin \theta \quad (1)$$

$$y' = y_0 \cos \theta + \delta x_0 \sin \theta \quad (2)$$

CORDIC [1] is a hardware efficient algorithm that can be used to compute the trigonometric operations in (1) and (2). Although a CORDIC engine requires low hardware resources, it is iterative in nature, which can lead to performance degradation in CORDIC-intensive applications.

Till date, there is limited reported work on hardware implementations for image rotation. In [2], a parallel architecture for image rotation using CORDIC is presented, where a 512×512 image is first divided into an 8×8 window grid. The rotated positions of the 64 window centres are first pre-calculated using the CORDIC engine and stored in the initialisation stage (Fig 1). Subsequently the rotated offsets of the individual pixels within a window are computed using CORDIC. Each offset is added in parallel to the 64 rotated window centres using 'local-adders' to generate the rotated positions of 64 corresponding points.

Although this method speeds up the rotation process through parallelism, it has a high initialization latency due to the CORDIC computations on every window-centre. Apart from that, a total of 64×64 computation stages are required for image rotation. Each stage consists of a CORDIC operation, and 64 parallel additions. The bottleneck of this

Fig 1: Illustration of Rotation Method in [2]

approach arises from the CORDIC-intensive computations, which limits the throughput of the rotation.

II. PROPOSED METHOD

This section describes the proposed hardware efficient schemes to address the computational bottleneck of the approach in [2].

2.1 Employing a hierarchical approach to compute the rotated centres

This method aims to reduce the number of CORDIC computations in the initialisation stage by recursively partitioning the image into quadrants. The partitioning can be performed to any degree of hierarchy. For illustration and comparison purposes, we restrict the partitioning to the extent whereby the size of the smallest quadrant is

comparable to the window size used in [2]. To explain this method, we take the example of an 8×8 -window grid and its 3 hierarchical quadrant layers as shown in Fig 2. Each centre belongs to a particular quadrant in the 3 quadrant layers. For example, centre P has {A, B, C} as its set of identifying quadrants in Layers 1, 2 and 3 respectively. The rotated position of any centre P (with respect to the image centre) is equivalent to the sum of the rotated positions of the centres of the identifying quadrants with respect to the corresponding Layer centres. Thus, the 8×8 window centres can be obtained by computing only the rotated positions of the 4 centres of each layer, and subsequently generating the rotated positions of all the centres through various addition combinations of the 12 centres. This effectively reduces the number of CORDIC computations from 64 to 12.

Fig 2: Hierarchical Generation of Centres

2.2. Exploiting symmetrical characteristics of the image

We can further reduce the number of CORDIC computations in the initialisation stage by exploiting the fact that the coordinates of the window centres in the hierarchical layers are symmetrical about the coordinate axes. Hence, in hierarchical layers with the same number of horizontal and vertical pixels, we only need to perform the CORDIC computations on a single centre. The remaining 3 image centres can be inferred by at most a simple negation.

Fig 3: Symmetrical Inference of Rotated Centres

Fig 3 illustrates this notion of inferring the rotated window centres based on the symmetrical characteristics of the

image. For example, by computing (x', y') , the rotated position of (x, y) , we can easily infer the rotated positions of $(x, -y)$, $(-x, -y)$ and $(-x, y)$.

2.3 Employing successive pixel computations

The CORDIC computations for computing the offsets within a window can be replaced with additions by exploiting the order of the incoming pixel coordinates. It is sufficient to calculate the offset of just one pixel, and increment the result with a constant value (i.e. $\sin \theta$ or $\cos \theta$) to obtain the offsets for the adjacent pixels. This can be explained by analysing equations (1) and (2) for input values (x, y) , $(x+1, y)$ and $(x, y+1)$, where the neighbouring pixels differ by exactly 1 unit. This approach replaces the 64^2 CORDIC computations with equivalent number of additions. The flowchart shown in Fig 4 elucidates the process.

Fig 4: Flowchart for Generation of Offsets

III. SYSTEM OVERVIEW

The overall rotation process can be divided into 2 parts as discussed in [2] – the generation of rotated centres and generation of offsets. To compare with the method in [2], we fix the number of hierarchical layers as 3, which generates 64 centres. We also assume that the hierarchical layers have equal number of horizontal and vertical pixels. Fig 5 depicts the main computations involved.

The rotated position of a single window centre from each hierarchical layer is first computed using CORDIC. Subsequently, the rotated values of the remaining window centres in the hierarchical layers are inferred through the 'Symmetry-based Inference Unit'. As discussed earlier, this requires at most a simple negation operation. The outputs of this unit are added in different combinations to generate the 8x8 rotated-positions of the window grid centres. These values are stored in the ROM to be used in the next stage. To generate the pixel offsets, the sine and cosine value of the angle are obtained using the CORDIC engine. Then rotation is performed on the first pixel of the window using the CORDIC engine to generate X' and Y' . Subsequent offsets are obtained by incrementally adding the sine and cosine values to the previously computed X' and Y' as described in Fig 4. As these offsets are generated, they are simultaneously added to all centres to obtain the final rotated pixel-coordinates.

Fig 5: Block Diagram of Proposed Architecture

IV. DESIGN TRADEOFFS

The CORDIC algorithm inherently suffers from quantisation errors due to usage of finite width registers in the data path [4]. In the proposed method, this error is accumulated with every addition, especially during the offset calculations. The error can be reduced by increasing the data widths of the registers and adders, for the computations of rotated centres and window offsets. Exhaustive simulations were carried out with registers-adders of varying data widths. Fig 6 shows a plot of average error per pixel for all input angles (0° - 45°) for data widths

of 20 and 25. The average error per pixel is calculated by comparing the final rotated positions with that obtained by pixel-by-pixel CORDIC rotation. It can be observed that if 20-bit data widths are used, the error is significantly higher than [2]. Increasing this to 25 bits offsets this error.

Fig 5: Plot of Error for Varying Register Size

V. RESULTS AND COMPARISONS

5.1 Experimental Set-up

The image size is fixed as 512×512 , and a 12-iteration CORDIC engine is used. The data width of the adders-registers used in the CORDIC engine for the method in [2] and the proposed method are 20-bit and 25-bit respectively. The outputs of CORDIC in Pipeline Stage-A (Fig 1) are truncated to 10 bits prior to the additions in Pipeline Stage B. So, the data widths of the local-adders in both methods are 10-bits wide. An 8×8 window grid is employed, which gives rise to 64 window grid centres and window size of 64×64 . To obtain this configuration, a hierarchy of 3 was chosen for the proposed method. The latency estimates are obtained from the Passport 0.35 Micron standard-cell library [5].

5.2 Computational Complexity

The entire rotation process essentially comprises of additions and CORDIC functions and these will be analysed separately for both the methods. In [2], 64 centres are rotated during the initialisation phase and this accounts to 64 CORDIC operations. In the proposed method, 3 centres are rotated and subsequently 2 additions are performed for every rotated centre. Hence, a total of 3 CORDIC computations and 128 additions are performed in the initialisation stage. For the generation of the offsets, the

technique in [2] performs a rotation for each of the 64×64 pixels in the offset-window, accounting to $64^2 = 4096$ CORDIC operations. In the proposed method, 1 CORDIC computation is performed to rotate the first pixel of the offset-window and another CORDIC computation to obtain the sine and cosine values of the input angle. Subsequently, the remaining offsets ($64^2 - 1$) are obtained through additions. Once the offsets and the rotated centres are generated, the final phase is similar for both the methods, where each offset is added to all the rotated centres using local-adders to generate 64 rotated pixels simultaneously. The number of different computations involved in each stage are summarised in Table I and it can be seen that there are fewer CORDIC operations in the proposed method.

TABLE I SUMMARY OF COMPUTATIONS INVOLVED

	Method in [2]		Proposed Method	
	CORDIC	Addition	CORDIC	Addition
Initialization	64	-	3	64×2
Pipeline A	64^2	-	2	$64^2 - 1$
Pipeline B	-	64^3	-	64^3

5.3 Latency Estimate

We begin by estimating the latency of the CORDIC computations for the method in [2] and the proposed method. It is assumed that all additions are implemented using CLA adders and the latency estimates are in terms of Full Adder Latency (T_{FA}). The estimated latency for an n -bit CLA addition is $\log_2 n T_{FA}$. Hence, a 10-bit add-operation is estimated to take $3.32 T_{FA}$, 20-bit add about $4.32 T_{FA}$ and a 25-bit addition takes $4.64 T_{FA}$. A CORDIC engine comprises of addition and shift operations. Assuming that the shift operations are carried out through MUX-based barrel shifters, a maximum shift of 12-positions warrants 4 levels of MUX. So the latency attributed to the shift operation is $4 \times T_{MUX}$, where T_{MUX} is the latency of a 2-1 multiplexer. A 12-iteration CORDIC operation involves 12 additions and 12 shifts and delay due to use of registers. We assume that T_{F-F} is the latency of a register. We can also assume from [6] that $T_{F-F} \approx T_{FA}$ and $T_{MUX} \approx 0.65 T_{FA}$. Base on these values, the CORDIC computational latency for the method in [2] is $96 T_{FA}$ ($T_{CORDIC-[2]} = [12 \times (4.32 T_{FA} + T_{F-F} + 4 \times T_{MUX})]$) while that of the proposed method is $99 T_{FA}$ ($T_{CORDIC-NEW} = [12 \times (4.64 T_{FA} + T_{F-F} + 4 \times T_{MUX})]$).

Hence, the initialisation latency for the method in [2] is $I_{[2]} = 64 \times 96 = 6144 T_{FA}$ while that of the proposed method is $I_{new} = 3 \times 99 + (64 \times 2 \times 4.64) = 891 T_{FA}$. The initialisation stage of the proposed method is about 7 times faster now since the number of CORDIC operations is reduced.

Referring back to the pipeline shown in Fig 1, the overall latency for [2] is $I_{[2]} + (64^2 \times T_{CORDIC-[2]}) = 399360 T_{FA}$. For the proposed method, the overall latency is given by $I_{new} + ((64^2 - 1) \times 4.64) + (2 \times 99) = 19199 T_{FA}$.

The latency estimates take into the account the pipelined processing of the two methods. For method [2], the CORDIC computations in Pipeline Stage A are done in parallel with the additions in Pipeline Stage B (see Fig 1). The latency of CORDIC in Pipeline Stage A is more time-consuming than the latency of the additions in Pipeline Stage B. Hence, the throughput of the pipelined implementation in [2] is governed by the latency of CORDIC, and this is used for the pipeline latency estimates. Similarly, for the proposed method, the addition with 25 bit operands to generate offsets is done in parallel with the addition of offsets and rotated centers using 10-bit operands. Hence, the former latency is considered for the pipeline latency estimates.

Hence, we have shown that the proposed technique can achieve an overall speed up of about 20 times over the technique proposed in [2]. These values are summarised in Table II.

5.4 Area Estimate

The area of the CORDIC engine used in [2], which has three 20-bit adders and 20-bit registers (for each X, Y and Z datapath), a pair of 20-bit wide barrel shifters and a ROM table is estimated to be $(20 \times 3 \times (2A_{FA} + A_{F-F})) + (20 \times 2 \times 4 \times A_{MUX}) + (20 \times 12 A_{F-F})$, where A_{FA} , A_{F-F} and A_{MUX} correspond to the area of a 1-bit Full Adder, Flip Flop and 2-1 MUX respectively. A 1-bit CLA is estimated to occupy twice the area of a 1-bit serial adder. The ROM is assumed to be implemented as flip-flops and for a 12 iteration CORDIC, 12 arctan values are stored. This requires an area of $12 \times 20 A_{F-F}$. It can also be inferred from [6] that $A_{F-F} \approx 1.13 A_{FA}$ and $A_{MUX} \approx 0.47 A_{FA}$. Hence the CORDIC engine in [2] is estimated to occupy about $534 A_{FA}$. Similarly, the area of the CORDIC engine used in the proposed method is about $668 A_{FA}$.

The architecture in [2] primarily comprises of a CORDIC engine, and 64^2 pairs of local-adders. In the proposed method, there is a CORDIC engine, a pair of adder-units to generate the rotated centers and the offsets, and the 64^2 sets of local-adders. The memory requirements to store the rotated centres is estimated to be about $64 \times 10 \times A_{FF} = 723 A_{FA}$. All the local-adders in both methods are implemented as 10-bit CLAs. Assuming there are 64 pairs of them (one for each centre's x and y coordinates), the area is estimated to be $64 \times 10 \times 2 \times 2 A_{FA}$. The adder-unit for generating the

offsets in the proposed technique comprises of 25-bit CLAs (area = $100 A_{FA}$).

The area estimates are summarized in Table II. It can be seen that the areas of the two architectures are comparable with the proposed method showing a marginal increase of about 3%.

TABLE II AREA & LATENCY ESTIMATES

FA = Full Adder	Computational Latency (T_{FA})	Area (A_{FA})
Method in [2]	399,360	3817
Proposed Method	19,199	3951

VI. CONCLUSION

We have proposed novel schemes that rely on hierarchical computation strategies, exploiting the symmetrical characteristics of the image, and successive pixel computations to achieve high-throughput image rotation at low cost. When compared to a previously

reported rotation architecture, the proposed schemes require negligible increase in hardware cost and can lead to a significant performance gain of about 20 times.

REFERENCES

- [1] Volder, J.E., "The CORDIC Trigonometric Computing Technique", IRE Trans. Electron. Comput., Vol. EC-8, pp 330-334, 1959
- [2] Ghosh, I., and Majumdar, B., "VLSI Implementation of An Efficient ASIC Architecture for Real-Time Rotation of Digital Images", International Journal of Pattern Recognition and Artificial Intelligence, pp 449-462, 1995
- [3] Ghosh, I., and Majumdar, B., "Design of an application specific VLSI chip for image rotation", Seventh International Conference on VLSI Design, pp 275 – 278, 1994
- [4] Yu, Y. H., "The Quantization Effects of the CORDIC Algorithm", IEEE Transactions on Signal Processing, pp 834-844, 1992
- [5] Avant! Passport 0.35 micron, 3.3 Volt SC Library CB35OS142, March 2000

High-Throughput Image Rotation using Sign-Prediction based Redundant CORDIC Algorithm

S. Suchitra

S.K. Lam

T. Srikanthan

Centre for High Performance Embedded Systems
Nanyang Technological University, Singapore- 639798

ABSTRACT

Real-time image rotation forms a core operation in many applications such as medical image processing and computer vision. High-throughput computations for image rotation are a common requirement in real-time image processing. A VLSI design for image rotation that employs CORDIC was discussed in [2]. In this paper, we propose novel techniques to increase the throughput of the CORDIC computations, thereby notably improving the overall performance of the rotation unit with acceptable increase in VLSI area. An efficient sign-prediction method called BTSP (Binary-Tree based Sign Prediction) is proposed in our redundant CORDIC system to eliminate the sign detection process. Our investigations show that the BTSP based CORDIC algorithm for real-time rotation of a 512×512-pixel image, has a significant speed-up over existing redundant CORDIC methods with reduced hardware area.

I. INTRODUCTION

Real-time image rotation forms a core operation in many applications such as medical image processing and computer vision. Rotation of a gray level image involves performing trigonometric computations on each pixel. For example the x' and y' values of a pixel, which has been rotated by an angle of θ is shown in (1) and (2), where (x_0, y_0) is the original position of the pixel, and $\delta = +1/-1$ when θ is +ve/-ve (“+ve” refers to positive and “-ve” to negative).

$$x' = \cos\theta (x_0 - \delta y_0 \tan\theta) \quad (1)$$

$$y' = \cos\theta (y_0 + \delta x_0 \tan\theta) \quad (2)$$

CORDIC, developed by Volder [1] is a popular hardware-efficient algorithm that can be employed to compute the complex trigonometric functions in (1) and (2). It performs a series of micro-rotations on a vector lying on the X-Y plane over a desired input angle using simple add-shift operations. The CORDIC algorithm is based on the principle that any angle can be approximated as a summation of n micro angles of the form $\arctan(2^{-i})$

(i.e. $\theta \approx (\sum_{i=1}^n \pm \arctan(2^{-i}))$). Equations (1) and (2) are

rewritten as (3) and (4) by replacing the multiplication with simple shift operations by i positions.

$$x_{i+1} = k(x_i - \delta_i y_i 2^{-i}) \quad (3)$$

$$y_{i+1} = k(y_i + \delta_i x_i 2^{-i}) \quad (4)$$

The $\cos\theta$ terms in (1) and (2) accumulates to a constant k that converges towards 1.6468. This is known as the scaling factor and can be incorporated either at the start or end of the CORDIC iterations. To evaluate δ at each step, a third variable z is introduced to keep track of the angle pending for rotation as shown in (5). z is initialized with the input angle and the sign of z at any point indicates the direction of the next rotation.

$$z_{i+1} = z_i - \delta_i \arctan 2^{-i} \text{ where } \delta_i = \begin{cases} +1, & z_i \geq 0 \\ -1, & z_i < 0 \end{cases} \quad (5)$$

CORDIC is an approximation algorithm and the smaller the value of $|z_i|$ is in the final iteration, the higher the precision of computation.

Ghosh and Majumdar [2] proposed a parallel architecture for Image Rotation using CORDIC where a 512×512 image is first divided into an 8×8 window grid. As explained in [2], the rotated positions of the

Fig 1: Illustration of Rotation Method in [2]

centers of all windows are first pre-calculated and stored in the initialization stage (Fig 1). In Pipeline Stage A, the CORDIC engine is used to compute the offset of each pixel within a window. In the Pipeline Stage B, this offset is simultaneously added to the 64 centers using local adders to obtain the positions of the 64 corresponding points of all the windows. The CORDIC engine employed for a 512×512 image in [2] performs 12 iterations on 20-bit operands followed by a 10 bit truncation. The number of CORDIC iterations is limited by the size of the image and does not overly increase for larger images. In this paper, we propose a novel CORDIC architecture for such a rotation engine to realize high throughputs.

II. BTSP CORDIC - THE PROPOSED ARCHITECTURE

The problem with conventional CORDIC units that employ serial adders for x , y , z computation is that the carry propagation delay manifests as a major bottleneck to the performance. Hence, we propose to use BSD (Binary Signed Digit), the radix-2 Redundant Number System to achieve speedup for CORDIC algorithms by exploiting the limited carry propagation property of such number systems [5]. However, redundant CORDIC carries with it the problem of sign-detection [3][4]. It will be demonstrated in this paper that the complexity of sign detection can nullify the benefits of using Redundant Number Systems, especially for CORDIC with low number of iterations. So, in order to justify the use of BSD, we need to hide the latency of BSD sign detection. In this paper, we propose an elegant sign prediction method called BTSP (Binary Tree based Sign Prediction) that eliminates both the sign detection circuitry and the z datapath. Also, we employ a simple pre-scaling method for scaling factor compensation.

2.1 BTSP Sign Prediction Scheme

From (5), it can be seen that for each of the iterations, the direction of rotation, given by δ_i , can be ± 1 . So, for n micro-rotations, there are 2^n possible rotation combinations. Hence, an n -bit CORDIC engine would approximate any given input angle to one of the following 2^n 'target-angles': $\pm \tan^{-1} 2^{-1} \pm \tan^{-1} 2^{-2} \pm \tan^{-1} 2^{-3} \pm \dots \pm \tan^{-1} 2^{-n}$. This can be visualized using a binary tree structure as shown in Fig 2, where the nodes describe the angles that are obtained by following the rotation directions specified by the branches. The leaf nodes correspond to the target-angles.

As the angle rotated by CORDIC approaches the input angle, higher accuracy is achieved. This leads us to an interesting corollary: the knowledge of the index of the closest 'target angle' is sufficient to deduce the CORDIC rotation directions that need to be taken. For example, if

Fig 2: Binary Tree Representation of 'Target-Angles'

the input is $0.000\dots 01$ and the closest-leaf is Leaf #0, then the sequence to be taken is "000...0", where 0 signifies a -ve rotation and 1 signifies a +ve rotation.

The basic steps of the BTSP are summarized below:

1. Pre-compute and store all these leaves
2. Match the input angle to the closest leaf
3. Derive the index of the closest leaf and obtain the sequence of δ_i based on the binary tree topology.

For a 512×512 image, about $2^{12} = 4096$ values need to be stored and matched against a single input angle to determine the closest-leaf. Hence, for the BTSP algorithm to work efficiently, the problems of the large storage and complex angle matching must be tackled.

2.2 Tackling the Storage Problem

To reduce the storage of the leaf angles, we can store only the +ve leaves. For negative operands, we deduce the directions for the corresponding +ve operand and then modify the operation sign of CORDIC based on the fact that $\sin(-\theta) = -\sin\theta$ and $\cos(-\theta) = \cos\theta$. To further solve the problem of storage, we make use of Theorem 1 as stated below [7]:

Theorem 1: In N bit CORDIC, the first $(\lceil N/3 \rceil - 1)$ signed digits cannot be precomputed

It follows from Theorem 1 that for $i > \lceil N/3 \rceil - 1$, the rotation directions can be directly inferred from the bit weights since $\tan^{-1} 2^{-i} \approx 2^{-i}$. The angle pending for rotation after $\lceil N/3 \rceil - 1$ iterations (referred to as 'Remaining-Angle') in 2's complement format can be converted to a polar form with digit-set $\{1, -1\}$ to infer the remaining rotation directions. The recoding mechanism uses a simple logic, which is independent of the operand length, shown in [4]. +1 and -1 would imply rotation directions +ve and -ve respectively. Hence, sign-detection becomes an issue only for the first $(\lceil N/3 \rceil - 1)$ rotation directions. This means that for an N -iteration CORDIC, we need to store only $2^{\lceil N/3 \rceil - 1}$ values. Since we are storing only +ve leaves ($\delta_i = 1$), only 4 leaves need to be stored for a 512×512 image. A snapshot of the leaves is shown in Fig 3. The

Remaining-Angle can be obtained by performing a subtraction between the input and the closest-leaf.

Fig 3: Illustration for Closest Leaf Matching

2.3 Tackling the Matching Problem

The complexity to match the input angle to the closest leaf is now greatly reduced since we need to check the input angle against only 4 values. The matching process can be further simplified by the following analysis. It can be seen in Fig 3 that the first 2 bits display an incremental pattern. Hence checking just 2 bits of the input angle would eliminate all but 2 leaves as candidates for the 'Closest-Leaf'. As an illustration, consider Input Angle = 0.101110100111. The first 2 bits signify that the input angle lies between leaves A and B. Validating whether the input angle is closer to A or B would normally require 2 subtractions and 1 comparison. Alternatively, a less costly approach is used to pre-compute and store all the midpoints of two consecutive leaves. So, a single comparison with the relevant midpoint would lead us to the closest-leaf. The overall-architecture for a 12-iteration BTSP CORDIC engine is shown in Fig 4. Note that the computation of the Remaining-Angle is done in parallel to the first 3 iterations. Due to the parallel inference of δ from the recoded Remaining-Angle, an extra rotation has to be performed for maintaining the accuracy [4].

The following describes an example of the 12-iteration CORDIC using BTSP. The midpoints of the leaves shown in Fig 3 are:

Mid Pt # 0	0.001101111111	0.219
Mid Pt # 1	0.011101110101	0.464
Mid Pt # 2	0.101101010110	0.709

Given an input angle (θ): 0.100110011001(0.6 rad),

1. Fix the first direction to be +ve; $\delta_1 = +1$.
2. Examine the first 2 bits to extract Mid Pt # 2
3. Perform comparison between Mid Pt # 2 and θ
4. Since θ is lesser than Mid Pt # 2, Leaf A (Leaf # 2) is the Closest-Leaf. This implies $\delta_{2-3} = +1, -1$
5. While Iterations 1-3 are being computed, subtract Closest-Leaf from θ to obtain the Remaining-Angle (0.000001000000)
6. Perform 2's Complement to Polar Conversion of bits 4 – 12 to obtain $\delta_{4-13} = +1, -1, -1, +1, -1, -1, -1, -1, -1$

Fig 4: BTSP-based CORDIC Architecture

2.4. Scaling Factor Compensation

To compensate for k in (3) and (4), a serial pre-scaling model described in [6] is employed. During the initialization process of the rotation engine, the X and Y input registers of the CORDIC engine are first loaded with the pre-scaled coordinates of the first center C_0/k . The subsequent centers are computed by continuously incrementing the coordinates by a constant value. So, for a 512×512 image, X or Y is incremented by $64/k$ to obtain the next pre-scaled center. During the calculation of the offsets for a single window in Pipeline Stage A (Fig 1), an incremental of each pixel coordinate by $1/k$ is performed. These incremented values are passed to the X, Y input registers of the CORDIC engine.

III. RESULTS & COMPARISONS

The proposed BTSP CORDIC engine is compared with Conventional 2's complement CORDIC, and those that use BSD sign-detection methods proposed by Takagi et al [3] viz. Double Rotation and Correction Rotation. Table 1 and Table 2 summarize the latency and overall-area estimates in Full Adder Latency (T_{FA}) and Full-Adder Area (FA) respectively. The estimates are standardized according to the Passport 0.35 Micron Library [8]. As specified in [2], a 12-iteration, 20-bit CORDIC is used for a 512×512 image. So, the X and Y datapaths are 20 digits long. The Z datapath is assumed to be 12 digits long. The BTSP Method performs an extra 13th rotation as mentioned earlier. The overall BTSP CORDIC architecture comprises of a pair of 20-digit redundant adders, registers and shifters, a 12-bit CLA for computing the "Remaining Angle", a CLA-based 12-bit comparator for deducing the closet-leaf, ROM storing seven 12-bit values and an angle recoder. Takagi's CORDIC require an additional sign detection circuit. For all the BSD methods, redundant adders are used, and a 1-Bit Redundant Adder (RA) is estimated to be 2FA. Latency of an RA is independent of operand length and is

estimated to be $2T_{FA}$. From (8), a register is 1.13 times the size of an adder.

LATENCY (In T_{FA})				
	Conv	Double	Corr (m=4)	BTSP
X, Y (or Z)	$\log_2 20 \times 12$	4×12	$[(0.25 \times 4) + (0.75 \times 2)] \times 12$	2×13
Shifting	$12 \times 4 \times 0.64$	$12 \times 4 \times 0.64$	$12 \times 4 \times 0.64$	$13 \times 4 \times 0.64$
Sign-Detect	-	$\log_2 3 \times 12$	$\log_2 6 \times 12$	-
Comparison	-	-	-	$\log_2 12 = 3.58$
Rem-Angle	-	-	-	3.58(hidden)
Recoding	-	-	-	0.17(hidden)
Total	82.58	97.68	91.68	62.86

Table 1: Comparison of Latency Estimates

AREA (in FA)				
	Conv	Double	Corr(m=4)	BTSP
X,Y	$2 \times 20 \times 2.13$	$2 \times 2 \times 20 \times 2.13$	$2 \times 2 \times 20 \times 2.13$	$2 \times 2 \times 20 \times 2.13$
Z	12×2.13	$2 \times 12 \times 2.13$	$2 \times 12 \times 2.13$	-
Shifter	$20 \times 4 \times 0.47$	$40 \times 4 \times 0.47$	$40 \times 4 \times 0.47$	$40 \times 4 \times 0.47$
ROM	$12 \times 12 \times 1.13$	$12 \times 12 \times 1.13$	$12 \times 12 \times 1.13$	$7 \times 12 \times 1.13$
Sign	-	6	12	-
Comp	-	-	-	2×12
Recoder	-	-	-	12×0.13
Adder	-	-	-	2×12
Total	273.48	459.44	465.44	390.08

Table 2: Comparison of Area Estimates

Shifting is carried out using a barrel shifter, implemented as a tree of multiplexers. In this case, the maximum shift of 12-bits warrants 4 levels of MUX. So the latency attributed to the shift operation is $4 \times 0.64 = 2.57 T_{FA}$. Sign Detection in case of Takagi is carried out using a CLA-based circuit. The area estimate for a CLA is about $2n$ FA and latency is $\log_2 n T_{FA}$, where n = length of operand. In case of Double Rotation, 3 digits are checked and 2 redundant additions are performed at every stage. In the case of Correction Rotation (for $m=4$), 3, 4, 5 and 6 digits need to be checked in consecutive cycles and every fourth rotation is a double-rotation. Hence a 6-digit sign-detection circuit is employed. A similar CLA-based circuit is employed for the comparison circuit in BTSP. The latency for a 12-bit comparison is about T_{FA} . The 'Remaining-Angle' is computed using a CLA. The angle recoding can be realized using a simple circuit consisting of inverters. The combined latency of CLA addition and recoding is about $3.58 + 0.17 = 3.75 T_{FA}$, which can be hidden completely during the 1st 3 iterations. As for ROM requirements (estimated as DFFs), it is assumed that 12 arctan values (of length 12-bits each) are stored for both the sign-detection methods. For the BTSP Method, 4 leaves and 3 midpoints are stored. ROM access time is ignored in the calculations.

It can be seen from Table 1 that the latency of BSD sign detection-based CORDIC is worse than that of the

conventional 2's complement CORDIC. The proposed CORDIC using BTSP has 31%, 55%, 46% latency gains when compared to Conventional, Double Rotation and Correction Rotation methods respectively. The proposed method has more area than conventional CORDIC, due to the usage of redundant numbers, but has lesser area than the other BSD-based sign detection methods.

IV. CONCLUSIONS

In this paper, we have proposed novel techniques to accelerate BSD CORDIC computation, which forms the most compute intensive core of an image rotation unit. The technique to eliminate sign detection and z data path employs only a simple lookup table. Our investigations reveal that the gain in terms of computation time with respect to conventional 2's complement and redundant number correction rotation method [3] is about 31% and 46% respectively. Moreover, the incorporation of a simple scaling-factor compensation method have contributed to the realization of a high-performance CORDIC unit that is suitable for realizing real-time image rotations.

V. REFERENCES

[1] Volder, J.E., "The CORDIC Trigonometric Computing Technique", IRE Trans. Electron. Comput., Vol. EC-8, pp 330-334, 1959

[2] Ghosh, I., and Majumdar, B., *Design of an application specific VLSI chip for image rotation*, Seventh International Conference on VLSI Design, pp 275 – 278, 1994

[3] Takagi, N., Asada, T., and Yajima, S., *Redundant CORDIC Methods with a Constant Scaling Factor for Sine and Cosine Computation*, IEEE Transactions on Computers, Vol. 40, pp 989-995, 1991

[4] Timmermann, D., Hann, H., and Hosticka, B.J., *Low Latency Time CORDIC Algorithms*, IEEE Transactions on Computers, Vol. 41, pp 1010-1015, 1992

[5] Parhami, B., *Carry-Free Addition of Recoded Binary Signed-Digit Numbers*, IEEE Transactions on Computers, Vol 37, pp 1470-1476, 1988

[6] Feng, Z., and Peter K., *A High Speed Hough Transform Using CORDIC*, Proc. Int. Conf. On Digital Signal Processing, Cyprus, 1995.

[7] Srikanthan, T., and Gisuthan, B., *A novel technique for eliminating iterative based computation of polarity of micro-rotations in CORDIC based sine-cosine generators*, Microprocessors and Microsystems 26, pp 243-252, 2002.

[8] Avant! Passport 0.35 micron, 3.3 Volt SC Library CB35OS142, March 2000.

ARRAY BASED ARCHITECTURE FOR EZW IMAGE ENCODING ON FPGA USING HANDEL-C

S. Suchitra

C.S.Lim

T. Srikanthan

Centre for High Performance Embedded Systems
Nanyang Technological University, Singapore- 637553

Abstract—Real-time image compression is usually a core operation in many embedded applications. Often in such applications, high throughputs are required. In this paper, we propose an array-based architecture for hardware implementation of an EZW encoder to achieve high throughputs. It exploits the inherent properties of parallelism exhibited by the EZW algorithm and also the spatial correlation properties of the DWT coefficients. Every spatial tree is generated and stored independently in memory blocks of an array and each of them subject to EZW separately through an array of Processing Elements. This architecture is implemented on FPGA using Handel C and the resource utilization is presented.

I. INTRODUCTION

The Embedded Zerotree Wavelet algorithm (EZW) proposed by Jerome Shapiro [1], a popular algorithm for entropy encoding of wavelet coefficients, has been widely used for image compression applications. The concept of a zerotree is based on the hypothesis that if a wavelet coefficient at a high scale is insignificant with respect to a given threshold T , then all wavelet coefficients of the same spatial location at lower scales are likely to be insignificant with respect to T . The original algorithm accepts as its input the wavelet coefficients of an image or frame of a video, and arranges the coefficients in a tree structure. The coefficient tree structure maintains the ancestor-descendant relationships amongst the wavelet coefficients in the same orientation but in different levels of decomposition. Based on a threshold, the value of which decreases in each scan of the tree, the EZW algorithm searches the coefficient tree structure for significant coefficients whose magnitudes are greater than or equal to the current threshold value. Most insignificant coefficients in higher scales represent roots of zerotrees, which forms the basis for compression, since each node in the tree need not be encoded separately as being insignificant.

Developing efficient hardware for EZW based compression has been attempted previously [2][3]. Time efficient encoding is a requirement for many high throughput image-processing applications. The main concern with developing high throughput encoders is the large number of input coefficients that need to be processed.

Hardware schemes like parallelism and pipelining can be employed reduce the overall processing time. A parallel architecture making use of depth first scan order is described in [4]. Storing all coefficients in a common memory and then having parallel processing units accessing these could result in the problem of memory bus contention.

This paper describes parallel generation of wavelet coefficients making use of the inherent property that the pixels contributing to the generation of each coefficient tree can be isolated from the others. A memory array structure is employed to separately store all the trees generated. It is further illustrated how this set up is conducive for the parallel execution of EZW algorithm on each block of the memory array. Descriptions of the processing elements employed to perform the EZW and the operations of the various flags in the working registers are also presented. The design is implemented using Handel C and the resource utilization on FPGA is reported.

II. INTRODUCTION

2.1 Memory Array for Parallel Generation of Coefficients

The concept of spatial trees in wavelets, owing to the spatial correlation across different frequency bands has an interesting implication in terms of the 2-D sub-band decomposition for the generation of DWT coefficients.

Fig 1: 8x8 Block of Image gives birth to entire spatial tree

The input image can be broken down into blocks that are self-sufficient for generating a spatial coefficient tree. Hence the process can be parallelized. This is illustrated in Fig 1 where the image is broken down into blocks of 8x8 and each of these can independently generate the wavelet trees. A scale-3 Haar Wavelet is used to demonstrate the concepts presented throughout the paper. If we know beforehand the number of levels of decomposition we want to achieve, then we can infer the block size required to produce a single tree. If we consider an image of size 128x128 for example, we can divide it into 256 blocks of size 8x8 and use each of these blocks to generate the DWT coefficients constituting elements of a level-3 Haar spatial tree. Also, each of these 256 image blocks can be processed independently and the resulting trees can be stored separately as blocks of a memory array. Having this kind of memory structure aids in simultaneous storage of the coefficient values, without the problem of memory bus contention.

Apart from facilitating parallel generation and storage of coefficients, employing such a memory structure has benefits for EZW encoding. This is because the identification of zerotrees needs the entire wavelet tree as input, which is present in entirety in each block of the memory array. This is elaborated in subsequent sections. Fig 2 shows the memory array structure used for storing these trees for a 128x128 image. The number of memory blocks used corresponds to the number of trees, which in turn is the same as the number of elements in the lowest frequency band.

Fig 2: Memory Array for storing spatial DWT trees

2.2 Scan Order within every block of Memory Array

Once the coefficients have been generated, each memory block is subjected to the EZW Encoding Algorithm in parallel. The first step is to employ a suitable scanning order to read in the values. Scanning within these trees is done following the conventional Morton scan order. For the example considered, each tree consists of 64 elements irrespective of the size of the entire image and the scan

Fig 3: Scan Order employed within each memory block

order is shown in Fig 3. The parent-children relationship that follows from such a scan order is shown in Table 1. The main root node has only 3 children and the leaves have no children. Other than that, all elements have 4 children each. As can be seen from Table 1, the indices of children are incrementally filled starting from index 1 in a horizontal fashion till the count 63 is reached (last child of node #15). After that, all subsequent nodes have no children, as they constitute the leaves.

Index of Node	Index of Child # 1	Index of Child # 2	Index of Child # 3	Index of Child # 4
0	1	2	3	-
1	4	5	6	7
2	8	9	10	11
3	12	13	14	15
⋮	⋮	⋮	⋮	⋮
15	60	61	62	63
⋮	⋮	⋮	⋮	⋮
62	-	-	-	-
63	-	-	-	-

Table 1: Parent-Children Relationship Table

The implication of employing such a scan order is the easy generation of the indices of children nodes. This is because for all nodes that have 4 children (#1 to #15), the indices of the children can be obtained by simply multiplying the index of the node by 4. In hardware, this translates to performing 2 simple shift operations.

2.3 Processing Elements (PE) Description

Encoding is done by an array of Processing Elements (PE) with every tree being processed by an independent Processing Element (PE). Each PE consists of 64 4-bit working registers (Reg₀ to Reg₆₃), maintaining bookkeeping information about one coefficient each. Each of these PE's access information from a memory block and do not pose any data-dependency problems. Hence, they can run in

parallel. The structure of the register is depicted in Table 2. The EZW algorithm is implemented on all the PE's simultaneously and the resulting bit streams are eventually reordered using a scheduler to obtain the final embedded stream. Bit-wise scanning of the memory blocks is done in the order as shown in Fig 3 and the registers are filled in ascending order (Reg₀ first, then Reg₁ and so on). The coefficients are assumed to be stored in the sign-magnitude format. Dominant pass is performed n number of times, separated by subordinate passes where n refers to the maximum number of bits used to represent the coefficients. The initial threshold for checking whether a coefficient is significant or not is set to 2^{n-1} . This value is halved for every new dominant pass.

	(Bit 0) SGN	(Bit 1) MAG	(Bit 2) PDG	(Bit 3) SIG
1	Positive Coefficient	Bit Weight of current position: 1	Pending Processing	Already found Significant
0	Negative Coefficient	Bit Weight of current position: 0	Already Processed	Not found significant yet

Table 2: Structure of the Working Registers

PDG: At the start of every pass, PDG is set to 1 for all 64 registers because all coefficients are pending processing for that pass. Once the encoding outcome of a coefficient is decided (p , n , z or t is output) PDG is cleared. Also, for cases when zerotrees are detected, all the descendants' PDG flags are cleared. This drastically cuts down on processing since it saves on the checking process later on the descendants, as they are already detected to be a part of the zerotree.

SGN: This is set once according to the sign of the coefficient being considered during the initialization process and is not changed for the entire encoding process.

MAG & SIG: MAG is initialized with the most significant bit (MSB) of the corresponding coefficient at the start of the encoding process. Thereafter, prior to every dominant pass, it is reloaded with the value of the next most significant bit. During the dominant passes, this bit is checked. Since the threshold is set as powers of 2, if MAG is a 1, then p or n symbol is output (p if SGN is 1 and n if SGN is 0). Also, the SIG flag is set to mark the detection of a significant coefficient (and remains a 1 for all the subsequent passes) and MAG is immediately replaced by the next significant bit value of that coefficient. This is to facilitate the subordinate pass, where the value of MAG is output if SIG is set to 1. Note that there is no reordering of coefficients done for the subordinate pass as originally proposed in [1]. This is done so that the hardware is not unnecessarily complicated.

If in the dominant pass, MAG is 0, then a check for zerotree is performed. Depending on whether a zerotree is detected or not, a t or z is output respectively.

Zerotree Detection: Tree traversal is done following the addresses of child nodes that are generated by shift operations as explained in section 2.2. According to the original algorithm, nodes that are detected to be significant are seen to be zero for subsequent passes and contribute to zerotrees. To incorporate this, in our zerotree detection process, we first check the SIG flag of the child nodes; if this is 1, then there is no need to check for the MAG flag. If however, the SIG is 0, only then we proceed to check for MAG flag. If this is also zero, then the process is continued till all descendants are covered. If a zerotree tree is detected, then a similar traversal is performed along its descendants and all PDG flags of the zero-tree are cleared.

III. HARDWARE IMPLEMENTATION

The design of EZW Encoder described in Section 2 was implemented with Handel C [5] and ported onto Xilinx Vertex 2 XC2V1000, FPGA. A memory structure of 256 memory arrays each with depth size of 64 is instantiated. This configuration of memory structure was based on a 3-level wavelet decomposition of a 128x128 image. Each of the memory arrays in the structure stores a coefficient tree from the top level to the bottom level as described in Section 2. The main reason for memory arrays would be in the facilitation for parallel processing. Each of the PE's for a single tree will be accessing the coefficient values of their respective trees simultaneously; therefore such a memory array structure would avoid bus contention problems.

The hardware implementation of the PE's essentially comprises of comparators, bit-shifters and working registers. Comparators are for performing threshold operations and for checking all the flags in the working registers as described in Section 2. Bit shifters help in tracking addresses of the child nodes in the memory array for all operations involving tree traversal. Every PE requires a 4-bit working register structure as described in Section 2 for the maintaining the various flags necessary for the dominant and subordinate passes. Fig 4 below shows the block diagram of the implemented hardware architecture and Table 3 reports the resource utilization of a single PE of the implemented hardware on FPGA.

Number of Slice Flip Flops	454 out of 10, 240	4%
Number of 4 input LUTs	1,929 out of 10, 240	18%

Table 3: Resource Utilization on FPGA for a single PE

Fig 4: Block Diagram of the implemented Hardware

Fig 5 shows the overall flow of the encoding process within a PE.

Fig 5: Overall Process Flow

IV. CONCLUSION

In this paper, we have demonstrated that certain properties exhibited by wavelet coefficients can be exploited to have a parallel architecture for high throughput image encoding using the famous EZW algorithm. Each spatial tree can be generated separately, stored and processed and the entire architecture can be visualized as an array. The design of the memory array and that of the underlying processing elements has been explained. The design of EZW Encoder described in Section 2 was implemented with Handel C and ported onto FPGA and the resource utilization has been reported.

REFERENCES

[1] Jerome M. Shapiro, "Embedded Image Coding Using Zerotrees of Wavelet Coefficients," IEEE Transactions on Signal Processing, vol. 41, no. 12, pp. 3445-3462, Dec. 1993

[2] Omaki, R.Y., Fujita, G., Onoye, T., Shirakawa, I., "Embedded zerotree wavelet based algorithm for video compression", TENCON 99. Proceedings of the IEEE Region 10 Conference , vol. 2 , pp. 1343 – 1346, Sept. 1999

[3] Ang L., Cheung, H. N., and Eshraghian, K., "VLSI decoder architecture for embedded zerotree wavelet algorithm", Proceedings 1999 IEEE International Symposium on Circuits and Systems, pp. 141-144, May 1999

[4] Hon Nin Cheung; Li-Minn Ang; Eshraghian, K., "Parallel architecture for the implementation of the embedded zerotree wavelet algorithm", ACAC 2000 , 5th Australasian Computer Architecture Conference, pp. 3-8, Feb.2000

[5] Handel-C Language Reference Manual, 3.1, Celoxica Ltd, Didcot, UK, 2001 (<http://www.celoxica.com/home.htm>)

REAL TIME TRACKING OF CAMERA MOTION THROUGH CYLINDRICAL PASSAGES

Suchitra Sathyanarayana, Srikanthan Thambipillai and *Chris. T. Clarke

Center for High Performance Embedded Systems
Nanyang Technological University, Singapore

*Department of Electronic and Electrical Engineering
University of Bath, United Kingdom

ABSTRACT

In this paper, we present a novel method for tracking the motion of a camera through cylindrical passages. The proposed technique is based on studying movement of feature points in the image stream captured during traversal of the camera. It applies various image-processing algorithms for analysis in an 'unrolled' domain to provide a quick estimate of camera motion. The proposed techniques have been evaluated using MATLAB on a sequence of test images and it has been shown that both the linear distance and rotation can be inferred using a small number of feature points, thereby justifying its use in real time systems.

Index Terms— Camera motion tracking, Image unrolling

1. INTRODUCTION

There have been many attempts in the past to track camera motion. In [1], camera motion is recovered from image sequences based on registration of silhouette cones. In [2], a method is proposed for estimating large camera motions for omni directional vision. There has been little work done, however, on estimating camera motion through cylinders. This holds a special significance, especially in applications like medical endoscopy, where the endoscopic camera traverses through cylinder-like tracts like the esophagus, bronchus etc. Existing methods that track camera motion in endoscopy are compute intensive, mainly suitable for computer graphics applications [3][4]. These are not suitable for direct porting into real time systems.

In an image sequence captured with a small time interval, consecutive frames show transformations like scaling, skew, rotation etc. The nature and degree of these transformations gives an indication of the camera motion during that elapsed time. In this paper, in order to detect camera motion, selected portions of the image containing unique properties, called feature points (or FP) are used. More specifically, the *coordinate positions* of feature points are tracked in consecutive image frames. Analysis of the displacements of FP positions is done for estimating the linear motion and rotation of the camera.

To manage the complexity of the problem, this paper tracks the linear motion of the camera along the cylindrical axis and the rotation about the axis, since these are the two degrees of motion most likely to occur in cylindrical passages.

2. PROPOSED CAMERA MOTION DETECTION

2.1. Linear motion tracking

The walls of the cylinder are projected as circular rings on the 2-D image and this gives a template for analyzing the feature points. It is seen that if the camera moves downward, into the cylinder in a linear fashion, then the feature points move radially outwards. That is, they are projected on rings of bigger radii. Similarly, when the camera moves upward linearly, the positions of feature points shift toward the center of the rings.

The magnitude of displacement of feature points when the camera is moved upward/downward along the main axis, is dependent on the relative distances of these points with respect to the camera. The points that lie physically closer to the camera show a greater apparent displacement when compared to points further away, for the same magnitude of movement of the camera. Points that are closer to the camera lie on circles with a larger radius compared to points that are further down the cylinder.

To verify that points at different depths show different apparent movements, we use a sheet of paper with markings that simulate feature points (Fig. 1 (a)). This is rolled into a cylinder and the snapshot of it is captured using the camera placed above it (Fig. 1 (b)). A pure downward movement of the camera along the axis is simulated, without causing any other movement to the camera or cylinder. The cylinder is recaptured, as shown in Fig. 1 (c).

Using MATLAB, we track the movement of $P = (132, 97)$, $Q = (294, 158)$ in Fig. 1 (b) to $P = (72, 31)$, $Q = (282, 130)$ in Fig. 1 (c). The displacement of P was measured to be 89.2 units, almost three times the value as that of Q (30.5 units). This shows that the apparent movement of P, which is closer to the camera than Q, is greater.

Fig. 1. Inconsistency in displacement magnitude

This experiment shows that FPs situated at different depths undergo different apparent displacements, for the same movement in camera. To standardize the readings that each FP-pair gives us, we perform an operation called *image unrolling* followed by a *normalization*. The process of unrolling sorts points based on depth information and creates a depth profile. Normalization is applied on this depth profile to even out differences that occur due to varying depths.

To illustrate better, Fig. 2 (a) shows a sheet of paper prior to rolling into a cylinder. The aerial view of this cylinder, as captured by the camera is shown in Fig. 2 (b). The unrolling process is equivalent to flattening it into a 2-D surface (Fig. 2 (c)). Figure 2 (d) shows the normalized unrolled image, which closely resembles the original surface in Figure 2 (a). Unrolling and normalization are explained below in terms of the underlying computations and how they help in achieving motion tracking of the camera.

Fig. 2. Image Unrolling Concept

2.1.1. Computations involved in image unrolling

Images captured by the camera along the axis of the cylinder reveal concentric circles (Fig. 2 (b)). Mathematically, unrolling refers to the process of mapping these circles of varying radii to straight lines of equal lengths, and then stacking these straight lines together to provide a flattened view of the cylinder. The length of the unrolled rectangle spans the angular range of 0° to 360° corresponding to the mapped circles. The breadth of the rectangle is proportional to the number of circles unrolled and stacked.

Through this process, the points in the cylinder are now arranged based on their depth in the unrolled image. In other words, each row in the unrolled domain corresponds to a ring of tissue at a particular depth in the passage. The location of a FP in the unrolled image gives information about its depth in the cylinder.

Consider a point P with coordinates (x, y) lying on a circle with radius r and origin (x_0, y_0) , as shown in Fig. 3.

Fig. 3. Unrolling transformation

If P makes an angle θ with the horizontal axis, then:

$$r = \sqrt{(x - x_0)^2 + (y - y_0)^2} \quad \theta = \tan^{-1} \left(\frac{y - y_0}{x - x_0} \right) \quad (1)$$

Let the dimensions of the unrolled canvas be $L \times B$. Point P would map to a point P' with coordinates (x', y') after the unrolling transformation. Each line in the transformed domain has a length L and corresponds to 2π radians. This leads us to (2):

$$x' = \frac{L\theta}{2\pi} \quad y' = B - r \quad (2)$$

Expressing x' and y' in terms of x, y, x_0 and y_0 using (1) and (2), we can derive the transformation equations.

2.1.2. Normalization

We now subject the depth profile obtained through unrolling to a normalization process to offset differences arising due to varying depths.

Fig. 4. Expressing depth as a function of camera angle

For this, the depth of a point in the cylinder is first expressed in terms of its angle of view with respect to the camera. This is done because the mathematical expression for the normalization is derived more easily using this angular representation of the depth information. To explain better, we make use of Fig. 4 (a), where C refers to the camera capturing the image of the cylinder with radius R . Here, α refers to the maximum field of view of the camera. When the camera is placed facing the cylinder axis, the closest points on the cylinder captured by the camera (eg. point P) lie at an angle of view of α . Point Q refers to a point lying at a greater angular depth of θ_1 . When the camera is moved forward by a distance d , the same point Q (which we will refer to as Q' to

indicate the point Q viewed from the second camera position) now has a different angular depth of θ_2 .

In order to derive the angular depth of a selected FP, the y coordinate of that FP in the unrolled domain is studied. This is because the circular rings in the image are stacked up vertically in the unrolled domain, which means the y coordinate represents the count of the circle that is unrolled. The largest circle in the image is captured at the maximum field of view of the camera, α and has a radius of R_{im} . The depth of this circle in the unrolled domain is also given by R_{im} . A value of 0 in the y axis of the unrolled domain means an angular depth of 0° . The angular depths of the rest of the positions are calculated proportionally within this range of 0° and α . θ_1 and θ_2 are calculated using the y coordinates of the initial position of FP (U_{init_y}) and final position of FP (U_{final_y}) respectively (Eq 3).

$$\theta_1 = U_{init_y} \frac{\alpha}{R_{im}} \quad \theta_2 = U_{final_y} \frac{\alpha}{R_{im}} \quad (3)$$

Fig. 4 (b) shows the projections of P , Q and Q' on a plane, which is taken as the approximate image plane. δ_Q refers to the angular displacement of point Q when the camera is moved by d . This displacement δ_Q illustrates that a linear movement of the camera into the cylinder along the axis makes the FPs move radially outwards. In the unrolled domain, this is reflected by a vertical displacement of the points. For different depths of Q , the displacement δ_Q and the corresponding displacement in the unrolled domain are also of different magnitudes. So, we need to derive an expression for linear distance traversed by camera (d) in terms of the initial depth of FP (θ_1) and final depth of FP (θ_2), so that the depth information is factored.

Referring to Fig. 4 (a), we can deduce the following equations:

$$\tan\theta_1 = \frac{R}{D} \quad \tan\theta_2 = \frac{R}{D-d} \quad d = \frac{R}{\tan\theta_2} \left(1 - \frac{\tan\theta_2}{\tan\theta_1}\right) \quad (4)$$

From (4), it is seen that information on depth of FP is factored into the final expression of distance moved by camera. As seen in Fig. 1, FPs at different depths in the image show different displacements for the same movement of the camera. Through this expression from normalization, the actual displacement of the camera ' d ' is independent of the location of the FP chosen in the image. To reiterate, unrolling facilitates deriving of θ_1 and θ_2 , which can further be used in (4) to deduce d . The distance d obtained above is the distance moved downward by the camera in absolute units (in this case cm , since radius was also measured in cm).

2.2. Detecting camera rotations

Analysis of the image in the unrolled domain can be extended to study rotations of the camera. Any movement of the camera in the cylinder causes FPs to move in various directions.

Unrolling standardizes the direction of motion and provides valuable information about whether the camera has been rotated or not during the elapsed time interval. If the direction of displacement of all FPs is downward in the unrolled domain, that means there has been no rotation but a pure downward motion (Fig. 5 (a)). For the case of just a pure rotation of the camera about the axis, the displacements of all FPs would be horizontal in the unrolled domain. (Fig. 5 (b)). The greater the horizontal component, greater is the rotation and vice versa.

For estimating rotations, there is no need for calculating angular depths of FPs and normalization. This is because irrespective of how deep the points are in the cylinder, the apparent rotations remain the same. Let us call the horizontal FP displacement as ΔX . The ratio of ΔX to the length of the unrolled image directly translates into the rotation of the camera. The detectable rotation is a range from 0° to 360° . If ΔX tends to the length of the unrolled image, then this reports the maximum detectable rotation of 360° .

Fig. 5. Movement of FPs in different domains

It has been shown that the motion vectors in the unrolled domain give information about the magnitude of linear movement of the camera as well as rotation. The vertical component is directly related to the linear motion and the horizontal component corresponds to the rotation of the camera.

3. RESULTS AND ANALYSIS

The proposed method of camera motion tracking with manual inputs for unrolling is tested in a simulated environment, using an image sequence shown in Fig. 6. All codes were written in MATLAB for a detailed functional verification of the proposed method.

Images of a rolled paper cylinder are captured from various heights of the camera. *Frame1* was captured at a height of 30 cm from the ground and *Frame7* at 24 cm from ground level, with no rotations of the camera. The rest of the intermediate frames were captured with approximately 1 cm change in camera height. So the net distance traversed by the camera is 6 cm and the rotation is nil. National Semiconductor LM9628 camera was used to capture images of size 640×480 .

Fig. 6. Image sequence considered

Table 1. Estimation of camera motion on image sequence

		(x_i, y_i)	(x_f, y_f)	(u_i, u_j)	(u_x, u_y, u_f)	dist (cm)	ra (deg)
1	2	(118,253)	(106,349)	(76,1253)	(78,1240)	0.91	-0.33
		(498,339)	(500,342)	(1300,1357)	(1306,1354)	0.90	-0.93
		(316,130)	(313,124)	(504,1322)	(501,1315)	0.93	0.44
		(166,443)	(156,449)	(2159,1260)	(2161,1248)	0.89	-0.23
						0.91	-0.27
2	3	(106,249)	(97,250)	(108,1288)	(102,1279)	0.88	0.90
		(503,302)	(510,301)	(1188,1307)	(1187,1300)	0.86	0.22
		(312,124)	(313,116)	(604,1316)	(606,1308)	1.05	-0.27
		(156,449)	(148,460)	(2117,1291)	(2112,1278)	1.34	0.70
						1.03	0.39
3	4	(150,271)	(150,268)	(75,1290)	(77,1283)	0.96	-0.31
		(544,166)	(551,157)	(939,1269)	(935,1258)	0.93	0.60
		(349,61)	(347,50)	(576,1248)	(574,1237)	0.80	0.28
		(301,450)	(298,453)	(1996,1348)	(1969,1344)	0.80	-0.54
						0.87	0.01
4	5	(151,269)	(142,265)	(86,1280)	(89,1271)	0.92	-0.47
		(487,262)	(490,260)	(1036,1366)	(1034,1362)	0.90	0.24
		(309,108)	(307,100)	(500,1283)	(501,1274)	0.87	-0.03
		(301,451)	(298,454)	(1971,1352)	(1975,1348)	0.88	-0.53
						0.89	-0.20
5	6	(71,242)	(56,236)	(92,1228)	(95,1212)	0.95	-0.45
		(541,292)	(550,289)	(1172,1293)	(1168,1284)	0.96	0.62
		(344,34)	(342,17)	(613,1227)	(609,1210)	0.99	0.51
		(195,453)	(188,459)	(2092,1298)	(2094,1289)	1.00	-0.24
						0.98	0.11
6	7	(133,258)	(122,253)	(109,1261)	(111,1249)	0.93	-0.39
		(550,287)	(556,281)	(1124,1308)	(1115,1301)	0.87	1.35
		(380,31)	(378,14)	(623,1205)	(620,1199)	0.90	0.56
		(297,457)	(288,457)	(1975,1353)	(1995,1349)	0.87	-3.06
						0.89	-0.38
						5.87	-6.35

It's maximum field of view is 22.5 degrees and the radius of the paper cylinder used was 4.5 cm. Since it is a rectangular image, it is assumed that the diagonal of the image subtends the maximum field of view.

To extract features, an edge profile of the simulated images was first obtained using wavelets. Subsequently, areas of the edge profile with highest edge-content was selected using entropy as a measure. The proposed method processes images in pairs where FPs are first located in the first image and their corresponding locations are tracked in the second. Image correlation was used for FP tracking. MATLAB was used for performing FP extraction and tracking.

The initial and final coordinate positions thus obtained are now analyzed in the unrolled domain. Centers for unrolling are fed in manually using the MATLAB command, *getpts*. The displacements are calculated and normalized using (4). Rotations are inferred from the horizontal components of FP displacements. Table 1 shows the initial and final coordinate values of the FPs in the image as well as in the unrolled domain for every image pair. The coordinates of the FPs in the first image of the pair are referred to as (x_i, y_i) and the corresponding coordinates in the second are referred to as (x_f, y_f) . (ux_i, uy_i) and (ux_f, uy_f) are used to denote the initial and final positions in the unrolled domain.

It can be seen that the net distance moved by the camera as estimated by the system is about 5.57 cm. The distance

recorded experimentally was 6 cm, indicating a 7% error in the distance measured using the proposed method. The rotation was detected to be a negligible value of -0.35° , which coincides with the experimental value since no camera rotation was performed while capturing the images. The error noted in rotation is 0.09%. These errors could be because of slight imperfections in geometry of cylindrical passages considered and approximation of parameters like center for unrolling, angle of view of camera, radius of cylinder for analysis.

4. CONCLUSION

The method proposed in this paper for detection of linear movements of camera requires only a few feature points, indicating its feasibility for use in real time systems. It was also shown that the overall output is independent of the feature points selected due to the proposed unrolling and normalization processes, thus making it highly suited for deployment in dynamic environments. The horizontal and vertical components of displacement values were elegantly utilized to detect rotations as well. This integrated method was successfully applied to a sequence of images captured in a simulated environment to estimate linear distance and rotation within a maximum error bound of 7%.

5. REFERENCES

[1] T.Okatani, and K. Deguchi, "Recovering camera motion from image sequence based on registration of silhouette cones", *IAPR Workshop on Machine Vision Applications*, 2000.

[2] J.W.Lee, S.You, and U.Neumann, "Large Motion Estimation for Omnidirectional Vision", *IEEE Workshop on Omnidirectional Vision*, pp. 161, 2000.

[3] T.Thormaehlen, H.Broszio, and P.Meier, "Three-Dimensional Endoscopy", *Technical paper*, cite-seer.ist.psu.edu/thormaehlen02threedimensional.html, 2002.

[4] K.Mori, D.Deguchi, J.Sugiyama, Y.Suenaga, J.Toriwaki, C.R.Maurer Jr, H.Takabatake, and H.Natori, "Tracking of a bronchoscope using epipolar geometry analysis and intensity-based image registration of real and virtual endoscopic images", *Med Image Analysis*, vol. 6(3), pp. 321-36, 2002.