Multi Camera Stereo and Tracking Patient Motion for Compensation in SPECT Scanning Systems

by
Suman Nadella

A Thesis
Submitted to the Faculty
of the
Worcester Polytechnic Institute
In partial fulfillment of the requirements for the
Degree of Master of Science
in
Computer Science

August 2005

APPROVED:

Prof. Michael A. Gennert, Thesis Advisor

Prof. Matthew O. Ward, Thesis Reader

Prof. Michael A. Gennert, Head of Department

Abstract

Patient motion, which causes artifacts in reconstructed images, can be a serious problem in Single Photon Emission Computed Tomography (SPECT) imaging. If patient motion can be detected and quantified, the reconstruction algorithm can compensate for the motion. A real-time multi-threaded Visual Tracking System (VTS) using optical cameras, which will be suitable for deployment in clinical trials, is under development. The VTS tracks patients using multiple video images and image processing techniques, calculating patient motion in three-dimensional space.

This research aimed to develop and implement an algorithm for feature matching and stereo location computation using multiple cameras. Feature matching is done based on the epipolar geometry constraints for a pair of images and extended to the multiple view case with an iterative algorithm. Stereo locations of the matches are then computed using sum of squared distances from the projected 3D lines in SPECT coordinates as the error metric. This information from the VTS, when coupled with motion assessment from the emission data itself, can provide a robust compensation for patient motion as part of reconstruction.

Keywords

Feature matching in multiple cameras

Multi camera stereo computation

Patient motion tracking

SPECT Imaging

Acknowledgments

First and foremost, I would like to thank my amazing thesis advisor and guide *Prof. Michael A.Gennert*. He did a lot more than just guide my thesis work. He advised, motivated and provided me with the encouragement that I needed from time to time to reach my goals. Whilst his patience and able guidance have given me knowledge, his trust in me has furnished me the necessary confidence in myself. I consider myself to be very fortunate to have him as my advisor. Thank you Mike, for everything!

I would also like to extend my grateful thanks to Prof. Matt Ward, for being the reader for this thesis. I express my gratitude to the members of University of Massachusetts Medical School, Dept of Nuclear Medicine who put up with all my amateur ways and guided me in many ways. In particular, I would like to thank Drs. Michael King, Philippe Bruyant, Richard Beach and Hennie Boening. I also always had consistent co-operation from my team members, Joel Morgenstern, Neeru Kumar, Songxiang Gu and Linna Ma. Special thanks for Prof. Emmanuel Agu for always encouraging me.

A very big thank you to my dearest sister Chandra and her husband Sunil. Not only did they put up with my tantrums and pick me up from school at midnights,

they did everything possible to make me feel at home. Special mention to their bundle of joy, little Rohan, who brightened up even my bleakest possible day! Thanks Mom and Dad for giving me everything I wanted, even before I asked for it. It's your blessings that made me accomplish all that I did. Thanks also to all my friends at WPI, for always showing confidence in me.

The project 'Patient motion detection and compensation in SPECT' is supported by National Institute of Biomedical Imaging and Bioengineering under grant R01-EB001457-01.

Contents

List of Figures

List of Tables

Chapter 1
Introduction

Single photon emission computed tomography (SPECT) is a form of medical imaging technology that provides a three-dimensional image of a patient's various biological systems. Reconstruction of three-dimensional images that are produced from the SPECT process is based on the composition of numerous two-dimensional images, or slices. Patient motion is an ever present potential cause of artifacts that can limit accuracy of diagnostic imaging [1]. The problem is especially significant for imaging modalities such as SPECT and PET which require the patient to remain motionless for protracted periods of time. Due to the serious consequences associated with errors in medical technology, such as misdiagnosis or delayed treatment of a serious illness, producing accurate images is an issue of great significance. Thus, the detection and correction of patient motion in SPECT imaging is important.

Compensation strategies for motion in SPECT imaging that rely exclusively on emission data itself are inadequate for clinical usage [1, 2]. The research, of which this thesis is a part, aims to devise a more robust method by using an external tracking system that provides additional and independent data [2]. The use of external

tracking devices, bringing information independent of SPECT data, is expected to be much more robust [2, 3].

The basic principle is to determine the 3D motion of the patient during the SPECT acquisition using optical cameras and stereo techniques, and then to use this information post-acquisition to compensate for motion in the SPECT data during reconstruction. A visual tracking system (VTS) is under development for this purpose. The VTS comprises a set of optical cameras, a SPECT/VTS calibration phantom, a garment with reflective spheres to track chest motion and a computer for camera control and data acquisition in synchrony with acquisition on SPECT [3]. The images obtained from the optical cameras, at the rate of up to 30 frames per second, constitute the raw data for detecting patient motion. Once the images are obtained, the reflective spheres are matched in all the views using the proposed feature matching algorithm. The 3D locations of these sets of sphere data, when computed using stereopsis, make it possible to compute the visible surface, that is, the patient surface that is visible in at least two cameras. When the patient moves, the images will change accordingly, and the surface is recomputed. The difference between the newly computed surface and a previously computed surface is used as an estimate of patient motion. This motion estimate may then be used for correction of the tomographic reconstruction by rebinning the raw gamma camera data to undo the motion prior to reconstruction [4]. The SPECT system operates in list mode. In this mode, all detected events are stored in a long list together with an event timestamp

[5]. Thus, it is possible to temporally register the time stamped images with the detected activity.

Chapter 2
SPECT Imaging

This chapter presents background concerning SPECT imaging, a brief overview of the various components of the scanning system and applications that make it such a significant diagnostic imaging modality.

2.1 Single Photon Emission Computed Tomography

Single photon emission computed tomography (SPECT) imaging is the process of reconstructing three-dimensional data concerning tissues of interest in a patient from two-dimensional images, or slices [6]. SPECT imaging is different from other forms of imaging that use radiation, such as x-ray imaging, in that a pharmaceutical labeled with a radioactive isotope is injected into the patient. The area of interest absorbs the pharmaceutical and emits gamma rays as the isotope decays. As the gamma rays exit the tissue, they can be captured using a gamma camera, as shown in Figure 1.

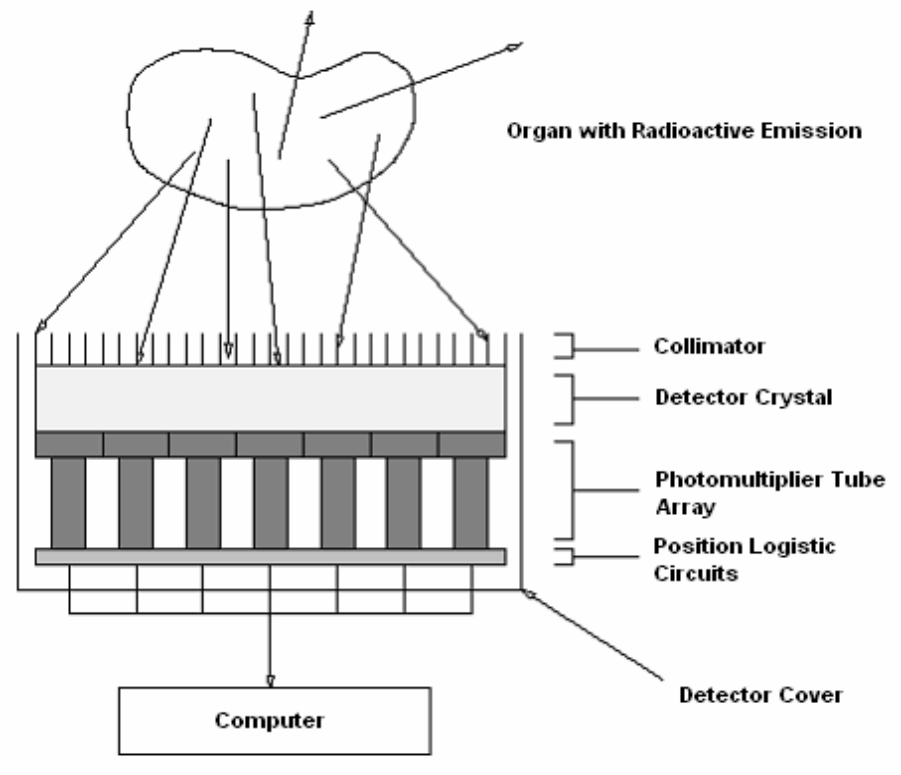

Figure 1 - SPECT Imaging Using a Gamma Camera [6]

A gamma camera is composed of five essentials parts: the collimator, detector crystal, photomultiplier tube array, position logistic circuit, and computer. Collimators are composed of lead or other materials that have strong radiation absorption, having holes to allow gamma rays traveling perpendicular to the detector crystals to pass. The function of the collimator is to provide uniformity in the image, allowing a tolerance of rays traveling within a small range of angles to make contact with the detector crystal. This provides a clear two dimensional projection of the area at the current angle of the camera head.

The detector crystal is usually composed of thallium-activated sodium iodide (NaI [T1]). Photons are released from the crystal as gamma particles that have passed

through the collimator come into contact with the crystal. These photon emissions occur in patches that are sparse and weak, so they must be amplified. Amplification is achieved using the photomultiplier tubes (PMT) attached to the crystal. The average gamma camera can have between 37 to 91 tubes in its array. The photomultipliers normally amplify the output of the crystals by a factor of 6 to 10 [6]. The amplified output from the PMT is passed to the position logistic circuits, where areas of activity are recorded and processed. This information is then output to a computing system where the data is analyzed.

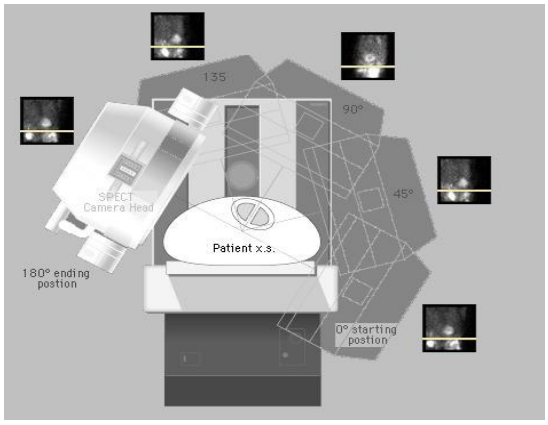

Figure 2 - Gamma Camera System functionality [7].

An example of a gamma camera system is shown in Figure 2, above. The gamma camera heads circle the patient as they lie on the flat platform in the center of the apparatus. Two-dimensional projections are obtained from the gamma camera heads, which are used to reconstruct three-dimensional estimates of the radionuclide distribution in the patient [7].

2.2 Applications

SPECT Imaging has a wide variety of uses in the present day and new studies are being conducted to expand its range. The major areas of application are [8] –

- Cardiac Perfusion - Measures the amount of blood flow to the heart muscle to determine healthy and unhealthy regions.
- Cardiac Volume – Determine the amount of blood pumped by the heart.
- Lesions – Detection of any unwanted lesions in the heart, lung or lymph nodes.
- Scans – Bone scans, Kidney/Renal Imaging, Brain Imaging

2.3 Significance of the Research

"Despite multiple advances in the technology of myocardial perfusion SPECT, patient motion remains a problematic source of error" [9]. In SPECT cardiac imaging, body motion has been determined to occur in ~25% of studies and ~5% of the time motion is significant enough to cause artifacts that can mislead diagnosis [10]. Generally it has been reported that motion of 2 or more pixels (>13mm) was sufficient to create minor to moderate defects in the tomographic data [11].

Respiratory motion is present in all cardiac perfusion SPECT studies. It can result in a blurring of the structures of the heart producing apparent decreases in activity in the inferior and anterior walls that can be mistaken for perfusion defects [12].

Upward creep is the term given to the slow upward movement of the heart in the chest during the course of imaging. Testing with various imaging agents and different stress conditions revealed that subtle changes in respiratory pattern related to recumbence, may also play an important role in the origin of upward cardiac creep [13].

Separate emission and transmission imaging are frequently employed in SPECT and PET. Despite being warned by the technologists to expect a change in motion of the camera heads as the system switches from emission to transmission imaging, some patients relax and move with the cession of the camera head movement at the end of emission imaging. A number of studies have reported the deleterious impact of a mismatch between emission and transmission scans in cardiac SPECT and PET [14].

Thus we can see that various forms of patient motion in total represent a major clinical problem the solution to which would significantly improve the accuracy of clinical imaging, as well as reduce patient care costs by decreasing the need for repeat imaging. The proposed visual tracking system is an innovative approach that has the potential for providing a generic solution that would operate in a clinical setting with minimum impact on current imaging protocols and cooperation from the patient.

Chapter 3
Visual Tracking System

This chapter covers a detailed overview of the VTS from a hardware perspective. Various components and the way they work together are discussed followed by a briefing of the various advantages of this system.

As mentioned in the introduction, the VTS is made up of optical cameras, a calibration phantom, a stretchy garment with reflective markers that are to be tracked and a computer for data acquisition and synchronization purposes. When the patient is lying down on the SPECT imaging table, six optical network cameras are positioned to look at him/her from both the head and foot side of the gantry [15]. These cameras are fixed in their locations in the clinic to prevent any disturbance in the positions due to external factors.

The illustration in Figure 3 visualizes the clinical deployment of the VTS in real world circumstances [16].

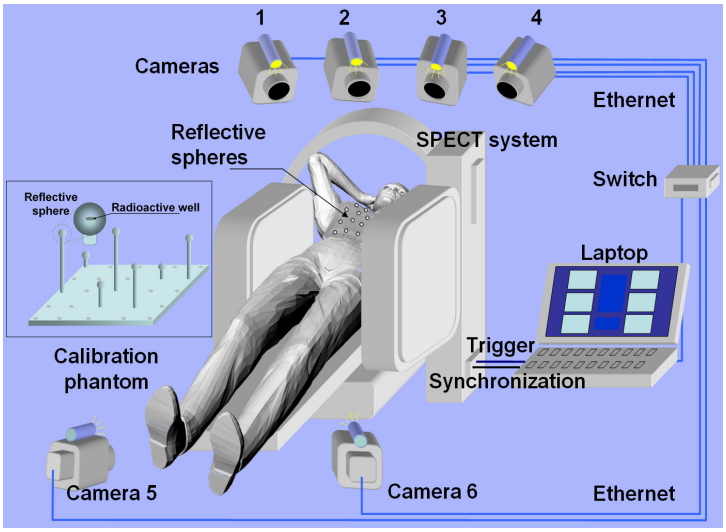

Figure 3 – Clinical deployment of VTS [16].

3.1 Gamma Camera

The Gamma (SPECT) Camera being used throughout this project is the Philips IRIX™ Exclusive Triple-detector Camera, belonging to the Dept. of Nuclear Medicine at University of Massachusetts Medical School.

3.2 Optical Camera

The optical cameras used are AXIS 2130 PTZ Network Cameras capable of transmitting sequential JPEG images over a network. This is an all-in-one integrated networked pan/tilt/zoom camera that allows users to remotely control the functions live over a local area network or the Internet. For mounting on the wall, the AXIS 2130R Wall Bracket was used. It transmits up to 30 frames per second, with a maximum resolution of 704x480 pixels. The Compression used it MJPEG and each

camera has a built-in web server that serves a video image stream on command. Figure 4 shows one of the Axis Network Cameras shown sitting on top of the hi-speed switch to which all cameras are attached when in use. On top of the camera is a light source using an LED.

Figure 4 - Philips IRIX Gamma Camera [17] and AXIS Network Camera [15].

The figure also shows the device for mounting the light source on the camera so that the light moves in synchrony with the camera when it is moved by pan or tilt [15]. In the clinic, these cameras aim at the patient's chest, thereby viewing the stretchy garment worn. This garment has 16 reflective spheres attached to it, which glow in a dark ambience when the light on top of each camera falls on it. The purpose of this is to provide features for matching in different views, thereby getting a depth estimate for each frameset.

3.3 Stretchy Garment with Reflective Spheres

Sixteen highly retro-reflective spheres are clipped on a garment made of black stretchable fabric. This garment is placed around the chest and the spheres 3D

motion is tracked by stereo techniques. Figure 5 shows one version of the design that was used in experiments. Inserts in the picture show the mounting pin with and without sphere attached. Presently a different pattern of spheres is designed so as to reduce the number of blobs that are being blocked. The new pattern reduces the co-linearity of their orientations and symmetry in the heights. A snapshot of torso phantom with this design can be seen in the center block of Figure 5. Note that the spheres on the garment are never radioactive.

3.4 Calibration Phantom

The 3D motion of the spheres is tracked using standard stereo computation techniques. A calibration is required beforehand to determine the transformation matrix used to compute the 3D locations of the markers in SPECT coordinates from their 2D image counterparts [18]. A calibration phantom as shown in Figure 5 was designed for this purpose.

Figure 5 – Left - One version of Stretchy Garment design with Reflective Spheres, Center – Asymmetrical version of the Garment, Right - Calibration Phantom [15].

Seven reflective spheres are placed on top of rods of various heights and whose upper end is a well that can receive a drop of radioactivity. During the calibration step, the phantom is placed in the field of view of the gamma camera. Snapshots are taken with the optical cameras and a SPECT acquisition is performed. Precaution is taken that all the seven spheres are visible in all camera views [16]. The significance of the number seven is discussed below in the Calibration section. The Software is run on the images acquired and the calibration matrix is computed. After calibration, the phantom is removed and the patient is installed on the bed, wearing the garment presented in figure 5. The optical cameras track the chest motion during the acquisition and a post-acquisition program converts the motion into the 3D SPECT coordinates system using the calibration.

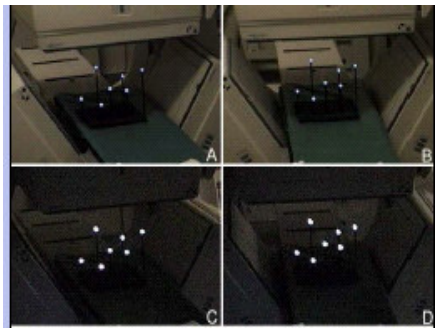

Figure 6 – Calibration Phantom on the SPECT table with and without lights on [15].

Figure 6 represents images acquired from a stereo pair of cameras. The top row is pictures of our calibration phantom lying on the bed of SPECT system with room lights on. The second row shows the visibility of the spheres of the calibration phantom with room lights off.

3.5 Clinical Setup

In figure 7 below, the first row shows stereo images acquired with room lights on a normal subject lying in imaging position on the bed of the SPECT system with a garment made up of four rows of four spheres each. The last row shows the same images as above, except the room lights have been turned off. Notice that the spheres are easily visible regardless of room lighting.

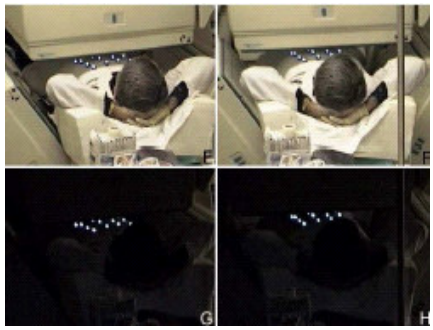

Figure 7 – Patient lying down on the SPECT table with and without lights on [15].

The cameras are connected to a laptop via a 1-gigabyte switch. A LabView program (National Instruments Co. Austin, TX) is used to -

1) Control camera individual pan, tilt and zoom
2) Synchronize the video acquisition with the gamma-camera acquisition
3) Store the 6 video streams as multiple-JPEG (M-JPEG) files

Figure 8 shows a snapshot of the GUI of the VTS system. Additional options include the ability to perform calibration, set FTP settings to acquire the required SPECT file from the network, default work settings and advanced manipulation

among other functionalities. The JPEG images acquired have a size of 352 x 240 pixels, which is sufficient to track the centroid of the spheres with 1-mm accuracy.

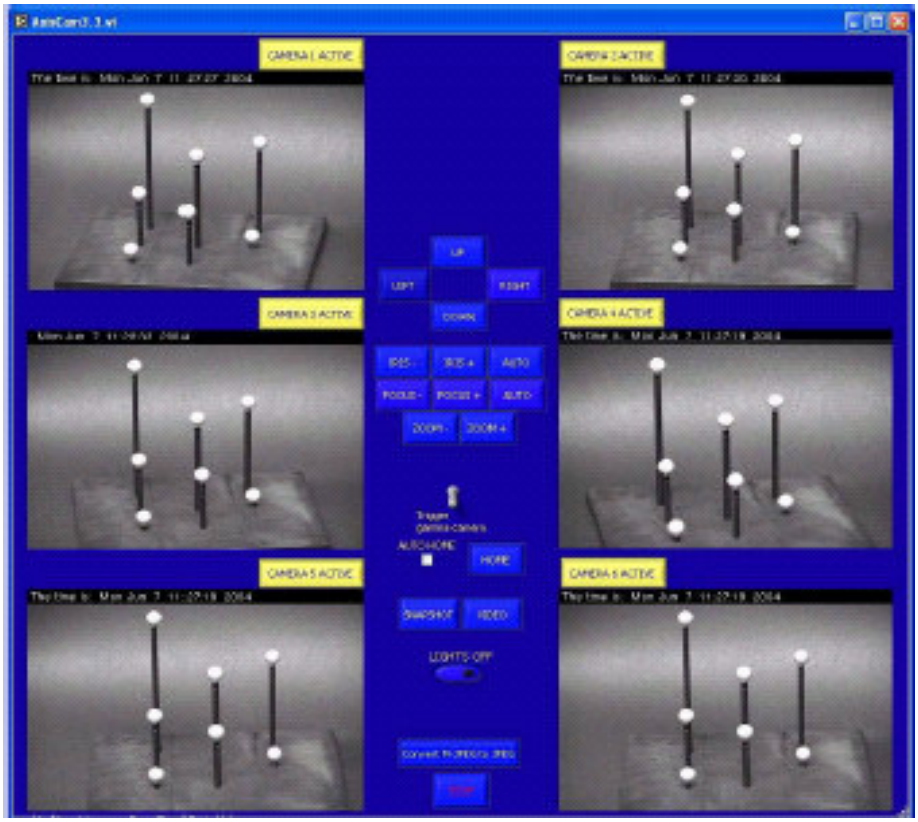

Figure 8 – GUI of the VTS system looking at the calibration phantom in 6 views [15].

3.6 Advantages of the System

Detecting patient motion during SPECT data acquisition using optical cameras has the following advantages:

- Allows higher resolution image reconstruction by reducing motion blur artifacts
- Does not depend on emission data
- Does not contact the patient, making it easy to deploy in a clinical setting

Using multiple cameras is expected to perform better than a two camera system by:

- Giving a longer field of view of the patient
- Reducing the occurrence of missing markers
- Reducing the ambiguity of stereo matching
- Improving the accuracy of 3D location determination

Chapter 4
Software Architecture

This chapter details the basic software architectural premise of the VTS system. The system has a series of network cameras connected by an IP network to a computer system running the VTS Software. High performance is achieved by acquiring and processing images in parallel [4]. Each camera is associated with a thread of control within the VTS. As images are acquired, they are stored in buffers until needed. Another thread is responsible for requesting images from a specific time, and processing those images as a stereo set. 2D blob detection is also done here to isolate the centroids of the markers. Likewise the capability of doing 3D blob detection to find the SPECT radioactive marker positions is also present so that it could be used during calibration.

XML is used as the standard for data transfer among all the modules in the system. This was incorporated keeping in mind that XML has various advantages over other formats when working on a .NET platform. Figure 9 shows the software architectural design of the system.

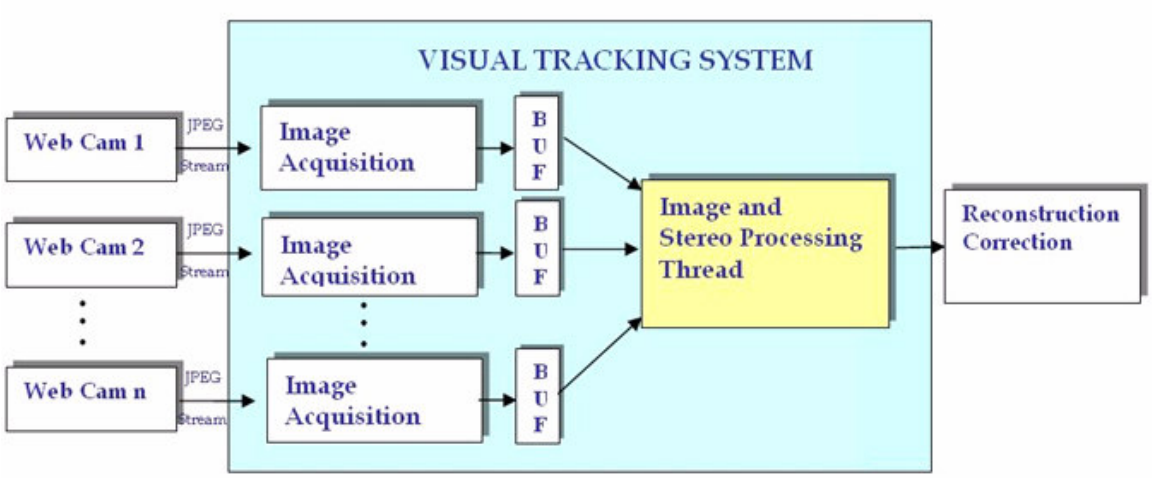

Figure 9 – Software Architecture Design on the VTS System [4].

4.1 Image Acquisition Threads

The software receives a video stream from each camera in each Image Acquisition Thread. The thread establishes a network connection to each camera, receives a Motion-JPEG (MJPEG) stream, and then parses and timestamps each JPEG video frame received from the camera. The JPEG images are then saved into buffers in the form of Motion Track Objects (MTOs) [4]. Each MTO has several properties and methods, one being the compressed JPEG data itself. Thus, video images can be acquired asynchronously.

4.2 Buffers

Each buffer consists of a FIFO containing MTOs. There is one buffer for each camera image stream. Images are buffered and later used by the Image and Stereo Processing Thread. The purpose of the buffer is to allow the frame grouping

function to select a set of frames with similar timestamps. When the Image and Stereo Processing Thread requests an image set, the buffers supply the frames whose timestamps are closest together.

4.3 Image & Stereo Processing Thread

The Image Processing Thread is responsible for grouping a set of frames (one from each camera), using image processing methods to perform conversion to a monochrome image with thresholding and detect two dimensional marker coordinates, finally matching the marker coordinates to produce a list of three-dimensional marker coordinates. The last two stages of blob matching and stereo processing are of concern to this thesis.

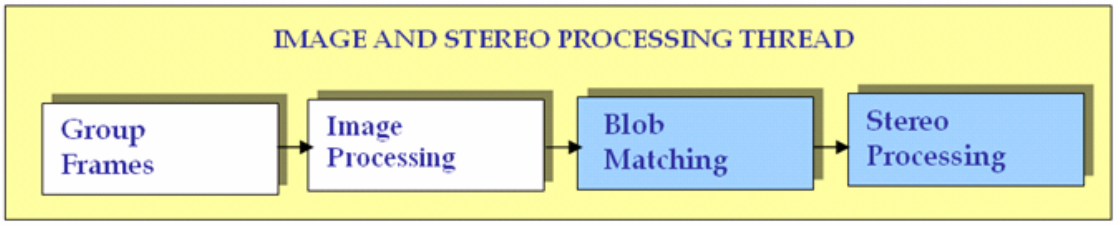

Figure 10 – Image and Stereo Processing Thread Components [4].

4.3.1 Group Frames

This module is responsible for producing a set of frames—one frame from each camera—to be used for stereo processing. A master camera is selected. A master frame is then selected from that camera's buffer by going back several frames previous to the current time. The timestamp from the master frame is determined,

and the module finds the frames from the remaining cameras closest in time to the master frame. The ability to 'go back in time' allows the module to essentially 'look into the future' to determine which frames are closest in time.

4.3.2 Image Processing

The Image Processing functions uncompress the incoming JPEG images, detect markers, crop the image, and save images to disk if requested. Results of this function consist of an array of two-dimensional marker coordinates written to XML files that are used in the remainder of the stereo processing functions.

- Uncompress JPEG - Images are received from the buffer as compressed JPEG images. JPEG images are decompressed into 24-bit RGB bitmaps for processing.

- Binarize Images - The images are binarized using a threshold value (default value is around 250, since we are dealing with grayscale images) to separate the markers from the background.

- Marker Detection - Segmentation is performed on the Images to locate the centroids of the markers. This module has both 2D and 3D routines, to be used during calibration.

- Image Cropping - These Images are then cropped to show only the markers. This function crops the image to the upper-right and the lower-left marker coordinates, with a degree of padding added to make sure entire markers are

preserved for use by later functions. Patient privacy concerns have prompted this in the event that images are saved back to disk or are transmitted in such a manner that patient privacy may be compromised.

The last two functions of the Image and Stereo Processing thread, namely blob matching and stereo processing constitute the essence of this thesis.

4.3.2.1 Blob Matching

The frame set with the detected blobs is passed on to the matching routine. Taking a camera as the base (by default the first IP camera) a set of possible matches for each marker in the corresponding frame of the second camera is created as a linked list. Then projecting these views into another view and using certain epipolar geometry constraints (detailed in the chapter on feature matching), the best match between all views is identified and written into another final match data structure. This is proposed to be used as the basis for matching the markers in the following frames, so as to reduce the overhead of performing the same process again and again when there may not be much motion between frames.

4.3.2.2 Stereo Processing

The final match data structure is passed onto this module for each frameset. Using the calibration results (written into XML files, one per camera) and matches between markers passed on, it computes the three dimensional real space locations of the markers. Here the results are in SPECT coordinates since it is taken as the

reference coordinate system. The computed 3D values are then passed on to the Reconstruction/Correction module where the compensation for motion is done.

Chapter 5
Calibration

The optical cameras are placed subject to certain physical clinical position constraints at some distance from each other and from the spheres. The PTZ (Pan, Tilt, and Zoom) features can also be set arbitrarily in order to get the best pictures of the field of view. This requires the stereo system to be calibrated with respect to the SPECT coordinates so that a motion detected by the VTS can be expressed in the SPECT coordinate system.

5.1 Theory

There are three main fields of reference concerning camera calibration: the camera reference frame, the image reference frame, and the real world reference frame [19]. Camera calibration is necessary to allow for the proper computation of the position of observed objects in the real world when using two or more optical cameras. In order to achieve this, the intrinsic and extrinsic parameters of the camera system must be known. A brief description of the math involved is given below. Trucco and Verri [19] define the intrinsic parameters of a camera as,

The focal length, f

The effective pixel size in the horizontal and vertical directions, s_x, s_y

The image center coordinates, and (o_x, o_y)

The radial distortion coefficient, k_1

These parameters are needed to characterize the optical, geometric, and digital characteristics of the camera. The extrinsic parameters of a camera specify the transformation between the camera and world reference frames. These parameters define the extrinsic parameters of a camera as,

A translation vector, T_w^c , and

A rotation matrix. R_w^c

The translation vector T_w^c is a three-dimensional translation vector that describes the relative positions of the origins of the camera and world reference frames. The rotation matrix R_w^c brings the corresponding axes of the two reference frames together. Each reference frame has its own respective coordinate system, and representing the position of a point in one frame in another reference frame is achieved through a series of rotations and translations on a point. Table 1 presents the representation of a point in each reference frame's coordinate system.

World	p^w	(x^w, y^w, z^w)
Camera	p^c	(x^c, y^c, z^c)
Image	p^i	(x^i, y^i, z^i), $z^i = f$

Table 1 - Point Representation in World, Camera, and Image Reference Frames

A point p^w in the world reference frame is represented by the point p^c in the camera reference frame through the following:

$$p^c = R_w^c \; p^w + T_w^c$$

Likewise, x^c, a point in the camera reference frame is represented by the point x^w in the world reference frame using the equation:

$$p^w = R_w^{c\,-1} (p^c - T_w^c)$$

A point $p^c = (x^c, y^c, z^c)$ in the camera reference frame is represented by the point $p^i = (x^i, y^i)$ in the image reference frame by:

$$x^i = -\frac{f}{s_x}\frac{x^c}{z^c} + o_x$$

$$y^i = -\frac{f}{s_y}\frac{y^c}{z^c} + o_y$$

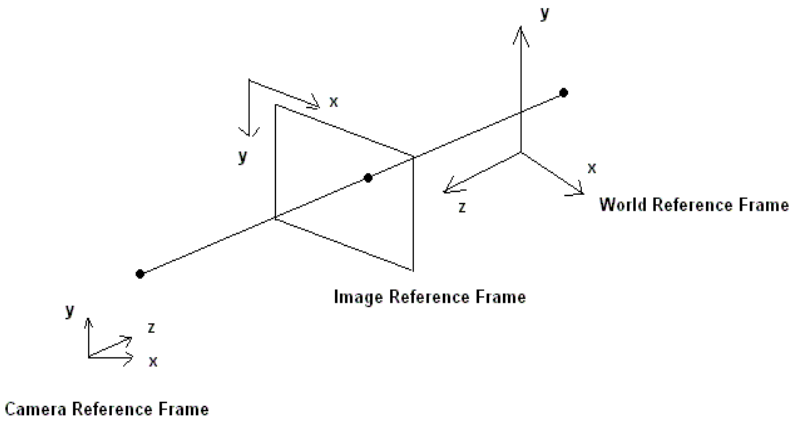

Figure 11 - World, Camera, and Image Reference Frames [19].

These relationships between points in different reference frames allow for easy representation of a point in one reference frame in another. The relationship between the three reference frames is summarized in figure 11.

5.2 Implementation

In order to compute the Camera Parameters, the Calibration Phantom described earlier in Chapter 3 is used. Seven reflective spheres are placed on top of rods of various heights and whose upper end is a well that can receive a drop of radioactivity. The arrangement is non-coplanar and asymmetric, guaranteeing a unique solution for the calibration equations [18]. Figure 12 shows the various stages.

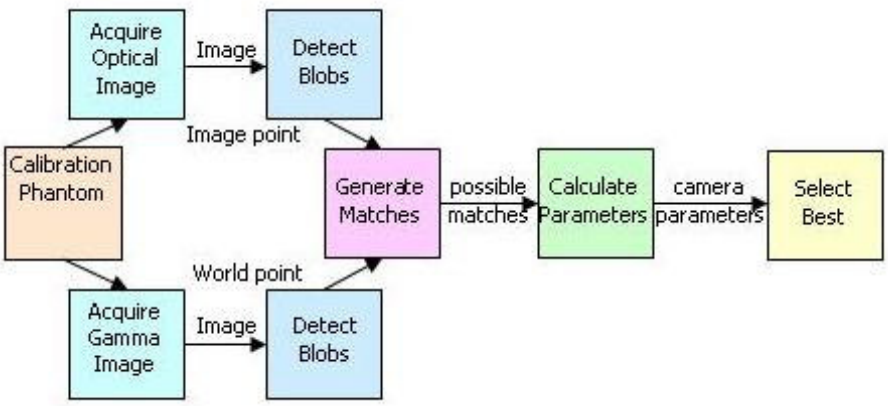

Figure 12 – Calibration Processing Flow [18].

Similar to the VTS operation, the Optical and Gamma Images are obtained and the centroids are determined, forming Image Point and World Point lists. The world to image transformation equations are written forming 2 equations with 12 unknowns.

World point $x^w = (x^w, y^w, z^w)$ projects to $x^i = (x^i, y^i)$ where

$$x^i = \frac{m_{11}x^w + m_{12}y^w + m_{13}z^w + m_{14}}{m_{31}x^w + m_{32}y^w + m_{33}z^w + m_{34}}$$
$$y^i = \frac{m_{21}x^w + m_{22}y^w + m_{23}z^w + m_{24}}{m_{31}x^w + m_{32}y^w + m_{33}z^w + m_{34}}$$

Rewrite to be linear in m_{ij} as

$$\left(m_{31}x^w + m_{32}y^w + m_{33}z^w + m_{34}\right)x^i = m_{11}x^w + m_{12}y^w + m_{13}z^w + m_{14}$$
$$\left(m_{31}x^w + m_{32}y^w + m_{33}z^w + m_{34}\right)y^i = m_{21}x^w + m_{22}y^w + m_{23}z^w + m_{24}$$

Each world / image point pair gives

$$\begin{pmatrix} x^w & y^w & z^w & 1 & 0 & 0 & 0 & 0 & -x^i x^w & -x^i y^w & -x^i z^w & -x^i \\ 0 & 0 & 0 & 0 & x^w & y^w & z^w & 1 & -y^i x^w & -y^i y^w & -y^i z^w & -y^i \end{pmatrix} \begin{pmatrix} m_{11} \\ m_{12} \\ \vdots \\ m_{34} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

N point pairs yield $2N$ equations, so we need ≥ 6 point pairs to compute m_{ij}. Thus we need at least 7 point pairs in order to solve this.

Now we solve $\mathrm{Am}=0$ where

$$\mathbf{A} = \begin{pmatrix} x_1^w & y_1^w & z_1^w & 1 & 0 & 0 & 0 & 0 & -x_1^i x_1^w & -x_1^i y_1^w & -x_1^i z_1^w & -x_1^i \\ 0 & 0 & 0 & 0 & x_1^w & y_1^w & z_1^w & 1 & -y_1^i x_1^w & -y_1^i y_1^w & -y_1^i z_1^w & -y_1^i \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_N^w & y_N^w & z_N^w & 1 & 0 & 0 & 0 & 0 & -x_N^i x_N^w & -x_N^i y_N^w & -x_N^i z_N^w & -x_N^i \\ 0 & 0 & 0 & 0 & x_N^w & y_N^w & z_N^w & 1 & -y_N^i x_N^w & -y_N^i y_N^w & -y_N^i z_N^w & -y_N^i \end{pmatrix}, \mathbf{m} = \begin{pmatrix} m_{11} \\ m_{12} \\ \vdots \\ m_{34} \end{pmatrix}$$

Singular Value Decomposition (SVD) is used to solve this, to find m from A's null space and all the parameters can be reconstructed from the matrix m. More about SVD can be found in Appendix A.

```
IPL = ImagePointList[
 ImagePoint(177.5,  82.0), ImagePoint(222.5, 211.0),
 ImagePoint(193.5, 349.0), ImagePoint(293.5, 289.5),
 ImagePoint(359.0,  83.5), ImagePoint(312.0, 347.0),
 ImagePoint(269.5, 437.0)]
WPL = WorldPointList[
 WorldPoint[88.3, 39.7, 62.2], WorldPoint[66.9, 87.3, 79.6],
 WorldPoint[82.4, 87.1, 31.0], WorldPoint[53.8, 87.5, 48.3],
 WorldPoint[42.0, 40.2, 71.6], WorldPoint[53.7, 86.8, 29.7],
 WorldPoint[70.5, 86.4,  6.8]]
Res = 9.05
CPs = Camera Parameters[
 T:[79.8, -19.6, 104.2],
R:[[-0.923, -0.262, -0.280],
   [-0.010,  0.747, -0.664],
   [-0.383,  0.610,  0.692]],
IC:[[317.2], [238.6]],
fx:650.7, fy:672.4]
```

Figure 13 - Sample Output with residual error=9.05 [18].

5.3 Quality Control

Because Calibration is a crucial step, the assessment of its validity should be part of daily quality control. However, performing a full VTS/SPECT calibration before each clinical acquisition would be a time-consuming task since it involves acquisition by both the SPECT system and optical cameras of a phantom made up of reflective spheres with activity at their centers. Though the attached optical cameras wouldn't change location relative to SPECT, it is observed that the PTZ values can drift with time or on reboot.

In order to deal with this, a method was devised where flat reflective disks are stuck onto the wall behind the gantry of the gamma camera, so that they are seen by

the optical cameras when there is no patient present [20]. Whenever the cameras are rebooted / drifted, a correction matrix is calculated for each camera using images of markers on the wall before and after the motion. This transformation, when applied on the new patient images, corrects the changes without the need for a new SPECT acquisition.

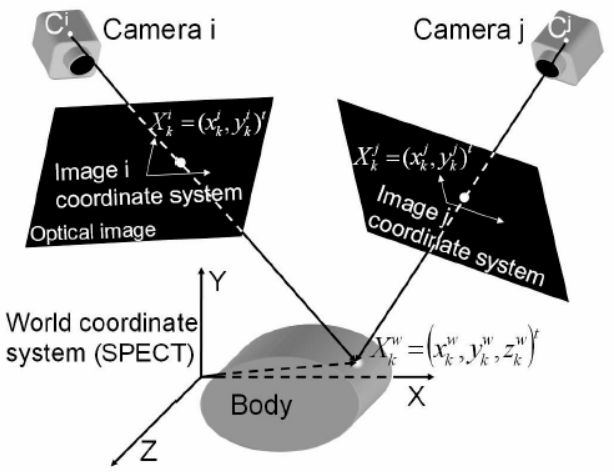

Figure 14 – Coordinate Systems for VTS and SPECT. The lines (C^i,X^i) and (C^j,X^j) intersect at X^w [20].

The center of the SPECT coordinate system is taken as one corner of the stack of reconstructed slices as shown in figure 14. The Z axis is perpendicular to the slices and the X and Y axes are in the plane of the slices. The output of the calibration is a (4,3) matrix M^i accounting for the relationship between the coordinates $X^i=(x^i,y^i,1)^t$ in the optical images of the i^{th} camera and the coordinates $X^w=(x^w,y^w,z^w,1)^t$ in the SPECT system of the same point such that $M^iX^w=X^i$.

The determination of the 12 coefficients of M^i requires at least 6 points detectable by both the VTS (providing the X^is) and the SPECT system (providing the

X^ws). For that reason, the calibration phantom consisting of 7 optically visible spheres whose center can contain a drop of radioactivity is used [20]. In order to check for a possible PTZ change since the last calibration, retro-reflective markers attached to the walls are employed and their locations in optical images acquired before each clinical acquisition to their locations in images acquired during the last VTS/SPECT calibration are compared. Let's call $X'^i=(x'^i,y'^i,1)$ the new location for any point X^i after camera motion. A (3,3) matrix P^i accounting for the PTZ changes is determined such that: $X^i=P^iX'^i$. Combining the equations, we obtain

$M^iX^w=P^iX'^i$, and if P^i is invertible, $(P^i)^{-1}M^iX^w=X'^i$.

Thus to compensate for camera motion the calibration matrix M^i is simply replaced by $(P^i)^{-1}M^i$. Figure 15 shows the results obtained.

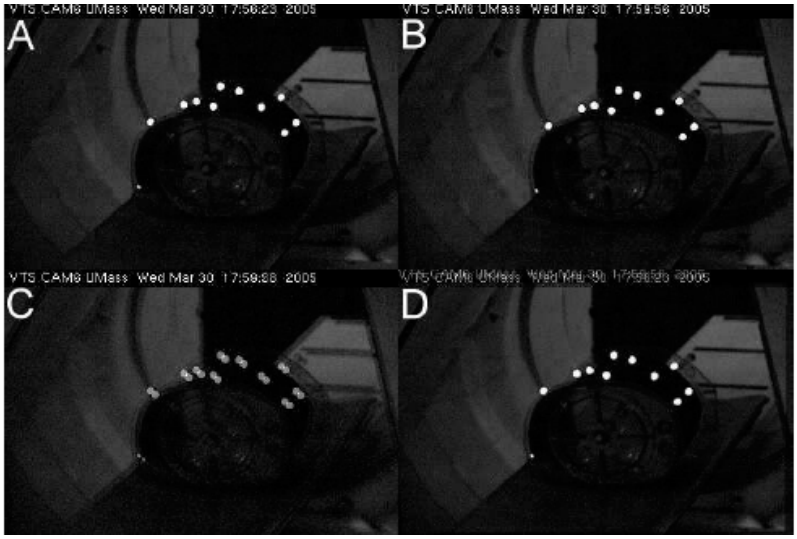

Figure 15 - A. Phantom before camera motion. PTZ parameters are saved. B. Phantom after camera was moved, rebooted and reoriented using saved PTZ parameters. C=A+B. Notice imperfect overlay. D=A+(B corrected). Overlay is almost perfect. [20]

Chapter 6
Implementation

This section details the software implementation details of the thesis. As a part of the VTS system, the design and protocols used are same as that of VTS as explained below. Initial forms of the system were written in Java and then ported into C++.

The Java work was done using Eclipse 3.0 [21] as the base. The Visual C++ coding is done using Microsoft Visual Studio.NET 2003. The code is packaged as dynamic link libraries (DLL) that can be ported into other applications. All the transfer and save operations on any form of data, including the VTS configuration data, is fixed to an XML file format. The Graphical User Interface is programmed using LabVIEW. All the work is done on the Windows OS. The Marker Matching portion needed the use of certain Computer Vision tools which were obtained from the Intel Open Source Computer Vision Library (Intel OpenCV 3.1). An overview of OpenCV and its components follows.

6.1 OpenCV

The *Open Source Computer Vision Library* (OpenCV), created and maintained by Intel, is a free, open source collection of computer vision routines geared mainly towards human-computer interaction, robotics, security, and other vision applications where the lighting and context of use cannot be controlled [22]. OpenCV provides a solid vision infrastructure and thereby allows work at a higher level rather than having to worry about the basics. It has BSD type license to promote free commercial and research use.

OpenCV support for vision is extensive. It supports routines for input, display, and storage of movies and single images [23]. The various functions supported are given below.

- Image processing
    - Convolution
    - Thresholding
    - Morphological operations
    - Flood fills
    - Histograms
    - Smoothing
    - Pyramidal sub-sampling
    - Full suite of image algebra and arithmetic
- Geometry
    - Delaunay triangulation
    - Calibration
    - Fundamental and essential matrices computation

- Image alignment
- Stereo depth calculation

♦ Feature detection

- Corner detectors
- Canny edge operators
- Blob finders
- Scale invariant features

♦ Shape descriptors

- Hu moments
- Contour processing
- Fourier descriptors
- Convex hulls
- Connected components

♦ Motion

- Optical flow
- Background learning and differencing
- Motion templates
- Motion gradients

♦ Learning-based vision

- Feature histogram comparison
- Image statistics
- Template-based correlation
- Decision trees
- Statistical boosting on up to convolution neural networks.

This library is actually a composition of 4 different partitions [24].

CV – Consists all the Computer Vision Algorithms

CVAUX – Beta Algorithms and useful gems

CXCORE – Linear Algebra and Matrix support

HIGHGUI – Media and Window handling

The Windows version comes with an installer while the Linux version needs to be unzipped and installed manually. The library can be downloaded from http://sourceforge.net/projects/opencvlibrary/ [25]. Version 3.1 was used with this project. To make it run in tandem with Visual Studio we need to indicate where the include files and the lib files are located. More on installing OpenCV on Windows and the functions used in this thesis are presented in the Appendix B.

Chapter 7
Feature Matching

When all 6 cameras take snapshots of the stretchy garment at one instance of time, we have 6 images to deal with. All of them go through software processing undergoing bitmap conversion, thresholding, blob detection stages and an XML file with the locations is written. Now we should be able to identify sets of each marker through all 6 views. This process requires a robust feature matching algorithm. This section discusses the geometrical conditions when tackling such a problem and further details of how this was implemented.

Image matching is considered a very crucial and at the same time, one of the hardest problems in stereo imaging [26]. Stereo images allow for the interpretation of an object, area, or scene from two distinct viewpoints. The different viewpoints allow for depth when matching points from corresponding images. The ideal matching situation would be to match each and every pixel in each image. However, this is infeasible, due to the fact that in this case, the intensity of a single pixel can be quite ambiguous to allow for matching with confidence. To resolve this, collections of pixels are matched; images are matched as corresponding areas and features [27].

In our case, we match features in the form of circular markers with a radius of around 5 pixels in the default zoom setting of the camera.

Figure 16 – Example Stereo Pair Images of the Torso Phantom

Considering the case of the stereo pair indicated in Figure 16, we need to match the glowing markers in both the images. This looks like a 2 Dimensional search space problem, where we need to isolate each feature in the left image and search for a very close match throughout the right image. But this has its set of limitations – the search space for each marker is the entire image making real time processing difficult and the markers are perfect circles of about the same size, so there will be multiple matches. By manipulating the geometry of the stereo system, we can reduce this to a one dimensional search problem (along a line) [8]. *The Epipolar Geometry constraint* states that given a feature point P_l in the left image, the corresponding feature point P_r in the right image must lie on the corresponding epipolar line [19]. The section below talks in detail about this.

7.1 Epipolar Geometry

Given a pair of stereo cameras, any point P in 3D space defines a plane π_P that goes through the point P and the two centers of projection of the two cameras [19]. This plane is called the Epipolar Plane. The lines where π_P intersects the image planes of the stereo cameras are considered to be the conjugated epipolar lines. The image in one camera of the other camera's projection center is called an Epipole. Thus, the geometry of a stereo system is described as Epipolar Geometry.

Vectors from the left and right image planes can describe a point P in 3D space. The vectors $P_L = [X_L, Y_L, Z_L]^T$ and $P_R = [X_R, Y_R, Z_R]^T$ refer to the same 3D point P from the left and right image planes, respectively. Furthermore, the vectors $p_L = [x_L, y_L, z_L]^T$ and $p_R = [x_R, y_R, z_R]^T$ refer to the point P in the respective image planes from the left and right projection centers, O_L and O_R.

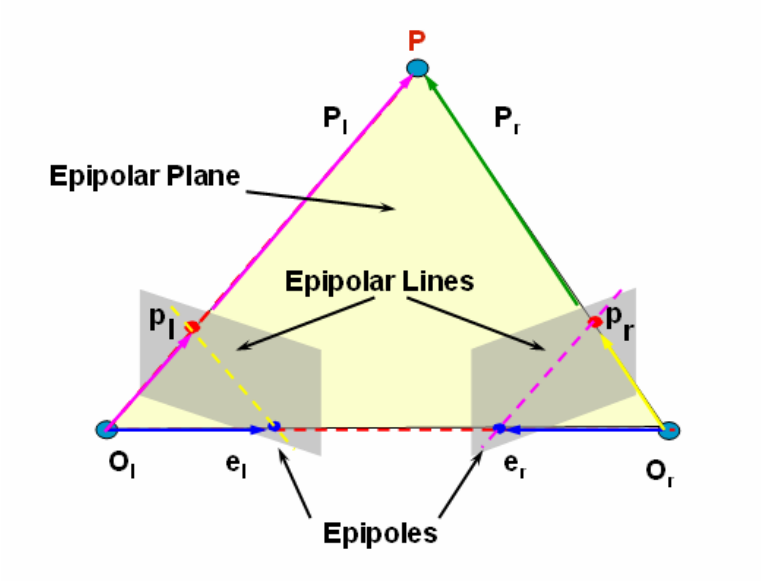

Figure 17 - Epipolar Geometry of a Stereo System [19].

The relation between points P_L and P_R is given by the equation

$P_R = R (P_L – T)$,

where R is a rotation matrix and T is a translation vector, given by $(O_L – O_R)$, in 3D space. The relation between the points p_L and P_L is given by the equation

$$p_L = \frac{f_L}{Z_L} P_L$$

where f_L is the focal length of the left camera and Z_L is the z-component of the vector P_L. A similar equation also applies to the right camera. The equation of the epipolar plane, π_P, is given by

$(R^T P_R)^T T \times P_L = 0$.

An Essential Matrix E which maps points and epipolar lines in camera coordinates is derived. This E is given by the equation

$E = RS$, where

$$S = \begin{bmatrix} 0 & -T_Z & T_Y \\ T_Z & 0 & -T_X \\ -T_Y & T_X & 0 \end{bmatrix}$$

and $p_R^T E p_L = 0$.

If we want to process in pixel coordinates, a fundamental matrix F is used which maps the points and epipolar lines, but in Pixel Coordinates. The fundamental matrix is given by the equation

$$\overline{p_R}^T F \overline{p_L} = 0.$$

7.2 Trinocular Geometry

Extending the concept of epipolar geometry into three views, the Trinocular Geometry is described. The relation between corresponding points X_1, X_2, X_3 in three views of the same scene is defined by a geometric object called a Trifocal Tensor [27]. This allows transfer of corresponding objects (points or lines) among the views. Given a point X_1 in the first view, the corresponding point in the second view must be searched for along an epipolar line l_{21} that is determined by the trifocal tensor. Moreover, if two matching points X_1, X_3 are given, the third match X_2 is the intersection of the epipolar lines l_{21} and l_{23}. The trifocal tensor can be estimated from six or more corresponding points in three views.

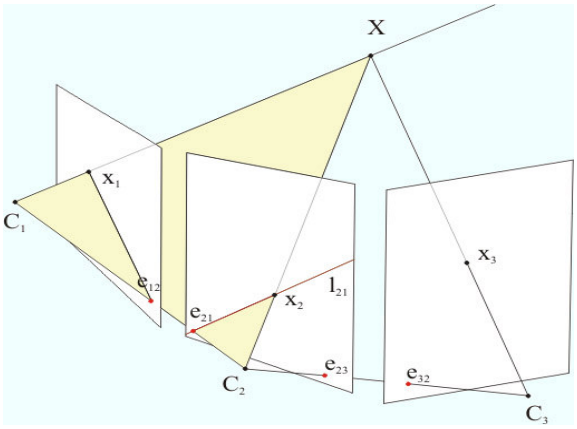

Figure 18 – Three View Geometry [28]

This thesis did not delve into the computation of the trifocal tensor but instead performed three view matching based on the principle behind it. Hence the mathematical derivation of the tensor is not presented. Further reference on this can be found at [29 - 31].

7.3 Multi-view Matching

In stereo processing, a short baseline means that the estimated distance will be less precise due to narrow triangulation. For more precise distance estimation, a longer baseline is desired. With a longer baseline, however, a larger disparity range must be searched to find a match. As a result, matching is more difficult, and there is a greater possibility of a false match. Another equally important problem is that there might be more markers hidden in the image. Using multiple cameras, giving more than one stereo pair, tackles this problem. The principle that the proper match lies where the epipolar lines from two other views intersect is extended to more than three cameras.

The algorithm projects all the possible matches found between views 1 and 2 onto view 3, and if there is the presence of any uncertainty at this point, the possible match sets between 1-2-3 are projected onto view 4. The set which has its intersection point coincide perfectly with a feature present in 4 is the right match. There exists a remote chance for ambiguity when using three views because of linear alignment of features in the images, but it is not possible when using more than three views. Thus this guarantees the best match in all ways.

7.4 Error Metrics

This section discusses the two possible error metrics that were considered for this problem. Initially the theoretical metric of finding the closest feature to the

intersection of epipolar lines was implemented. Later because of certain shortcomings that cropped up, calculating the sum of perpendicular distances from epipolar lines was used.

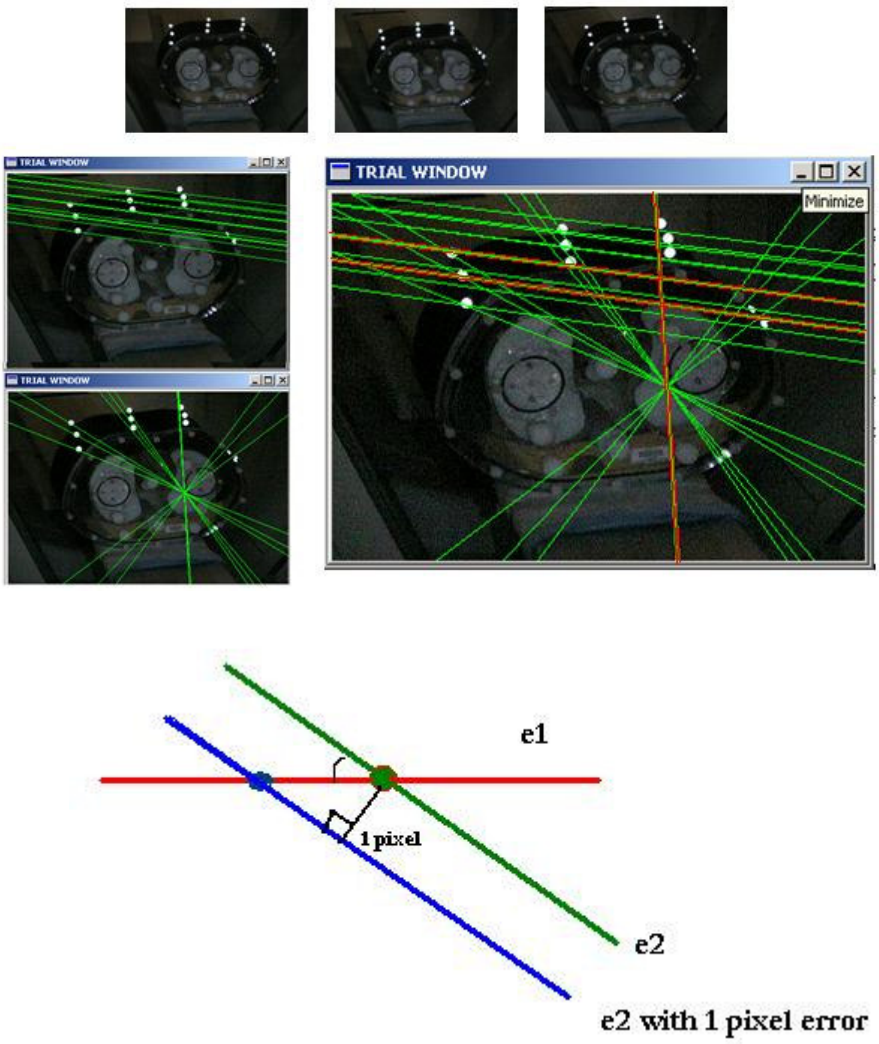

Figure 19 – Case of epipolar line overlaps (marked in red). Top – three views at a particular instant, Left – Epipolar projections 1 to 3 & 2 to 3, Right – Intersection of Epipolar lines, the overlapping lines make it hard to find the right match. Bottom – Geometry of error computation

Figure 19 shows a general case of epipolar projection from two views onto the third view. As seen in the lower right portion, there is a possibility that epipolar lines overlap (as shown marked, the red lines are actually two epipolar lines). This is due to the co-linear camera centers and possible symmetric arrangement of markers on the garment. Also due to the orientation of the cameras with respect to the SPECT table, the epipolar lines from two views projected onto the third view can sometimes have very small angle between them, making slight changes give high error values. Since this is not totally avoidable due to physical constraints on where the cameras can be placed in the clinic, there is a need to find a metric which doesn't get influenced by them.

7.4.1 Intersection Metric

For the case presented above, intersection was used as a metric to evaluate the robustness. This did not quite work well for the datasets. In case of lines with small angle between them, error as small as one pixel in drawing the line results in inverse of sine of the angle variation from the correct location (Figure 19 – Bottom). Also overlaps gave nearly same error values, thereby making it hard to distinguish the right match from the false possibilities.

7.4.2 Perpendicular Distances Metric

In order to reduce the sensitivity to the camera orientations, we choose sum of perpendicular distance to act as the metric. This guarantees that the error limits to the

minimum distance from the point, which is the right error. Overlaps still gave nearly the same errors, but since we are dealing with low deltas, it's possible to distinguish them. The logic is that overlaps always differ by a certain small value. This does not make much impact when dealing with high errors but is significant in the case of low errors. Thus the code supports sum of perpendicular distances from the epipolar lines as an alternate error metric. The centroid location which has the least value for this is considered the best match. Details follow in the algorithm description.

7.5 Algorithm

This algorithm gives a method for feature matching between three or more views. We assume that all the views show all the markers in the initial set of frames. This means that there are equal numbers of features detected in all the cameras for the case where basic matching is performed. Table 2 gives the algorithm for 4 view geometry. A pair of camera is selected randomly (views 1 and 2 are used as default, but this can be changed by user selection if needed). Epipolar lines are projected from view 1 onto view 2. For every marker in view 1 a listing of all the markers close to its corresponding epipolar line in view 2 is made. Depending on the alignment of the cameras, some of these listings may have more than one marker (This is because the centroid locations are within 1-2 pixel accuracy and co-linear camera center positioning makes epipolar lines overlap for linearly oriented markers).

Thus the advantages of having the multi cameras looking at the patient can be exploited here. The methodology below is discussed for a single marker. A third view

is considered. The marker in view 1 and each of the markers that were identified to be possible matches in view 2 are projected onto this. For each set, we find a feature with minimum perpendicular distance from these lines. Using a weighted function between errors in two and three views, the least value combination is recorded. This is repeated for all possible sets from 1 and 2. Theoretically only one of the set should match with the correct location of a feature, while other sets are expected to have high distances.

This gives us the right combination for one marker and an iterative process on this method; inculcating the already identified match as prior knowledge, all the matches are found for the set of 3 images. If the rare case of more than one possible set occurs for 3 views, it is projected onto the fourth image and only one set will match with a feature. This is then followed by the iterative identification for all the rest of markers. The extension to additional images is straightforward. The next section discusses issues relating to efficiency of this procedure and the results.

- Begin
- Pick any two camera views C1 and C2
- For every feature detected in C1, Fi = F1(C1) to Fn(C1)
  - Project the epipolar line L12(Fi) onto C2
  - For every feature in camera C2, Fj = F1(C2) to Fn(C2)
    - Δ1 = Perpendicular distance from Fj to L12(Fi)
    - If this lies within a threshold limit,
      - Add to the list of possible matches for Fi
    - End If
  - End For loop
  - If list of matches for Fi has more than one value
    - For each possible match P
      - Project P and Fi onto C3 and find epipolar lines L13(Fi) and L23(P)
      - For features Fk= F1(C3) to Fn(C3)
        - Δ2 = Perpendicular distance of Fk from L13(Fi) and L23(P)
        - Δ3 = weighted value between Δ1 and Δ2
      - End For loop
      - Fm is identified which has min Δ3 for set Fi,P
    - End For loop // performed for all possible P values
    - If unique solution found //only one set has very less delta
      - Propagating back, the best match between C1,C2,C3 is found
      - Projecting them onto C4, find feature closest to intersection
    - Else //more than one possible match sets found between C1,C2,C3
      - For each possible match set Fi, P, Fm
        - Repeat same steps as done for case with three views
        - The feature with minimum value for delta is the best match
        - Propagating back, the best match between all cameras is found
      - End For loop // Performed for all markers in C1
    - End If //Matches found in all cameras
- End For loop
- End

Table 2 – Algorithm to perform feature matching using 4 views

7.4 Results

All the results presented were run on trinocular stereo setting. Due to the fact that running even the simplest of the tests needs access to the IRIX Gamma Camera in the clinical setting and administration of radioactivity, a large number of data sets were not available. Existing two camera datasets were not of use for our purpose. Three camera datasets where all markers were not visible initially were eliminated too. The program was run on two complete sets of data, each of which had three camera views.

One of them is presented below. First we demonstrate shortcomings of the two camera method, and then proceed onto how an extra view makes the detection of matches more robust. A Calibration phantom with 7 spheres is used initially to find the fundamental Matrix. Images are then taken from the VTS using the Torso Phantom as the subject. The Stretchy garment had 9 markers placed on it. Figure 20 shows the calibration images and the torso phantom images used per camera

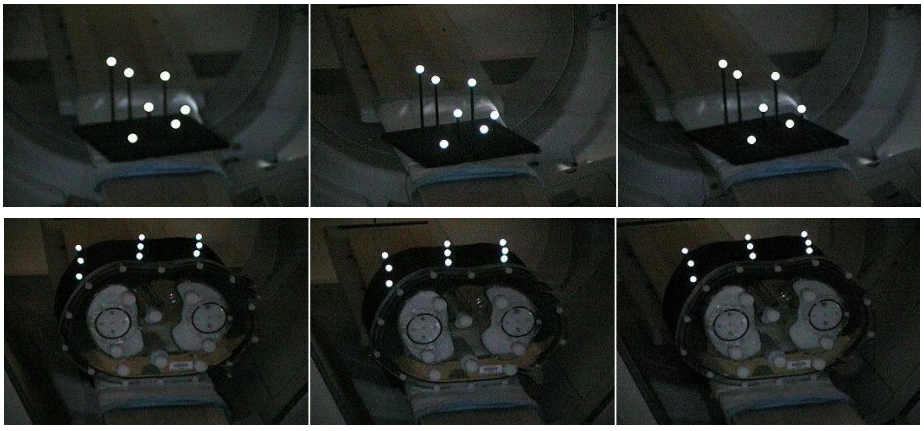

Figure 20 – Upper row shows calibration phantom and bottom row shows torso phantom from three different camera views at the same instant of time.

The Fundamental Matrices obtained from it are given below in Figure 21. Note that F is not decomposed into various components, since we could get the epipolar lines using the OpenCV library rather than manual coding. Table 3 lists the features.

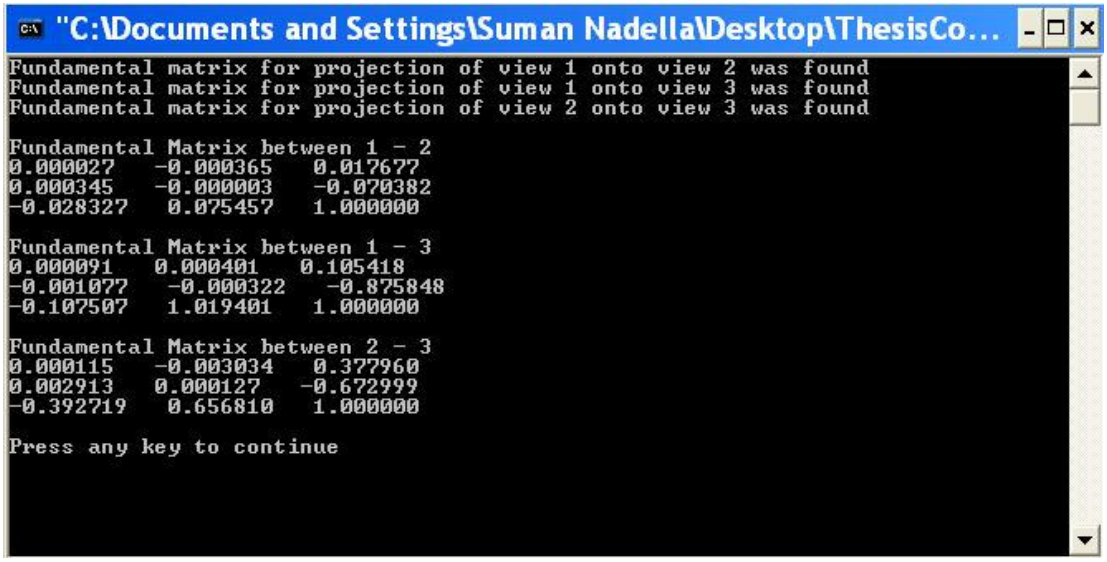

Figure 21 – The Fundamental Matrices obtained

Location	List 1 (x y)	List 2 (x y)	List 3 (x y)
0	86.00 34.00	85.00 44.00	79.00 38.00
1	159.00 22.00	157.00 30.00	152.00 23.00
2	225.00 21.00	222.00 29.00	217.00 19.00
3	88.00 49.00	90.00 58.00	86.00 53.00
4	158.00 32.00	158.00 41.00	154.00 33.00
5	225.00 31.00	224.00 38.00	221.00 29.00
6	86.00 67.00	90.00 76.00	88.00 71.00
7	156.00 44.00	159.00 52.00	157.00 44.00
8	222.00 40.00	223.00 47.00	223.00 38.00

Table 3 – Centroid locations of features detected from torso images.

The feature centroids were extracted into list1, list2 and list3. For demonstration purposes, they are put in the right order such that 0^{th} location in the list 1 should match 0^{th} location in list 2 and 3, so on. Running this on the first two cameras, the resultant list of possible matches is obtained. Note that the ambiguous cases have similar error values, thus needing another view to select between them.

As figure 22 shows, for feature 0, the resultant matches 0 and 7 both have almost identical error values. Likewise for feature 8, the wrong match of 5 has less error (0.9) compared to the right match 8 (4.03). The epipolar line projection for feature 8 of camera 1 onto camera 2 is illustrated in Figure 23 with highlighted error area. While the green line should be a perfect match to the green blob, it is closer to the red blob. This illustrates the potential difficulties of matching using only 2 views.

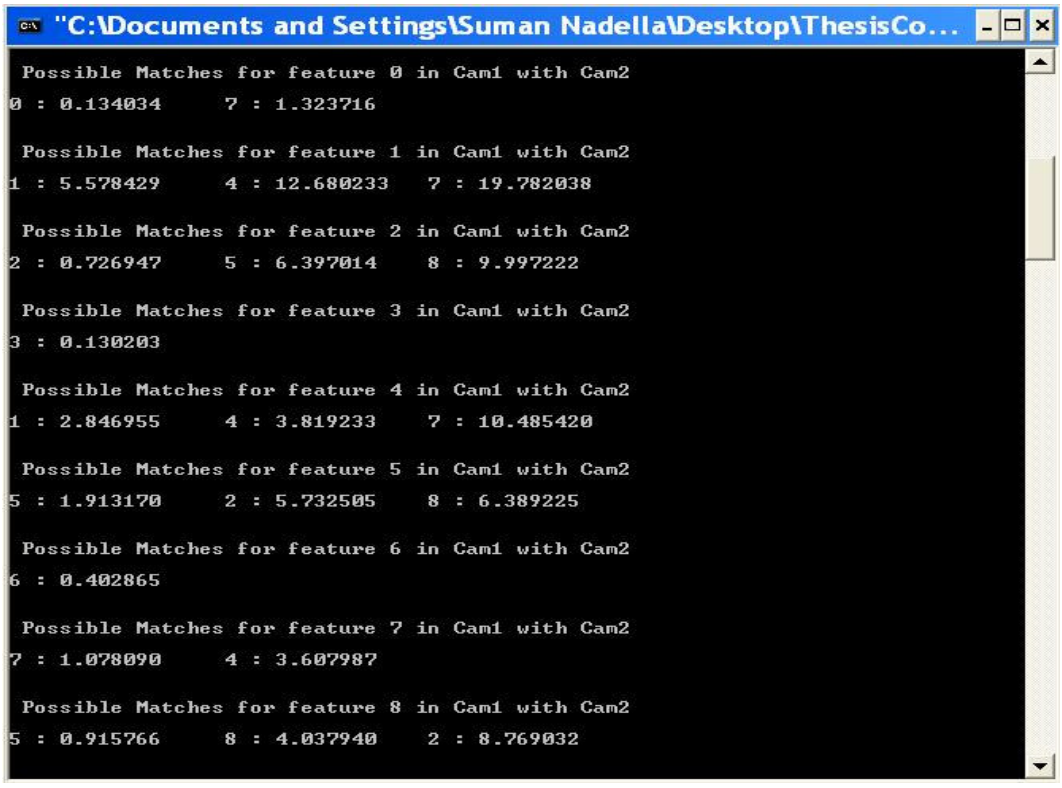

Figure 22 – Results of 2 camera matching with error values printed beside the feature.

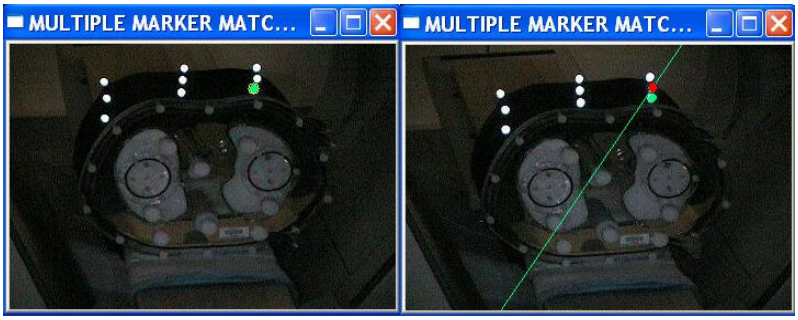

Figure 23 – Epipolar line projection showing possible wrong match

Bringing the third view in, and computing the error for all the combinations from the previous stage, the results are obtained.

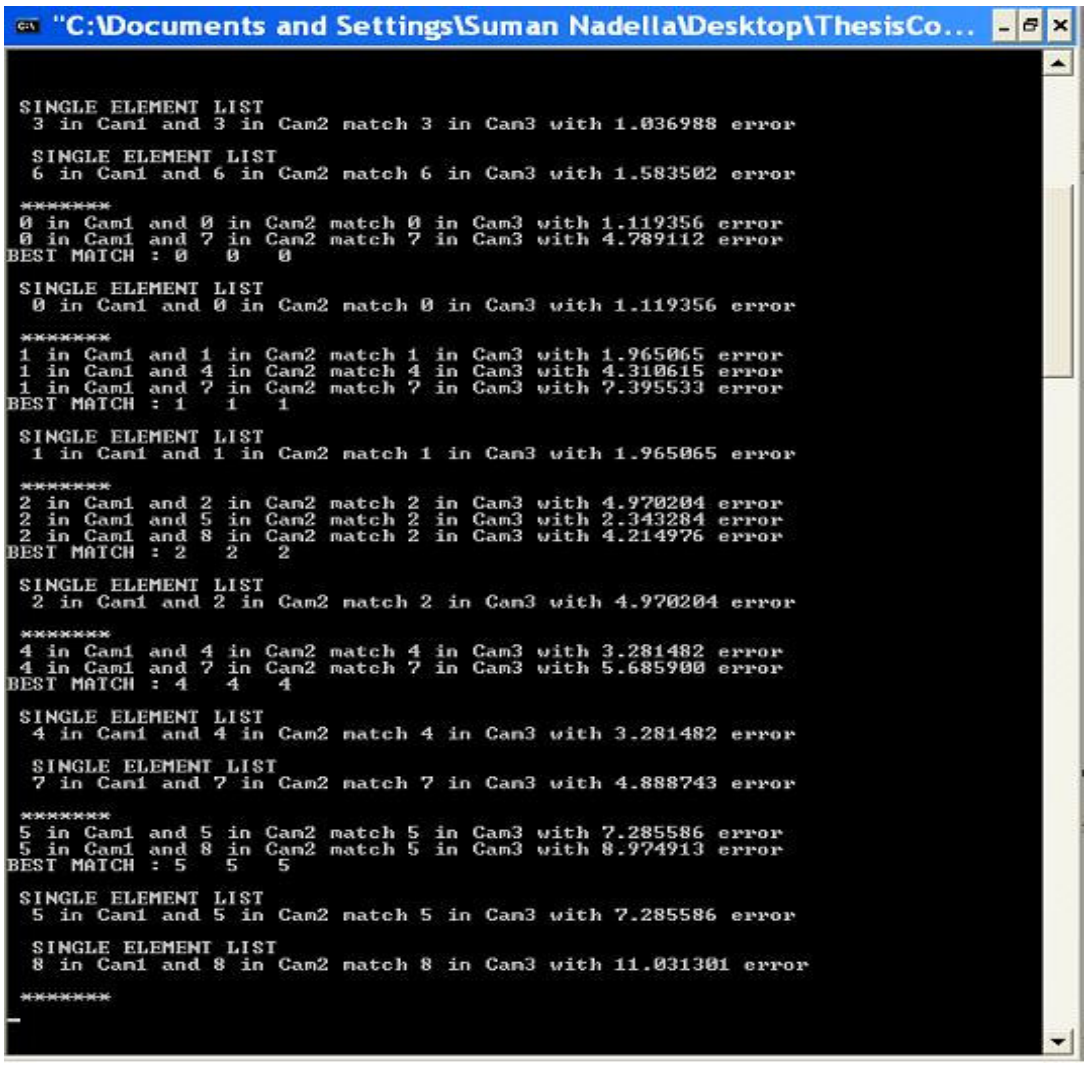

Figure 24 – Elimination process of the possibilities

As seen in Figure 24, the test cases of feature 0 and 8 are solved. In the case of feature 0, the error in the correct match is far smaller than the other possibilities. When propagating the matches, feature 5 is already matched, so the presence of it as an option for 8 doesn't exist, thus giving the right value. Features 3 and 6 are correctly matched in the two view case itself. Using this knowledge and removing all the already matched feature points from the remaining lists, we attain the right result.

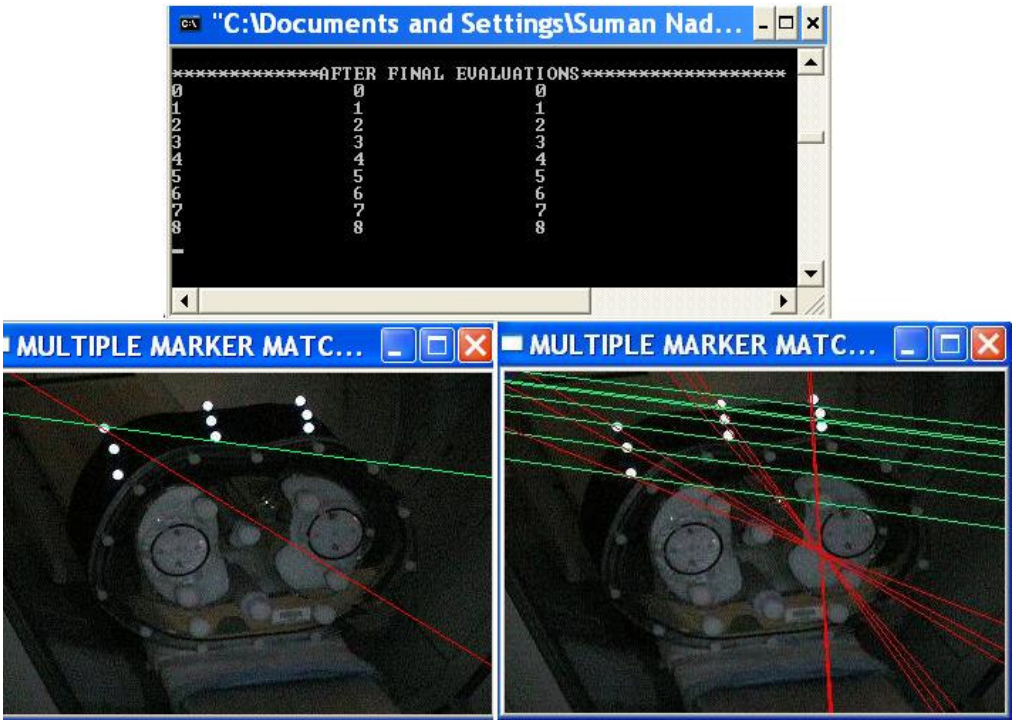

Figure 25 – Top - Final Evaluation Results, Left - Intersecting epipolar lines for feature 0, Right - Visualization of all markers of 1 and 2 projected onto 3

7.5 Robustness

The robustness of the algorithm is measured by the accuracy in matching. Given the initial constraint that all the markers are visible in all the cameras, the

algorithm illustrates perfect results for the test cases. Any remaining ambiguity can be resolved using the fourth view, though results involving it are not presented here. The average error values range from around 0.05 to 7 pixel^2. This is considered sufficient since this value doesn't affect the stereo computation in any way and acts as just a metric for choosing a proper set. The average difference in error between the best case and the next best case is around 4 – 13 pixel^2 per pair of cameras, resulting in good confidence in the procedure.

7.6 Special Cases

The presence of Partial or Missing markers is the biggest special case encountered. Markers can be blocked partially or completely by another depending on the way they are arranged and seen. Also markers can go out of the Field of View of at least one camera. The matching algorithm doesn't give correct results when the total number of markers found in each camera is not the same. In order to tackle this, an initial constraint is imposed so that all the markers are visible in the first frame through all images. Once each marker has all of its remaining counterparts present in the list, matching can proceed. There after, a method to isolate the possible movement zone of it can be implemented, so that even when a blob becomes partially visible, we can monitor its origin and thereby match it. More on this radius tracking is described in chapter 9, Future Work.

Chapter 8
Stereo Computation

Once all the marker matches have been identified in all cameras, for a particular frame, the most probable object position for it is determined by minimizing the sum of square errors using the maximum likelihood method [32]. The minimum error condition is calculated using a nonlinear optimization method. This way, we are taking the anisotropic errors into account without the need for a linear approximation [32]. This effectively produces the actual physical location of markers on the patient with respect to the real world coordinate system.

8.1 Algorithm

Since we know the camera Intrinsic and Extrinsic parameters, a procedure called reconstruction by triangulation can be applied. Rays are projected back into space from the left and right image points; they do not necessarily intersect due to errors induced by noise. So, the point where the distance between the two rays is a minimum is the estimation to the point P in space. This will be the midpoint of the perpendicular line that can be drawn joining these two rays. When there are more

than 2 views, the number of rays increases, and the point with minimum distance from all of them is estimated. The algorithm is given below in Table 4. Figure 26 shows general stereo computation model.

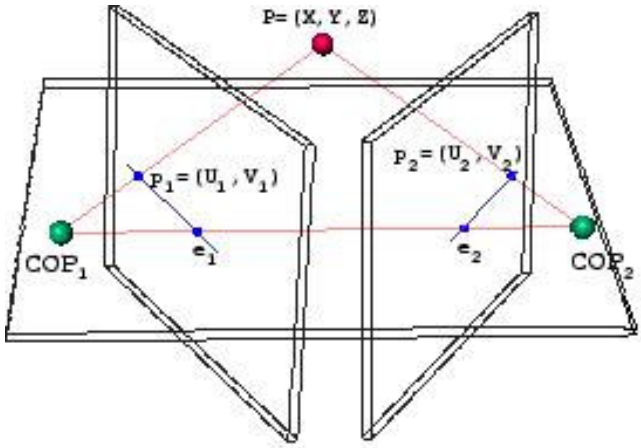

Figure 26 – Stereo Computation with Triangulation [39]

- Begin
- For each marker Mi = M1 to Mk
  - For each camera view j = 1 to n
    - Project the world lines Lj = (Xj, dj) and define Pj on Lj as Xj + sjdj
    - Assuming a point Pi as the 3D location of this marker, find sj such that it minimizes the distance between Lj and Pi
    - Insert sj into all the Lj equations to get P'i, point on Lj closest to Pi
    - Minimize the SSD of the distance from hypothetical Pi to all the lines Lj
    - An optimal value for Pi coordinates is obtained
  - End For loop
- End For loop
- End

Table 4 – Algorithm for Stereo Computation using Multiple Views

Going into the mathematical details, considering the procedure for one marker with n view locations in the image plane, call the 3D point to be estimated as P.

Each ray is projected as Lj = (Xj, dj) and the point P is defined as P = Xj + sjdj where dj represents the unit direction vector, Xj represents the start of the line, j indicates the view, such that Xj ⊥ dj.

For each projected line, find s such that it minimizes the distance between Lj and P.

$$\underset{s}{Min} \| \vec{X} + s\vec{d} - \vec{P} \|^2$$

Differentiating with respect to s,

$$\frac{\partial}{\partial s}\left(\vec{X} + sd - \vec{P} \right)^2 = 0$$

$$2 * \left(\vec{X} + s\vec{d} - \vec{P} \right) * \vec{d} = 0$$

$$s = \left((\vec{P} - \vec{X}) \bullet \vec{d} \right) / \| \vec{d} \|^2$$

Since X ⊥ d, the dot product becomes zero. Solving for s, it thus becomes

$$s = \left(\vec{P} \bullet \vec{d} \right) / \| \vec{d} \|^2$$

Since now the value of s is known, substituting this is in the above equations define P'j as the point on Lj closest to P.

$$P^{\iota} = \vec{X} + \left((\vec{P} \bullet \vec{d}) / \| \vec{d} \|^2 \right) \vec{d}$$

Now minimizing the distance between P and P'j on Lj with respect to P,

$$\underset{P}{Min} \sum_j Dist\left(\vec{P}, \vec{L}_j \right)$$

which implies

$$\underset{p}{Min} \sum_j (\vec{P} - \vec{P}j^{\iota})$$

In order to minimize this expression, differentiating with respect to P,

$$\frac{d}{dP} \sum_j \| \vec{P} - \vec{X}j - \vec{P} \bullet \vec{d}_j \, d_j \vec{} \|^2 = 0$$

$$\frac{d}{dP} \sum_j \| I \vec{P} - \vec{d} \, j \vec{dj}^T \vec{P} - \vec{X} \, j \|^2 = 0$$

Taking $A = \left(I - \vec{d} \, j \vec{d}^T \, j \right)$ and $\vec{b} = \vec{X}_j$

$$\frac{d}{dP} \sum_j \| A \vec{P} - \vec{b} \|^2 = 0$$

$$\sum 2 * A^T (A\vec{P} - \vec{b}) = 0$$

Solving for P,

$$\vec{P} = \left(\sum_j A^T A \right)^{-1} \sum_j A^T \vec{b}$$

The error metric used here is thus the sum of squared differences of distance of world lines projected back from the location P.

8.2 Results

This section shows the results from the application of the stereo algorithm on two and three views of the phantom. The data used here is the calibration data, so that we can compare it against the actual world points which we already know. Projecting the image points matched in the views, we should be able to get world points which are in coordination with the SPECT blobs detected.

The image point and world point datasets are given below in Table 5. Using the left and right camera data, and running through the algorithm explained above, the results obtained are in Table 6. These values have very good correspondence to the expected world point values. Due to the lack of third camera data, one of the data sets was replicated and ran through the algorithm as a test case when using three cameras. The results are shown in Table 7.

Left Camera -	
397.5393	216.3102
288.7942	37.1857
238.0532	299.6868
220.0982	397.6539
342.9025	352.7877
379.1498	300.2672
493.6267	87.2075
Right Camera –	
245.2291	231.0073
109.5918	91.9847
251.3399	378.2990
372.0693	438.3803
382.9129	335.0588
339.4298	287.7620
332.4588	43.6162
World Points –	
62.9475	84.1305 93.1065
85.7475	26.1863 78.8210
80.9624	80.8171 36.8197
65.2208	82.7305 9.7733
48.6138	85.9099 37.2383
50.5062	86.1225 57.9189
30.6109	35.3974 84.5358

Table 5 – Image and World Point Data Sets for Stereo

62.7863	83.7852	92.7078
85.5566	26.1372	78.9858
80.6661	80.2685	36.2584
64.8774	82.3670	9.2953
48.7491	85.2790	36.4256
50.1795	85.4736	57.5204
30.5257	35.2494	84.6647

Table 6 – Stereo Computation results for 2 camera views

62.7870	83.7536	92.7307
85.5536	26.0632	79.0181
80.6693	80.2319	36.2958
64.8836	82.3286	9.3471
48.7506	85.2639	36.4420
50.1805	85.4574	57.5353
30.5255	35.2453	84.6666

Table 7 – Stereo Computation results for 3 camera views

It can be seen that these values lie within a maximum of +/- 1 mm accuracy. Since we are dealing with 3D locations and these values are used for reconstructing the SPECT slice data, this should be close to perfect and the accuracy achieved is almost ideal since 1 mm translates into around $1/4^{th}$ of a pixel. The average RMS accuracy comes up to 0.3415 cm.

.

Chapter 9
Future Work

The present implementation of VTS for feature matching and stereo computation has been shown to be robust within a few limitations. The immediate future work to be done would be elimination of these limitations. Matching has the working premise that all the markers are viewable in all cameras for the first frame. This should be modified so that there is a frame tracking module which traverses all the incoming frames and starts processing only when all the markers are visible. Further work on computing the centroid of partial blobs would be useful to eliminate the initial 'all markers fully visible' constraint. A procedure to identify the centroid of the partial blobs depending on their shape and curvature is to be identified and implemented. Performing feature matching on each frame grabbed at a frame rate up to 30 FPS utilizes a lot of resources and makes real-time processing difficult. Thus we should store the initial positions and matches extracted from the first frameset and perform tracking around a limited search space. This space will be identified by using extensive sample data for different possible motions. Considering the velocity of motion per frame, the feasible motion for various non rigid movements such as

sneezing, coughing and turning should be taken from sample patients and an estimate prepared identifying the maximum possible range of marker motion. Once the radius is identified, we will have a domain to search for each marker in the next frame without running the algorithm on all the features again. A possible question is how much motion can be allowed for reconstruction to be robust. If the patient moves above this threshold, then there will be a need for a total new acquisition. From the sample data, the upper limit on the patient motion, above which there needs to be a reacquisition needs to be identified.

Chapter 10
Conclusion

This thesis successfully implemented algorithms for feature matching and stereo location computation for multi-view geometry. It fits into the Visual Tracking System to accurately compute 3 dimensional locations of markers on patients. The matching algorithm gives correct feature matches along multiple views (tested with four views) within certain initial assumptions. Stereo computation generates the coordinates in the SPECT world, by using the matches identified for each marker along all the views with an accuracy of +/- 1 mm. This gives a surface estimate which is recomputed when the patient moves. The difference estimate is used for correction of the tomographic reconstruction by rebinning the raw gamma camera data to undo the motion prior to reconstruction.

Appendix A

Singular Value Decomposition (SVD) is a very powerful set of techniques dealing with sets of equations or matrices that are either singular or numerically very close to singular. A matrix is called singular if it does not have an inverse, that is, if and only if its determinant is zero. SVD allows one to diagnose the problems in a given matrix and provides numerical answer as well. This section explains how SVD would help in optimization and the actual mathematics behind it [35].

For any given matrix A ϵ $R^{m \times n}$ there exists a decomposition

$A = UDV^T$ such that

U is an m × n matrix with orthogonal columns

D is a n × n diagonal matrix with non-negative entries

V^T is an n × n orthogonal matrix

$$\begin{bmatrix} A \end{bmatrix} = \begin{bmatrix} U \end{bmatrix} \begin{bmatrix} D \end{bmatrix} \begin{bmatrix} V^T \end{bmatrix}$$

The diagonal values of D are called Singular Values of A

The column vectors of U are the Left Singular Vectors of A

The column vectors of V are the Right Singular Vectors of A.

The SVD can be performed such that the diagonal values of D are descending i.e.

$$d_1 \geq d_2 \geq ... \geq d_n \geq 0$$

The diagonal values of D are the square roots of the Eigenvalues of A^TA and AA^T (Thus there is non-negativity of the elements of D). A number is called an eigenvalue of a matrix if there exists a nonzero vector such that the matrix times the vector is equal to the same vector multiplied by the eigenvalue [36]. This vector is then called the Eigenvector associated with the eigenvalue. If,

$$A\,\mathbf{x} = \lambda\,\mathbf{x}$$

For some scalar λ, then λ is called an eigenvalue of A, and **x** is said to be an eigenvector of A corresponding to λ.

It holds for the left singular vectors u_i: $A^TAu_i = d_i^2u_i$

And for right singular vectors v_i: $AA^T v_i = d_i^2v_i$

That is, the left singular vectors u_i are eigenvectors of A^TA and the right singular vectors v_i are eigenvectors of AA^T.

SVD can explicitly construct orthonormal bases for the null-space and the range of a matrix. Columns of U corresponding to non-zero elements of D span the range and Columns of V corresponding to zero elements of D span the null-space.

SVD can be used for *linear optimization* by using the following property -

Let v_n be the right singular vector corresponding to d_n (the smallest element of D)

The product Ax with $||x||_2 = 1$ has the minimal value for $x = v_n$.

The minimizing property of the last right singular vector v_n can be used to solve the following minimization task –

Given the linear function f = Ax, f: R^n -> R^m to be minimized with the constraint that the solution x is not trivial (x != 0), it can be shown that the solution is the right singular vector x = v_n corresponding to the smallest singular value d_n.

The proof is not given here, but further references can be found at [37].

EXAMPLE SVD COMPUTATION

To understand how to solve for SVD, let's take the example of a matrix [38].

$$A = \begin{bmatrix} 2 & 4 \\ 1 & 3 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

In this example the matrix is a 4x2 matrix. As previously stated, the eigenvectors of AA^T make up the columns of U

$$A.A^T = \begin{bmatrix} 2 & 4 \\ 1 & 3 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} . \begin{bmatrix} 2 & 4 & 0 & 0 \\ 1 & 3 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 20 & 14 & 0 & 0 \\ 14 & 10 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} = W$$

Now that we have a n x n matrix we can determine the eigenvalues of the matrix W.

Since W $\mathbf{x} = \lambda \mathbf{x}$ then (W- λI) $\mathbf{x} = 0$

$$\begin{bmatrix} 20-\lambda & 14 & 0 & 0 \\ 14 & 10-\lambda & 0 & 0 \\ 0 & 0 & -\lambda & 0 \\ 0 & 0 & 0 & -\lambda \end{bmatrix} \mathbf{x} = (W - \lambda I)\mathbf{x} = 0$$

For a unique set of eigenvalues, determinant of the matrix (W-λI) must be equal to zero. Thus from the solution of the characteristic equation, $|W-\lambda I|=0$.

Solving this and substituting, we obtain the following equations:

$19.883\ x1 + 14\ x2 = 0$

$14\ x1 + 9.883\ x2 = 0$

$x3 = 0$

$x4 = 0$

Upon simplifying the first two equations we obtain a ratio which relates the value of x1 to x2. The values of x1 and x2 are chosen such that the elements of the D are the square roots of the eigenvalues.

Thus a solution satisfying the above equation is,

$x1 = -0.58$ and $x2 = 0.82$ and $x3 = x4 = 0$ (this is the second column of the U)

Substituting the other eigenvalue we obtain:

$-9.883\ x1 + 14\ x2 = 0$

$14\ x1 - 19.883\ x2 = 0$

$x3 = 0$

$x4 = 0$

A solution that satisfies this set of equations is,

$x1 = 0.82$ and $x2 = -0.58$ and $x3 = x4 = 0$ (this is the first column of the U).

Combining these we obtain:

$$U = \begin{bmatrix} 0.82 & -0.58 & 0 & 0 \\ 0.58 & 0.82 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Similarly A^TA makes up the columns of V so we can do a similar analysis to find the value of V.

$$A^T.A = \begin{bmatrix} 2 & 4 & 0 & 0 \\ 1 & 3 & 0 & 0 \end{bmatrix} \begin{bmatrix} 2 & 4 \\ 1 & 3 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

and similarly we obtain the expression:

$$V = \begin{bmatrix} 0.40 & -0.91 \\ 0.91 & 0.40 \end{bmatrix}$$

Finally as mentioned previously the S is the square root of the eigenvalues from AA^T or A^TA. and can be obtained directly giving us:

$$S = \begin{bmatrix} 5.47 & 0 \\ 0 & 0.37 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Note that: $\sigma 1 > \sigma 2 > \sigma 3 > \ldots$

Appendix B

Installation procedure of Intel OpenCV library and the Syntax for important function calls used in the code for this thesis are presented.

Installing OpenCV

Installing OpenCV on Windows is pretty straight forward. Download executable installation from SourceForge [25] and run it. It installs OpenCV, registers DirectShow filters and does other post-installation procedures. After that you may start using OpenCV [33]. Also, it is possible to build core OpenCV binaries manually from source distribution for Linux (though, executable installation includes sources as well). Download and unpack the OpenCV-*.tar.gz package somewhere, e.g. C:\MySoft\ (the root folder is referred further as <opencv_root>).

Add <opencv_root>\bin to the system path. Under Windows 9x/ME it is done by mofifying autoexec.bat. Under NT/2000/XP it can be done by instantly at MyComputer--right button click-->Properties->Advanced->Environment Variables. Highgui requires graphic libraries by default, so remove HAVE_JPEG, HAVE_TIFF

and HAVE_PNG from preprocessor definitions and libjpeg.lib, libtiff.lib, libpng.lib and zlib.lib from linker command line. The resultant Highgui will be able to read & write most of jpeg's, bmp's, uncompressed tiff's, pxm's and sun raster images; capture video from AVI or camera via VFW and write AVIs via VFW.

To build OpenCV from sources, you need to have some C/C++ compiler. Microsoft Visual C++ 6.0 or higher is a preferred variant, because most of the demos are written for it, using MFC.

If you are going to build DirectShow filters, acquire and setup DirectX SDK as described in <opencv_root>\docs\faq.htm.

If you are going to build MIL-enabled version highgui, setup MIL include and library paths in Developer Studio.

If you are going to build MATLAB wrappers, you need to have MATLAB C/C++ interface libraries and setup Developer Studio properly.

If you are going to build TCL\TK demo applications (including with the source package only), you will need TCL\TK and BWidgets. The easiest way to obtain both is to download and install ActiveTcl from http://www.activestate.com.

Open <opencv_root>_dsw\opencv.dsw.

Choose from menu Build->Batch Build->Build. Because the produced binaries should be compatible with Visual C++, you can then use the DLLs with VisualC++ -

build applications etc. Some applications need HighguiD.lib and cvD.lib, the debug versions of the library. To solve this, under the Project menu, select Settings. Then select the Link tab and change the library to use to cv.lib and highgui.lib. Or else, build debug versions -- Under the build menu select "Set Active Configuration", click on "Debug" and the build all to build a debug version of cv.lib named cvd.lib. Repeat this for highgui. Another method would be to rename cv.lib and highgui.lib to cvd.lib and highguid.lib. This wouldn't enable debugging though.

Inoder to test the OpenCV Binaries,

Run samples at <opencv_root>\samples\c. (Note: some of the demos need AVI or Camera, e.g. motempl.c) or run algorithmic tests: <opencv_root>\bin\cvtest.exe. It will produce cvtest.sum and cvtest.lst. cvtest.sum should contain all OK's.

Syntax

OpenCV library calls for computation of Fundamental Matrix, Epipolar lines are among the foremost of the functions used. Other simple calls utilized include input/output of images, matrices, visualization etc. The syntax and functioning of the major components follows [34].

BASIC STRUCTURES

CvPoint2D32f - 2D point with floating-point coordinates

```
typedef struct CvPoint2D32f
{
   float x; /* x-coordinate, usually zero-based */
   float y; /* y-coordinate, usually zero-based */
}
CvPoint2D32f;
```

CvMat - Multi-channel matrix

```
typedef struct CvMat
{
   int type; /* CvMat signature , element type and flags */
   int step; /* full row length in bytes */
   int* refcount; /* underlying data reference counter */
   union { …. } /* data type selector */
} CvMat;
```

IplImage – IPL (Intel Image Processing Library) image header

```
typedef struct _IplImage
{
   int  nSize;             /* sizeof(IplImage) */
   int  ID;                /* version (=0)*/
   int  nChannels;         /* Most of OpenCV functions support 1,2,3or 4 */
   int  alphaChannel;      /* ignored by OpenCV */
   int  depth;             /* pixel depth in bits */
   char colorModel[4];     /* ignored by OpenCV */
   char channelSeq[4];     /* ditto */
   int  dataOrder;         /* 0 - interleaved color channels, 1 - separate color
                           channels. cvCreateImage can only create interleaved */
   int  origin;            /* 0 - top-left origin, 1 - bottom-left */
   int  align;             /* Alignment of image rows (4 or 8) */
   int  width;             /* image width in pixels */
   int  height;            /* image height in pixels */
   struct _IplROI *roi;            /* image ROI */
   struct _IplImage *maskROI;      /* must be NULL in OpenCV */
   void  *imageId;                 /* ditto */
   struct _IplTileInfo *tileInfo;  /* ditto */
   int  imageSize;                 /* image data size in bytes
                                   (=image->height*image->widthStep */
   char *imageData;                /* pointer to aligned image data */
   int  widthStep;                 /* size of aligned image row in bytes */
   int  BorderMode[4];             /* border completion mode, ignored */
```

```
    int  BorderConst[4];            /* ditto */
    char *imageDataOrigin;          /* pointer to a very origin of image data*/
}
IplImage;
```

cvSize

pixel-accurate size of a rectangle

```
typedef struct CvSize
{
    int width; /* width of the rectangle */
    int height; /* height of the rectangle */
}
CvSize;
```

PROCESSING FUNCTIONS

cvCreateMat

Creates new matrix

CvMat* cvCreateMat(int rows, int cols, int type);

rows - Number of rows in the matrix.
cols - Number of columns in the matrix.
type - Type of the matrix elements.

cvSetReal2D

Change the particular array element

void cvSetReal2D(CvArr* arr, int idx0, int idx1, double new_value);

arr - Input array.
idx0 - The first zero-based component of the element index
idx1 - The second zero-based component of the element index
new_value - The assigned value

cvmSet

Return the particular element of single-channel floating-point matrix

void cvmSet(CvMat* mat, int row, int col, double value);

mat - The matrix.
row - The zero-based index of row.
col - The zero-based index of column.
Value - The new value of the matrix element

cvmGet

Return the particular element of single-channel floating-point matrix

double cvmGet(const CvMat* mat, int row, int col);

mat - Input matrix.
row - The zero-based index of row.
col - The zero-based index of column.

cvCrossProduct

Calculates cross product of two 3D vectors

void cvCrossProduct(const CvArr* A, const CvArr* B, CvArr* C);

A - The first source vector.
B - The second source vector.
C - The destination vector.

cvFindFundamentalMat

Calculates fundamental matrix from corresponding points in two images

int cvFindFundamentalMat(CvMat* points1, CvMat* points2, CvMat* fundMatr, int method, double param1, double param2, CvMat* status=0);

points1 - Array of the first image points of 2xN/Nx2 or 3xN/Nx3 size (N is number of points). The point coordinates should be floating-point .
points2 - Array of the second image points of the same size and format as points1
fundMatr - The output fundamental matrix or matrices. Size 3x3 or 9x3 (7-point method can returns up to 3 matrices).

method - Method for computing fundamental matrix
CV_FM_7POINT - for 7-point algorithm. Number of points == 7
CV_FM_8POINT - for 8-point algorithm. Number of points >= 8
CV_FM_RANSAC - for RANSAC algorithm. Number of points >= 8
CV_FM_LMEDS - for LMedS algorithm. Number of points >= 8
param1 - The parameter is used for RANSAC or LMedS methods only. It is the maximum distance from point to epipolar line, beyound which the point is considered bad and is not considered in further calculations. Usually it is set to 0.5 or 1.0.
param2 - The parameter is used for RANSAC or LMedS methods only. It denotes the desirable level of confidense the matrix is the correct (up to some precision). It can be set to 0.99 for example.
status - Array of N elements, every element of which is set to 1 if the point was not rejected during the computation, 0 otherwise. The array is computed only in RANSAC and LMedS methods. For other methods it is set to all 1's.

cvComputeCorrespondEpilines

For every input point on one of image computes the corresponding epiline on the other image.

void cvComputeCorrespondEpilines(const CvMat* points, int pointImageID, CvMat* fundMatr, CvMat* corrLines);

points - The input points: 2xN or 3xN array (N number of points)
pointImageID - Image ID there are points are located, 1 or 2
fundMatr - Fundamental matrix
corrLines - Computed epilines, 3xN array

VISUALIZATION FUNCTIONS

cvNamedWindow

Creates a window (image placeholder)

int cvNamedWindow(const char* name, unsigned long flags);

name - Name of the window which is used as window identifier and appears in the window caption.
flags - Defines window properties. Currently the only supported property is ability to automatically change the window size to fit the image being hold by the window. Use CV_WINDOW_AUTOSIZE for enabling the automatical resizing or 0 otherwise.

cvLine

Draws simple or thick line segment

void cvLine(CvArr* img, CvPoint pt1, CvPoint pt2, double color, int thickness=1, int connectivity=8);

img - The image.
pt1 - First point of the line segment.
pt2 - Second point of the line segment.
color - Line color (RGB) or brightness (grayscale image).
thickness - Line thickness.
connectivity - Line connectivity, 8 (by default) or 4. It is possible to pass 0 for 8.

cvCircle

Draws simple, thick or filled circle

void cvCircle(CvArr* img, CvPoint center, int radius, double color, int thickness=1);

img - Image where the line is drawn.
center - Center of the circle.
radius - Radius of the circle.
color - Circle color (RGB) or brightness (grayscale image).
thickness - Thickness of the circle outline if positive, otherwise indicates that a filled circle has to be drawn.

cvShowImage

Shows an Image

void cvShowImage(const char* name, const CvArr* image);

name - Name of the window to attach the image to.
Image - Image to be shown.

cvLoadImage

Loads an image from file

IplImage* cvLoadImage(const char* filename, int iscolor CV_DEFAULT(1));

filename - Name of file to be loaded.
iscolor - If >0, the loaded image will always have 3 channels;
if 0, the loaded image will always have 1 channel;
if <0, the loaded image will be loaded as is (number of channels depends on the file).

cvWaitKey

Waits for pressed key

int cvWaitKey(int delay CV_DEFAULT(0));

delay - Delay in milliseconds.

Bibliography

1. M.A. Gennert, J.K. Ho, A.C. Quina, J.H. Wang, P.P. Bruyant, M.A. King. *Feasibility of tracking patient respiration during cardiac SPECT imaging using stereo optical cameras*. Medical Imaging Conference IEEE 2003-Portland OR-Abstract M14-291.
2. M.A. Gennert, P.P. Bruyant, M.V. Narayanan, and M.A. King. *Assessing a system to detect patient motion in SPECT imaging using stereo optical cameras.* NSS-IEEE 2002 Conf. Record, vol. 3, pp. 1567- 1570.
3. P.P. Bruyant, M.A. Gennert, G.C. Speckert, R.D. Beach, J.D. Morgenstern, N. Arora, S. Nadella and M.A. King. A *robust visual tracking system for motion detection in SPECT: Improved design and validation against the Polaris infra-red tracking system*. Medical Imaging Conference IEEE 2004 Rome, Italy. Accepted.
4. J.D. Morgenstern, M.A. Gennert, S. Nadella, N. Arora, G.C. Speckert, P.P. Bruyant, M.A. King. *A real time multi-threaded system to detect patient motion in SPECT imaging using multiple optical cameras*. To be published in *Proc. Nucl. Sci. Symposium* 2004. IEEE
5. P.P. Bruyant. *Analytic and iterative reconstruction algorithms in SPECT*. J Nucl Med 2002 ; 43 : 1343-1358.

6. http://www.physics.uba.ca/~mirg/home/tutorial/tutorial.html, Vancouver Hospital and Health Sciences Center Medical Imaging Research Group. Accessed August 07, 2005.
7. http://info.med.yale.edu/intmed/cardio/imaging/techniques/spect_camera/ Yale University section of Cardiovascular Medicine. Accessed August 07, 2005.
8. M.A.Gennert, ISRG WPI Talk on Motion Detection and Correction in SPECT Imaging – 9th January 2004.
9. N. Matsumoto, D. S. Berman, P. B. Kavanagh, J. Gerlach, S. W. Hayes, H. C. Lewin et al., *Quantitative assessment of motion artifacts and validation of a new motion-correction program for myocardial perfusion SPECT*, JNM , vol. 42, no. 5, pp. 687-694.
10. E. H. Botvinick, Y. Y. Zhu, W. J. O'Connell, and M. W. Dae, *A quantitative assessment of patient motion and its effect on myocardial perfusion SPECT images*, J.Nucl.Med, vol. 34, no. 2, pp. 303-310.
11. J. A. Cooper, P. H. Neumann, and B. K. McCandless, *Effect of patient motion on tomographic myocardial perfusion imaging*, J.Nucl.Med, vol. 33, no. 8, pp. 1566-1571.
12. P. H. Pretorius and M. A. King, *A study of possible causes of artifactual decreases in the left ventricular apex with SPECT cardiac perfusion imaging*, Nuclear Science, IEEE Transactions on, vol. 46, no. 4, pp. 1016-1023.
13. Fitzgerald and P. G. Danias, *Effect of motion on cardiac SPECT imaging: recognition and motion correction,* J.Nucl.Cardiol, vol. 8, no. 6, pp. 701-706.
14. K. Murase, S. Tanada, T. Inoue, Y. Sugawara, and K. Hamamoto, *Effect of misalignment between transmission and emission scans on SPECT images*, JNM.,21,152-156.

15. P.P. Bruyant, M. A. Gennert, G. C. Speckert, R. D. Beach, G. Boening, J. Morgenstern, N. Arora, S. Nadella, M. A. King. *New design for a visual tracking system to detect patient motion in SPECT*. SNM Meeting 2004-Philadelphia. Poster #1338.
16. P.P. Bruyant, M.A. Gennert, G.C. Speckert, R.D. Beach, J.D. Morgenstern, N. Kumar, S. Nadella and M.A. King. *A robust visual tracking system for motion detection in SPECT: Hardware solutions*. To be published *Proc. Nucl. Sci. Symposium* 2004 IEEE.
17. http://radiology.unm.edu/Residency/RadiologyRotations/ Department of Radiology, Health Sciences Center, University of New Mexico, Accessed August 08, 2005.
18. M.A. Gennert, P.P. Bruyant, M.V. Narayanan and M.A. King. *Calibrating optical images and gammacameras images for motion detection.* J Nucl Med 2002;43: 222P.
19. *Introductory Techniques for 3D Computer Vision,* E. Trucco and A. Verri, Prentice-Hall, Inc. 1998, ISBN 0-13-261108-2 , Pg 123-136.
20. P.P. Bruyant, M.A. Gennert, S. Nadella, M.A. King. *The Visual Tracking System (VTS) for patient motion detection in SPECT: Quality control of the stereo calibration.* Submitted, MIC meeting 2005, Puerto Rico.
21. http://www.eclipse.org, Eclipse IDE Home Page, Accessed January 03, 2005.
22. http://www.intel.com/technology/computing/opencv/overview.htm, Homepage of Intel OpenCV Library, Accessed August 12, 2005.
23. G. Bradski, A. Kaehler, V. Pisarevsky, *Learning-Based Computer Vision with Intel's Open Source Computer Vision Library*, Intel Technology Journal, Vol. 9, ISSN 1535, May 2005, Pg 119-130.

24. http://ai.stanford.edu/~dstavens/cs223b/stavens_opencv_optical_flow.pdf, Computing Optical Flow using OpenCV, David Stavens, Accessed December 15, 2004.
25. http://sourceforge.net/projects/opencvlibrary/, OpenCV Download Page, Accessed December 14, 2004.
26. Barnard, Stephen and Martin Fischler. *Computational Stereo*. Computing Surveys. Vol. 14. No. 4 : 553-572 (1982)
27. E.Vincent, R.Laganirere, *Matching feature Points in Stereo Pairs; A comparative study of some matching strategies*, MG &V, vol. 10, pp.237-259, 2001.
28. http://www.ien.it/is/rec3d/structure.html, Trinocular stereo, Accessed April 15, 2005.
29. A. Shashua and M. Werman. *On the trilinear tensor of three perspective views and its underlying geometry*. International Conference on Computer Vision, 1995, 920-925.
30. R. Hartley. *Lines and points in three views and trifocal tensor. IJCV*, 22:125-140, 1997.
31. Étienne Vincent and Robert Laganiere, *Matching feature points for Telerobotics*, in Proc. 1st International Workshop on Haptic Audio Video Environments and their Applications, pp. 13-18, Ottawa, Canada.2002.
32. Nobiki, A. Naruse,H.Yabuta,T.Tateda, *Accurate Multi-Viewpoint stereo measurement using nonlinear optimization method,* Proceedings of the 1993 IEEE/RSJ International Conference on Intelligent robots and Systems, pp 1851 - 1856, July 1993.
33. http://www.ece.osu.edu/~chiench/VSC++_setup_for_cv.htm, Setting up OpenCV in Visual Studio, Accessed December 17, 2004.

34. http://www.cs.bham.ac.uk/resources/courses/robotics/doc/opencvdocs/, Function Syntax Reference, OpenCV, Accessed August 17, 2005.

35. wwwnavab.in.tum.de/twiki/ pub/Chair/TeachingWs04CV/svd.pdf, Accessed August 25, 2005

36. http://en.wikipedia.org/wiki/Eigenvalue, Eigenvalue Definition, Accessed August 25, 2005

37. online.redwoods.cc.ca.us/instruct/ darnold/laproj/Fall98/JodLynn/report2.pdf, SVD Derivation, Accessed August 25, 2005

38. http://web.mit.edu/be.400/www/SVD/Singular_Value_Decomposition.htm, Example on SVD, Accessed August 25, 2005

39. http://www1.cs.columbia.edu/~jebara/htmlpapers/SFM/node8.html, Stereo Computation Model, Accessed August 26, 2005.