IMPLEMENTATION OF AN INTERACTIVE VOLUME VISUALIZATION SYSTEM USING COMMODITY HARDWARE

By

JASON GRAHAM

Bachelor of Science in Electrical Engineering

Oklahoma State University

Stillwater, OK

2005

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
July, 2009

IMPLEMENTATION OF AN INTERACTIVE VOLUME VISUALIZATION SYSTEM USING COMMODITY HARDWARE

Thesis Approved:

Dr. Guoliang Fan

Thesis Adviser

Dr. Chris Hutchens

Dr. Martin Hagan

Dr. A. Gordon Emslie

Dean of the Graduate College

ACKNOWLEDGMENTS

I would like to thank my committee members for providing me this opportunity. I especially thank Dr. Fan for always being invested in his students' learning and having contagious enthusiasm. Thank you to Dr. Hutchens for getting me started in research. I would like to thank all the members of VCIPL who are all great people that are fun to be around, even when we're all working. Finally thank you to my parents who are always there for me, I love you both.

TABLE OF CONTENTS

Chapter Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

CHAPTER 1

INTRODUCTION

As the tools we use to approach computer vision problems become more sophisticated, implementing complex CV systems on commodity hardware becomes more feasible. Computers are faster, graphics processing is more powerful and versatile and other hardware components are cheaper and more readily available. Recently, portable handheld projectors became available to consumer markets. These miniature projectors use LED lamps and operate at lower resolution compared to larger (but still portable) projectors. Several companies are already attempting to adapt this technology for mobile devices such as smartphones in the future. This would complement the already ubiquitous camera, which has found its way into nearly every cell phone produced. These devices have modest specifications by the standards of other consumer cameras and projectors, as does commodity hardware. A piece of commodity hardware is readily available, but may not be ideal for a given situation, requiring special consideration to compensate.

Some of the most demanding CV tasks require the real-time processing of image data to produce interactive output. In this document, a volume visualization system and its implementation are described. It allows users to interactively view cross sections of a 3D dataset in real-time using a simple handheld display surface and a camera projector system. As a user moves the projection surface, the cross section image is augmented on the surface to reflect its location in 3D. The user experiences a volume dataset that exists in a specific region in space and by moving the display surface within that region a corresponding cross section of the set can be obtained.

This process requires a number of computations on image data for every frame displayed on the surface. If these computations take too long to complete, the user will lose the sensation of interactively navigating a dataset.

Forming an implementation of such a system on commodity hardware—specifically a DV camera, portable presentation projector and a laptop—presents a challenge. Real-time constraints must be met with the restricted CPU and graphics rendering power of a laptop, and using a DV camera requires compensating for the delay inherent in polling images. Even with such limitations a real-time vision system is still possible with the right algorithm enhancements.

1.1 Related Work

Producing the proposed volume visualization system requires a number of key components. Central to the system is the work Visual Odometry [3], which describes an approach to localization and mapping for a mobile robot. A stereo camera obtains data from the scene as the robot moves around and features are extracted and matched between images at video rates. Feature tracks are built across time by matches that overlap sequences of frames. Thus if a match spans the first two frames and the feature in the second frame also matches to a feature in the third frame, a track of length three is formed. Tracked features are used to reconstruct the motion of the robot by triangulation and camera pose estimation. Preemptive RANSAC is used to estimate the pose of the camera rig from frame to frame. This overall framework of feature matching followed by preemptive RANSAC for hypothesis refinement is adapted for the visualization system.

Preemptive RANSAC [4] is a hypothesis verification framework that limits the number of computations used for estimation by eliminating (preempting) poorly performing hypotheses. RANSAC is short for “Random Sample Consensus”, a well established algorithm that separates inliers from outliers by optimizing a performance function on data. The method selects a random sample of the available data and declares it as a hypothesis or representative set. Other data members are then tested against the set to generate a cumulative error score. A number of

hypotheses are tested and the best hypothesis is selected as the one with the lowest error. Preemptive RANSAC adds a layer of constraints and particular execution method to the generic algorithm to ensure the hypothesis selection occurs in a predictable amount of time. In a real-time system this is invaluable as the hypothesis verification step is only one of several that must be executed on every frame. The number of hypotheses is fixed and an iterative process begins. A single observation from the dataset is used to score all the generated hypotheses. It is assumed that the highest error hypotheses contain outliers and a specific number are rejected. A new observation is selected and the process repeats until a single hypothesis remains.

The volume visualization system implemented in this work was first developed as part of an augmented reality system called the Universal Media Book [1]. The UMB's main mode is to act as a book displaying static images or video. Each time a page is turned the book would update the displayed data to reflect the next page. The user could browse through the virtual book much like a normal book. Within this book framework a page might contain a special element that indicates a volume dataset. When the user encounters such a page he can remove the page from the book and use it as the display surface to navigate the volume dataset. The book mode and volume display modes both function on the same procedure of geometric estimation frame by frame in real-time, but have different constraints. In book mode the page geometry is clamped at the spine and the outer edge of the page moves along a semicircle. In volume visualization mode the data is clamped to a region of space and a slice is determined by the intersection of the display surface with that region of space.

1.2 System Overview

This system's function is to augment an unmarked surface with data determined by the location of the surface in 3D space. A projector and camera are used to project data onto the surface which is seen by the camera and used to estimate the location of the surface in the projection space. The first step to describing the system is to define the interface model. Volume data is modeled as a region of 3D space. During calibration the coordinate system for 3D space is defined with a particular origin and orientation. From the origin the volume extends in the positive x and y directions and the negative z direction. The user can see cross sections of this volume data by moving a display surface into the corresponding region of 3D space. As the user moves the display surface through the defined region it is updated with new images determined by the location of the surface in the volume. This gives a user visual feedback of the location of the display within the data, allowing them to adjust the surface to find interesting points in the volume.

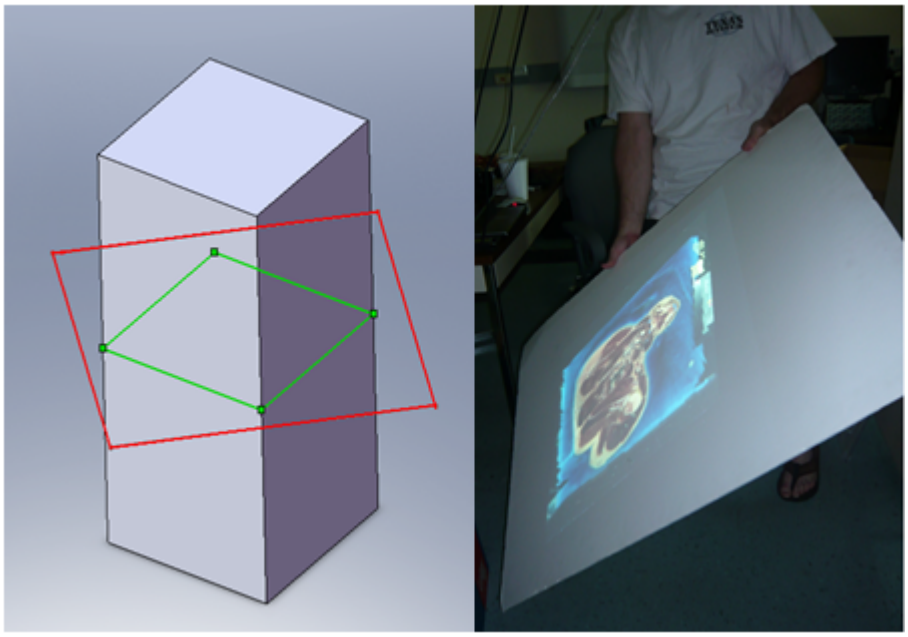

Figure 1: Left: Red rectangle is the handheld display surface, gray volume is the dataset of interest and green rectangle is the cross-section image that corresponds to the intersection of the dataset and display surface. Right: A user interacts with the Visible Human Database

1.2.1 Calibration

As the system requires knowledge of real 3D points, the projector/camera pair must be calibrated to a common 3D reference. This is the origin and orientation mentioned earlier. Calibration is the process of finding a set of parameters that allow an accurate mapping of points from the 3D universe to locations in the device's image frame. Parameters are divided into two groups: intrinsic parameters that describe physical device properties and extrinsic parameters that describe the location and orientation of the device relative to the defined 3D coordinate system. There are a few possible techniques available to calibrate cameras; all depend on the use of 2D to 3D correspondences. In 2000 Zhang [9] proposed a method designed to minimize the amount of user input required. His method assumes a 3D reference frame defined by the pattern in view of the camera. This pattern has evenly spaced fiducials that are easily detected so they can be paired with given 3D points. The user moves either the camera or the pattern and acquires several views. All of the correspondences are then used to calculate a closed form solution for the intrinsics and extrinsics. This solution is then refined through maximum likelihood estimation to generate a set of intrinsics for the device and extrinsics for each view obtained.

One pass of the geometric calibration algorithm calibrates the camera, but another pass is required to calibrate the projector. The key difference between the camera and projector is that a camera receives light as a sensor along light vectors from objects in the scene where a projector generates light and imposes it on the scene along its image light vectors. To use Zhang's calibration method on the projector a set of 2D to 3D correspondences must be made. While the camera could be directly calibrated with its own image data a projector can only impose data on the scene. If a pattern is projected on a known plane and the camera is calibrated and views this plane, the scene is sufficiently constrained to determine the 3D locations of the projected 2D data through the camera calibration. With the correspondences found, Zhang's algorithm returns the intrinsics and extrinsics for

the projector as well. With both the camera and projector calibrated to the same 3D frame, it is possible to map a point common to the camera and projector images to a 3D location.

Geometric estimates of the location of the display surface are formed by comparing a camera image to the estimate of the camera image. No static markers are used on the display surface to assist in determining its 3D position; it is determined entirely from correspondences found in the cross-section data between the camera and projector frames. With accurate correspondences a solution set of 3D points is found that defines the location and orientation of the display surface within the volume display area. A new cross section of data is determined from this information and used to augment the display surface. By repeating this process of geometric estimation and reapplying a new image as fast as possible, real time navigation of the dataset is realized.

A photometric calibration step is necessary to ensure accurate correspondences can be generated in the online algorithm. As matching occurs between the camera and projector frames, it is necessary to correct for the color space change that occurs between sending an image to the projector and the resulting image in the camera. In the main algorithm this translates to correcting the grayscale image of the projector data to make it appear more like the grayscale of the camera image. The mapping is thus from a color image from the framebuffer to a grayscale image. A look-up table is generated that describes this mapping by collecting camera pixel intensity values that correspond to projected colors. A color pixel from the projector framebuffer appears as the sum of the intensities for each color value as determined by the LUT.

To find accurate correspondences a tracking method used on a stereo camera pair [3] is adapted to function using the image from the camera and the image sent to the projector. Features are extracted from the two images and matched. The set of matched features is refined to a solution set that determines the newest estimate for the geometry of the planar surface. Before feature extraction, the estimate of the

camera image must be generated. The camera views the projected data, so the geometry used to generate the projected data is used to generate the expected camera image.

1.2.2 Feature Extraction and Matching

Features are extracted from the camera image and the warped projector framebuffer image using the Harris detector [5]. This detector creates a response image where high values correspond to sharp corners. A window around a given image patch is shifted in all directions. If the intensity of the window changes dramatically when moved in any direction, the result is a high response. If the window were to be shifted and not change intensity much, then the current image region might be low variance or a simple edge. Because the output of the detector is the corner response for every pixel in the image a refinement step is necessary to identify feature locations. A technique known as non-maximal suppression is applied to the response image. As a window is passed over the image, the central pixel is deemed to be a feature location if it has the highest intensity of all pixels within the window. This produces a predictable density of features determined by the size of the NMS window and the complexity of the image.

With feature locations identified for the camera image and the expected camera image, the next step is to identify features that exist in both images. A feature is determined to exist at a particular location identified by NMS, but that location by itself is insufficient to provide accurate feature matches. Accurate matches should have features spatially close to one another, so matches are only considered if they fall below a distance threshold. To enable matching a feature is described by an image patch surrounding the location found by NMS. If features are within the distance threshold they are compared by calculating the normalized correlation score between the images patches. This process is optimized to reduce the amount of per-match computation required. If the NC score is above a minimum threshold then it is possible a match is found. To verify a match a mutual consistency check is

made: if the features both select each other as the highest available NC score in their respective neighborhoods, then it is a match.

1.2.3 Geometry Estimation

The set of match points contains inlier and outlier matches. Inlier matches can be triangulated accurately to locate a 3D point on the display surface. Outlier matches will produce triangulation results inconsistent with the orientation of the board. To determine which matches qualify as the most representative inliers a preemptive RANSAC [4] scheme refines the results to a set of four matches that likely reflect accurate 3D points on the display surface. Preemptive RANSAC is optimized for situations where producing a result in a limited amount of time is more important than an optimal result. A set of hypotheses are formed from random sets of four match points. Homographies are computed to describe the mapping from the features in the camera frame to the expected camera frame and back for each hypothesis. To determine if the hypothesis contains inlier matches it is tested against other matches. The homographies allow scoring by computing the reprojection error between the location of a point in a frame and the expected location. The expected location is computed by projecting a feature's match back into its own frame. Each time error is computed the same match is used for all the generated hypotheses and the hypotheses with the highest error are preempted. Preempted hypotheses are removed from consideration. After a certain number of iterations only one hypothesis will remain which has survived each round of preemption, ending with the lowest score. This winning hypothesis is the final output of the preemptive RANSAC scheme for hypothesis creation and selection.

The winning hypothesis contains four match points which must be triangulated to find their corresponding 3D location. The calibration information for both devices is composited together with each triangulated point into a 4x4 matrix that undergoes singular value decomposition. The result is a 3D point lying on the display surface. With four of these 3D points a normal and centroid for a best fit plane are similarly found through SVD. This plane represents the latest geometric

estimate of the location of the display surface. With this data a new cross section is projected on the surface to start the process again.

In this implementation a DV camera is used that has an internal image buffer. Issuing a command to acquire an image from the camera shifts these buffers and returns the image stored in the last buffer. This presents a problem in implementing the algorithm such that each pass generates a new geometry which is used to compute a new image that is immediately displayed. Instead a certain number of estimates must be stored so that each time a camera image is acquired it can be matched to the appropriate estimation of its appearance according to the older geometric data. A simple geometry buffer is implemented to solve this issue. Each new estimate is immediately displayed on the board, but the geometric information must be stored so that when the camera finally returns an image with that geometric representation the warped projector image will contain the right data at the right orientation to allow matching and a new geometric estimate.

1.3 Contributions

Using publicly available libraries and readily available hardware a real-time volume visualization system is implemented. The visualization system is computationally demanding and runs in a continuous feedback loop where camera images must be compared to constructions of the camera images to generate a new estimate. Requiring the camera to image the scene as part of an algorithm that completes execution several times a second places an important restriction on the camera. It must be able to image the scene very quickly and return the image to the algorithm very quickly. For machine vision cameras this is no problem as they can perform an operation like this in milliseconds. A digital video (DV) camera, however, was not designed for this task but was rather designed to capture and store video sequences. To overcome this problem a geometry buffer is implemented, allowing the estimates to queue and line up with the camera images.

A camera capture library designed to allow frame capture from commodity cameras provides frame acquisition through a C++ function. When using the frame grabber function to get a specific image of a changing scene it becomes obvious that the images yielded by the camera are old compared to the images being placed in front of the camera. This delay is assumed to exist because an internal frame buffer stores three frames and shifts images along this buffer with the oldest image being sent to the computer. This assumption implies that a complementary buffer designed to delay estimation to match the camera delay could solve the problem. To compensate for the camera being late on providing the appropriate image, geometric estimates are placed into a buffer before they are used and the expected camera image is built from the stored estimates. As a result this real time volume visualization system can be used with a cheap and readily available DV camera.

The previously implemented system [1] uses such a machine vision camera. It was thus possible to set the system up as a loop where each iteration solved the current display geometry and updated the projected data. In my system this is not possible as the output camera image lags behind the scene it is imaging. The geometry buffer solves this problem, but complicates the algorithm at the stage of producing the expected camera image. The UMB assumes that the expected camera image can be created from the current projector image by warping it to the camera frame. Since the current projected image is several frames ahead of the camera's image in my system, the data contained in each image can be radically different. Thus the expected camera image must be recreated from the stored geometric data each frame. This requires an extra call to the OpenGL display function without displaying the data to the projector. Even with this extra step the system is able to execute its necessary functions in real-time.

Additionally, the photometric calibration method used in the Universal Media Book to map projected color to camera intensity performed poorly due to the camera's tendency to switch sensitivity modes as the image changed in overall intensity. A calibration pattern is devised to maintain an overall neutral intensity

and provide sharp edges, preventing the camera from changing the white balance or auto-focus. A simple method for calibrating the camera and projector using OpenCV calibration functions for both is also described.

CHAPTER 2

SYSTEM IMPLEMENTATION

Several software libraries are used to provide important basic functionality such as hardware interfacing and to provide implementations of useful image processing and computer vision tasks. Access to the graphics hardware is provided through OpenGL with windowing in GLUT, which also dictates the program's overall execution procedure. GLUT uses a system of callback functions to control the windowed graphics system. Callback functions are triggered by program events such as keyboard strokes or if a flag is set to update the display. Multiple events can be queued for execution such as a keyboard, redisplay and timed callback event. If no callbacks are waiting in the queue, then the default event is to trigger the idle callback repeatedly until a new event occurs. GLUT provides much of the interface used to initiate an OpenGL graphics environment. OpenGL contains definitions allowing 3D texture data to be stored on and used directly by the graphics card to texture a 2D scene.

The DV camera provides images over the Firewire bus at up to 15 frames per second. Accessing these images requires the ability to interpret the image data coming from the firewire bus. An independently developed camera capture library called VideoInput provides access to the camera as an image acquisition device. It is important to note that the DV camera used is not compatible with any acquisition standards where more advanced camera control libraries are available. While this library enables us to ask for an image from the camera, the camera does not necessarily deliver an up to date image. This is the main reason the DV camera is difficult to use in a real time setting.

Table 1: System platform specification.

Camera 	Canon Elura 40MC Framerate: 15 fps Resolution: 1024 x 768 Interface: FireWire
Projector 	Toshiba TLP X100 Framerate: 60 fps Resolution: 1024 x 768 Interface: VGA
Computer 	Toshiba Tecra M7 CPU: Intel Core Duo, 2.0 GHz Main Memory: 1 GB
Graphics 	NVidia Quadro NVS 110M Memory: 128 MB
Software Libraries 	OpenCV, OpenGL, GLUT, VideoInput

The OpenCV libraries provide a number of functions for common computer vision tasks. Portions of geometric calibration are provided by OpenCV as well as data structures used throughout the implementation to organize information. As well as controlling the overall program execution, OpenGL provides access to texturing functions and the ability to produce a virtual 3D scene and rasterize it into an image. GLUT provides all windowing related functionality and VideoInput provides a frame grabbing function to access the camera's images.

This system is split into two main programs. The calibration program performs all the offline calibration steps and the volume display program contains the online algorithm for generating the display using the procam. The steps taken in the calibration program need to only be performed once or very occasionally, such as if the system were moved. The volume display program is designed to start displaying immediately, and can be used to navigate the dataset after a few seconds of camera setup. Upon starting, the volume display program loads all the stored calibration information. If the user specifies that extrinsics calculations are needed then some brief chessboard interaction is required before volume display commences. Once the display begins to update a few seconds after the first volume slice is shown, the user can move the display surface through the volume.

The volume visualization program operates in the framework of OpenGL. This means that all execution takes place in callback functions. An idle function is called when no other callbacks are waiting. The display callback is used to change the displayed scene and the keyboard callback is called when the keyboard is pressed. The idle function controls the overall program loop. As the system diagram shows, the volume visualization program is a loop with a geometry buffer. Starting from the geometry buffer a run through the program can be traced. The geometry buffer data is used to create an estimate of what the camera image should be. This is the same as the projector image warped to the camera frame. It is compared to the image acquired by the camera to find matches. These matches are refined to a set that are likely to produce a good estimate. They are triangulated and used to form an

estimate of the surface geometry. The new geometry is added to the geometry buffer and the process is started again.

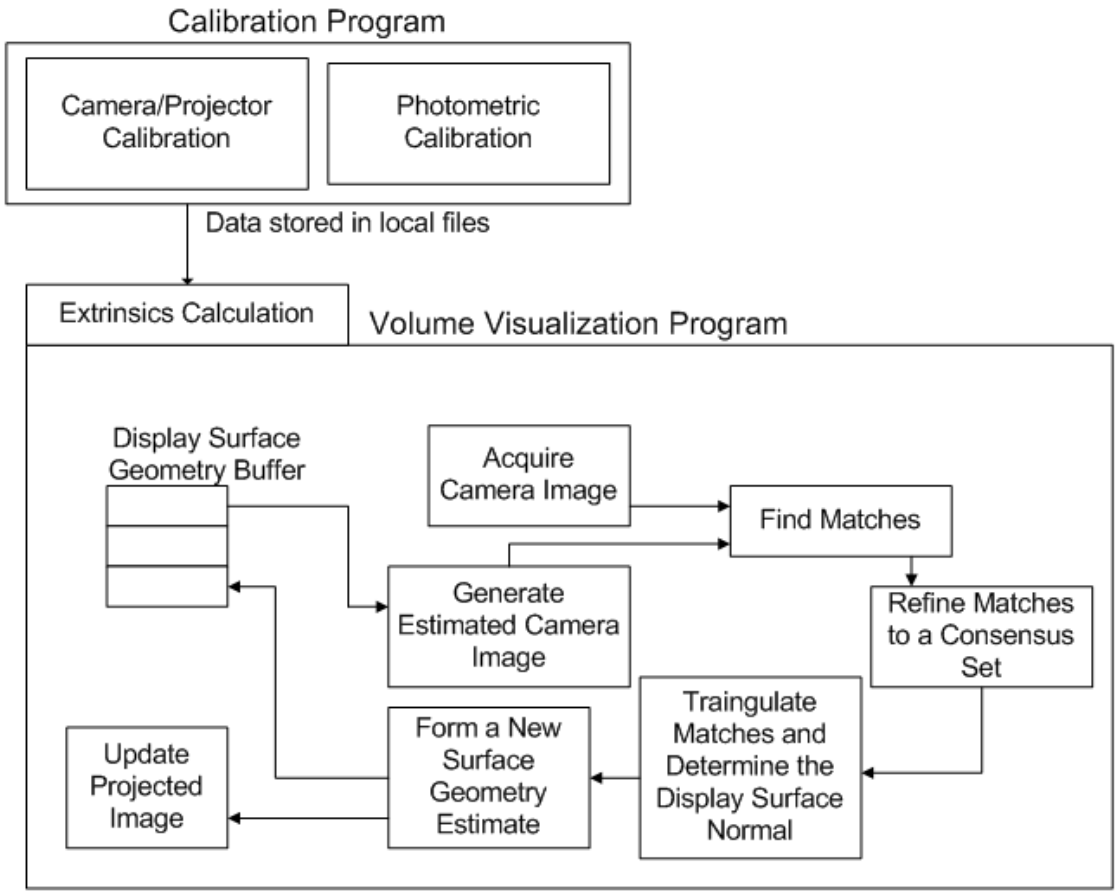

Figure 2: System flow diagram

2.1 Calibration

The calibration program offers the option to calibrate a camera or projector or both. The resulting information is stored in a file to be read by the volume display program. These files can also be loaded by the calibration program to verify previous calibrations. Photometric calibration can be performed at this point as well, storing the look-up table to be read later by the volume display program.

Camera calibration is a well studied problem in computer vision. The result of calibration is a mapping from 3D points to their locations in the 2D camera image. In this system a camera is calibrated using Zhang's [9] method. Zhang wanted to describe a method that was simple to perform and highly robust, designed to allow anyone to calibrate a camera without special knowledge. Calibrating the camera requires one extra tool: a calibration chessboard. This board must be a rigid plane with a printed or drawn chessboard. It is easy to accurately identify the corners of the chessboard in the camera image, so it is used as a well defined reference plane in 3D space. The key to a successful calibration is finding good correspondences between 3D points and where they show in the 2D camera frame. This is accomplished by assigning 3D values to the corners of the calibration pattern and associating them with detected corners from the camera image.

2.1.1 Camera Calibration

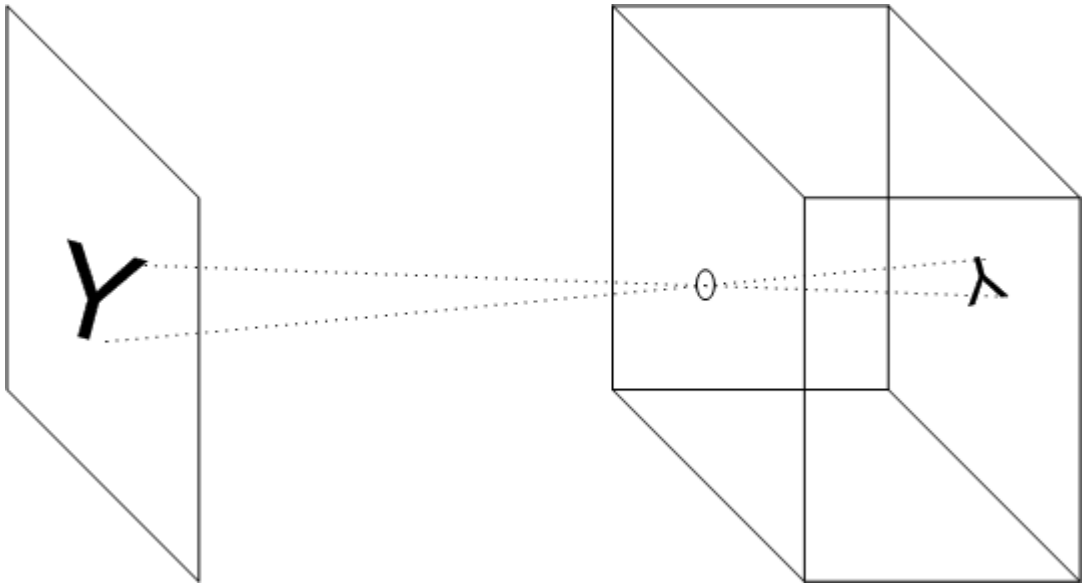

Figure 3: Pinhole camera model. An image is linearly projected onto the imaging surface inside the camera.

A pinhole camera model assumes that a camera is much like a tiny aperture where rays project an inverted image of the scene facing the pinhole onto the inside wall of the camera. This is a linear map from 3D coordinates to a 2D image location

on the back of the camera. Here $\bar{x}$ is an image coordinate (2D) and $\bar{X}$ is a scene coordinate (3D):

$$\bar{x} = P\bar{X}. \qquad (1)$$

The matrix *P* is defined by the intrinsic and extrinsic parameters of the camera:

$$P = K[R|t]. \quad (2)$$

The matrix *K* is composed of the intrinsic parameters:

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \quad (3)$$

Where f_x and f_y are the focal lengths in each dimension and (c_x, c_y) is the camera center. The matrix *R* is a rotation matrix and *t* is a translation vector.

Intrinsics are physical definition of the camera's internal geometry and extrinsics define the location of the camera in the 3D coordinate system. The intrinsic parameters are represented by the matrix *K,* and the extrinsics by $[R|t]$. By manipulating the equation for the camera matrix *P,* it is possible to arrive at a closed form solution for the intrinsics and extrinsics if enough correspondences are provided to properly constrain the parameters.

This solution is treated as an initial guess for a maximum likelihood approach to estimating the camera parameters. With the initial guess it is possible to project 3D points into the 2D image frame. An error function is defined to measure the difference between the projected 3D points and the locations of the corners in the image. The initial guess doesn't contain provisions for lens distortion, so at this step distortion is assumed and initially guessed to be zero. By minimizing the error function through the Levenburg-Marquardt algorithm, a more accurate solution is found.

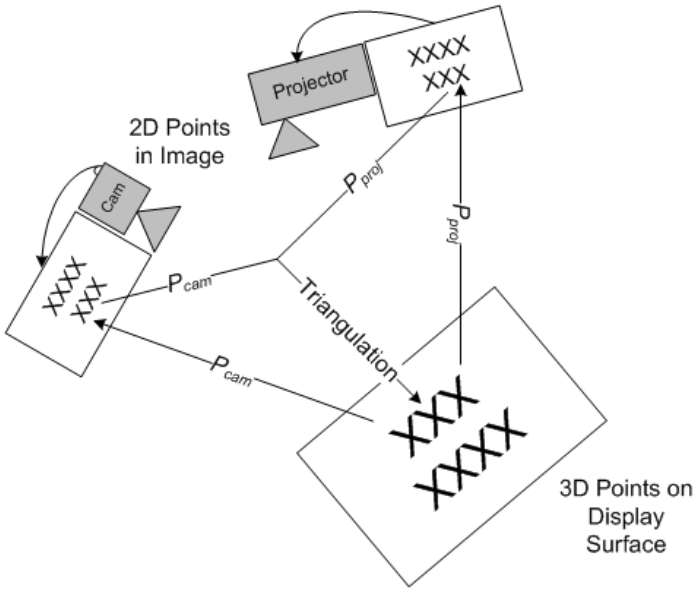

Figure 4: Calibration projection matrices provide a one-way mapping from 3D to 2D, but both projection matrices are required to go from 2D to 3D via triangulation.

To begin calibration, the user places the pattern in view of the camera. An image is acquired and processed to find the pattern in the image. OpenCV provides an implementation that can find a chessboard in an image based on the number of rows and columns on the board. To avoid rotational ambiguity one dimension should have an odd number of elements. Each found corner corresponds to a 3D coordinate defined by the dimensions of the calibration pattern. With the 19 by 14 board as an example the upper left corner corresponds to $X = [0, 0, 0]$ while the lower right corners is $X = [18, 13, 0]$**.** A number of views of the pattern are taken by moving it to different locations in the camera view and different depths relative to the camera. At minimum two views can provide calibration results, but more views are preferable. A large number of views helps to ensure that the parameters of the camera are well constrained and lens distortion is modeled well. At this point the user has generated a set of 2D points across several views of the calibration pattern which are assumed to always correspond to 3D values lying on the $z = 0$ plane. With the correspondences stored, the OpenCV implementation of Zhang's algorithm is used to find the intrinsics and extrinsics, including distortion parameters.

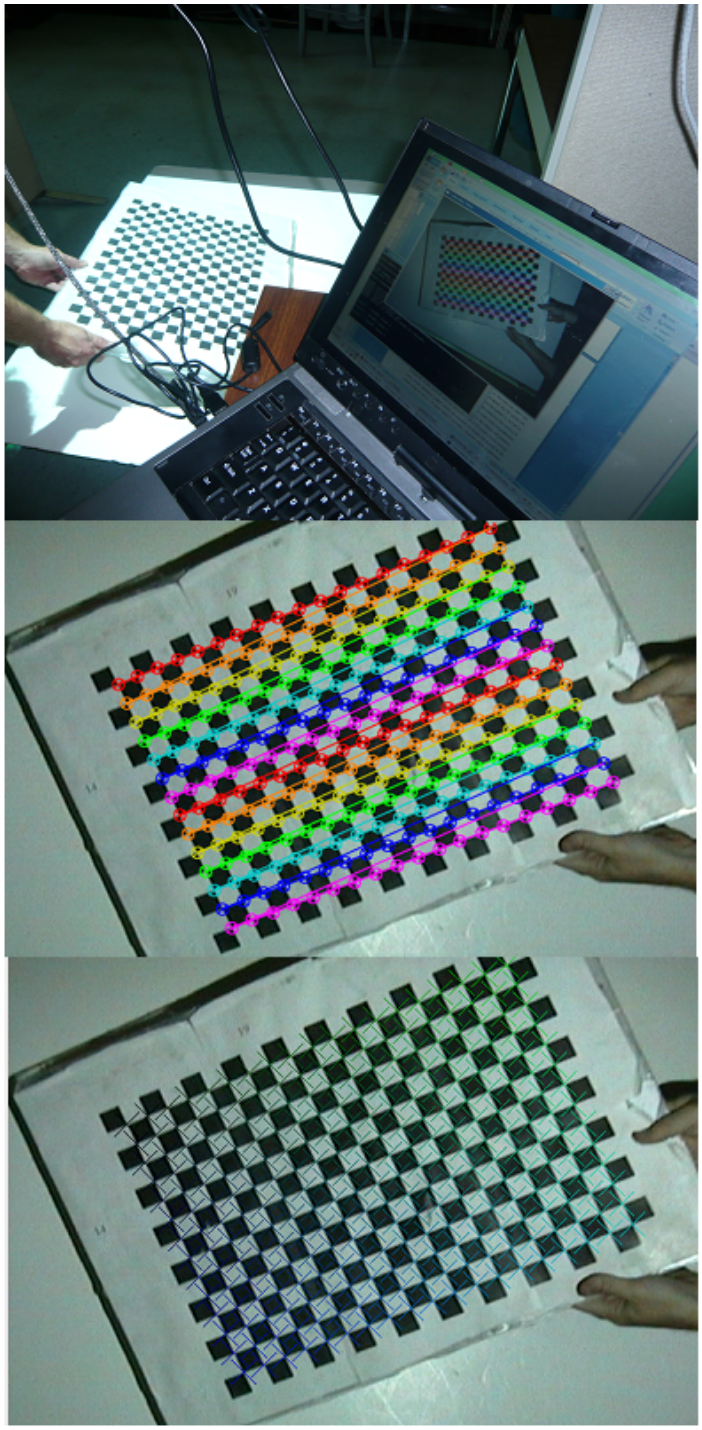

Figure 5: Top: Camera undergoing calibration with a physical calibration pattern. Middle: Physical pattern with found corners. Bottom: Reconstructed corners using 3D points and calibration information.

Each time an image is acquired and the pattern is found within it, the image of the found corners is shown to the user calibrating the system. This ensures that each calibration image is unique and accurately finds the location of the chessboard in the image. If contrast in the image is low then the results can be bad enough to reject a camera image. When using cameras that automatically adjust white balance and focus, this verification step filters out the anomalies that would reduce the accuracy of the result. When pattern corner acquisition is complete, a camera image is augmented with all of the found corners from the calibration process. This is useful for determining if the chessboard was captured in a wide variety of positions or if they are relatively homogenous. If this point cloud looks good enough then the results of the calibration are showed one at a time. Each camera frame used to capture the chessboard is augmented with the reprojection of the chessboard corner locations. The image shows an x on or near the clearly visible chessboard corner, allowing the user to visibly assess the amount of error present in various locations for this calibration.

In summary, the user starts the calibration program and places the physical pattern in view of the camera. After deciding the number of calibration images the program begins to find chessboard patterns from the camera image interactively. The image of the found corners is shown and the user presses a key to accept or reject after repositioning the pattern. This is repeated until the decided number is acquired. An image of the cloud of points found is shown and accepted or rejected (if rejected the user must start over). Then each calibration image is shown in sequence with the reprojected 3D points to allow a last visual check of the calibration accuracy before it is used in the main program.

2.1.2 Projector Calibration

The projector has a more complex calibration procedure. Assuming the same pinhole camera model, a projector could be seen as the opposite concept: rays leave the box through the hole and impose a color on the scene. Using this model it makes sense to pursue the same calibration technique as is used on a camera. The problem is acquiring the correspondence points.

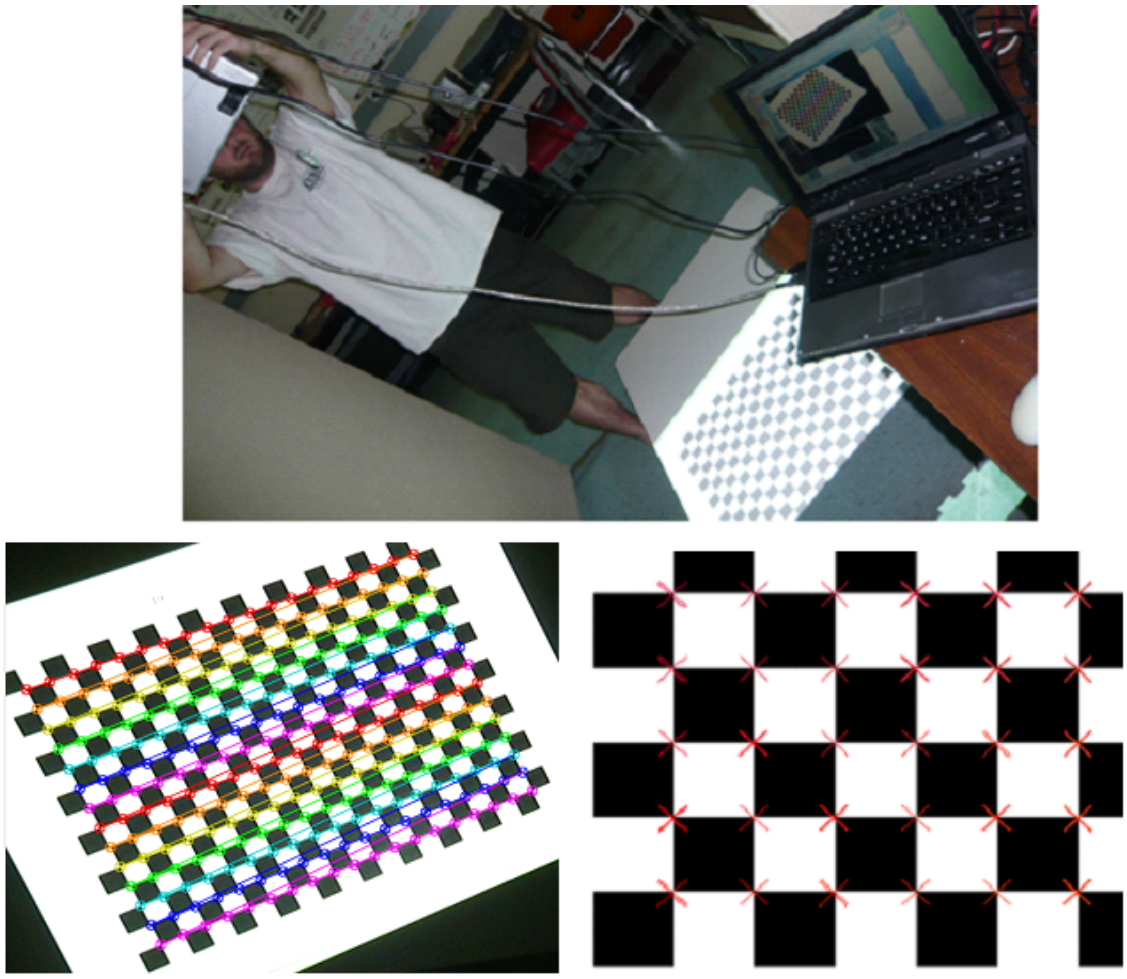

Figure 6: Top: The projector is held in hand and pointed at the z=0 plane to project a pattern for calibration. Left: Resulting calibration image with found corners. Right: Reconstructed projector frame points using 3D points and calibration information.

The camera forms an integral part of the solution as it is the most readily available way to measure the scene. Since the projector is designed to augment the scene, we would like to project the calibration pattern and find it somewhere in the scene. The 2D portion of the correspondences would be found from the image provided to the projector and the 3D points would have to be reconstructed from the view of the scene. The calibrated camera can map 3D points into its 2D image, but cannot do more than define a ray in 3D space. By constraining it to a known surface the ray can be reduced to a point: the 3D location of the pattern corner in space. The image from the camera will show the pattern projected onto a planar surface assumed to be the plan $z = 0$. This convenient assumption allows the 2D points found in the camera image to be converted to 3D points.

To transform the 2D camera locations into 3D coordinates residing on the $z = 0$ plane, the transformation matrix for the camera is manipulated. This matrix, P, defines the mapping from 3D points to 2D points and is a 3x4 matrix, and thus not invertible. The $z = 0$ assumption means that a column of the P matrix can be removed as it would have no effect on the calculation. Given

$$x_{cam} = \begin{bmatrix} x_c \\ y_c \\ 1 \end{bmatrix},$$

$$P_{cam} = \begin{bmatrix} \bar{p}_1 & \bar{p}_2 & \bar{p}_3 & \bar{p}_4 \end{bmatrix},$$

where $\bar{p}_n$ is a 3x1 column vector, and

$$X_{3D} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix},$$

then:

$$x_{cam} = P_{cam} X_{3D}. \qquad (4)$$

This is the standard projection equation for determining where in an image a 3D point will be given a completed camera calibration. In this situation the camera is calibrated already so the camera matrix is available. Assume that X_{3D} has a z coordinate of zero:

$$X_{3D} = \begin{bmatrix} x \\ y \\ 0 \\ 1 \end{bmatrix}.$$

Then

$$p_{cam} = [\overline{p_1} \quad \overline{p_2} \quad \overline{p_3} \quad \overline{p_4}] \begin{bmatrix} x \\ y \\ 0 \\ 1 \end{bmatrix}$$

implies

$$[\overline{p_1} \quad \overline{p_2} \quad \overline{p_4}] \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = P_{\Pi} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \qquad (5)$$

Where P_{Π} is now invertible. This implies that 3D points located at the plane $z = 0$ can be found based on their locations in the 2D camera image.

$$\begin{bmatrix} sx_{3D} \\ sy_{3D} \\ s \end{bmatrix} = P_{\Pi}^{-1} \begin{bmatrix} x_{cam} \\ y_{cam} \\ 1 \end{bmatrix} \qquad (6)$$

This presents a very simple way to calibrate the projector using the same tools as were used to calibrate the camera. By converting all the camera locations to 3D points on the $z = 0$ plane, the 2D-3D point correspondence set is complete by using the locations of the corners in the pattern image provided to the projector. As before, the OpenCV implementation of Zhang's algorithm provides intrinsics, extrinsics and distortion parameters from this correspondence set.

The user experience is similar to camera calibration, but requires repositioning of the projector rather than the physical calibration pattern. Also the user must orient the projector so that the projected pattern falls on the display surface in view of the camera. The camera to scene relationship must not change as it would invalidate the 3D points found and provide an invalid result. The user chooses a number of views to acquire and the program begins to search for the pattern image in the camera view. As with the camera, the projector calibration verifies that the found corners are accurate with the user by waiting for a key press. The user repositions the projector and acquires the next view. This is repeated until all views are acquired. As with the camera calibration, a point cloud shows how well projected patterns spanned the camera image and the locations of the reprojected camera points are shown for each frame. This verifies that the conversion from 2D camera points to 3D points works well in the camera frame. The projected pattern image is also augmented with the projections of 3D corners in the projector frame. These corners show how well the projector calibration works.

With projector and camera calibration complete, their intrinsic parameters and distortion coefficients are stored to be used as needed. These parameters do not need to be recalculated. Extrinsics may change over time as the camera and projector might move relative to each other. If this happens and extrinsics are not recalculated, then essential later steps will suffer from inaccurate results. Thus extrinsics are recalculated from time to time as part of the main algorithm.

2.1.3 Photometric Calibration

An important part of the main algorithm is matching features. The basis of this process is the comparison of grayscale image patches. When a pixel is projected onto a surface and then viewed by a camera, the color of the pixel in the camera image will not necessarily be the same as when it was projected. This implies a mapping from projected color intensity to camera image color intensity. As the images are converted to grayscale as part of the algorithm, the mapping is constructed from color directly to gray values. This photometric calibration step increases the likelihood of finding good matches.

Table 2: A sample of the Look-Up Table generated by photometric calibration. The values in the color columns are the perceived camera gray value for a pixel with given input intensity in that color.

Input Color Intensity	Red	Green	Blue
98	91.1514	135.402	176.858
99	92.6463	136.805	181.2
100	94.2715	137.6	182.896
101	94.3026	138.993	185.509

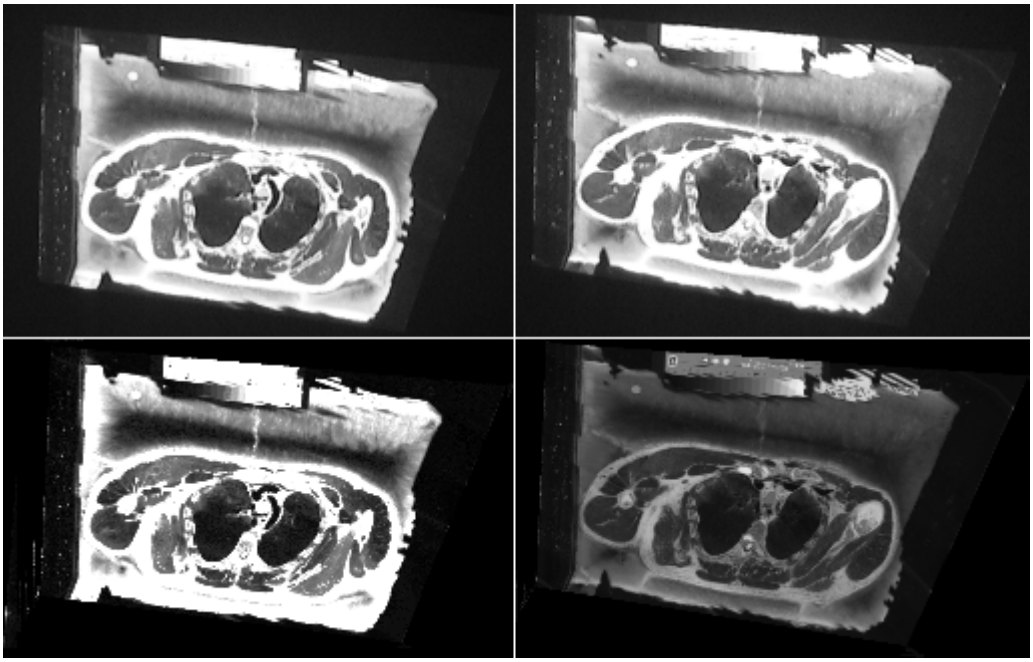

Figure 7: Photometric calibration result versus standard grayscale conversion. Top: camera images. Bottom left: photometric calibration result applied to an expected camera image. Bottom right: expected camera image converted to grayscale.

The mapping is highly non-linear, so a look-up table is used to store individual data points for each projected color and perceived camera intensity. The simplest method of filling this table would be to project each color intensity individually in the full frame of the projector, and then average the camera's perceived intensity in the projected region to store for that color value. The camera prevents this simple method from working effectively, however. The camera will attempt to focus, and if the scene is just a single color then the intensity will modulate with the focusing attempt. Also the camera will attempt to adjust its white balance as the scene intensity changes. This directly changes perceived intensity for a given color. To correct for these problems a pattern is introduced that is complex enough to maintain focus and keeps a constant intensity average to prevent the white balance from changing. In the pattern the bright regions lower in intensity as the dark regions increase in intensity. The outer border is a constant gray and keeps focus as the intensities in the two regions overlap. Calibration values are obtained by averaging the intensities in the color-varying parts of the pattern. Later, this data is

used to map a projected color image to the camera frame for comparison. Each pixel is computed from the values in the table according to this equation:

$$pixel = LUT_r(R) + LUT_g(G) + LUT_b(B) - 2 * AvgZero$$
$$AvgZero = [LUT_r(0) + LUT_g(0) + LUT_b(0)]/3 \quad . \qquad (7)$$

The pixel returned is the gray value as seen in the camera for a given projected color pixel with values [R, G, B].

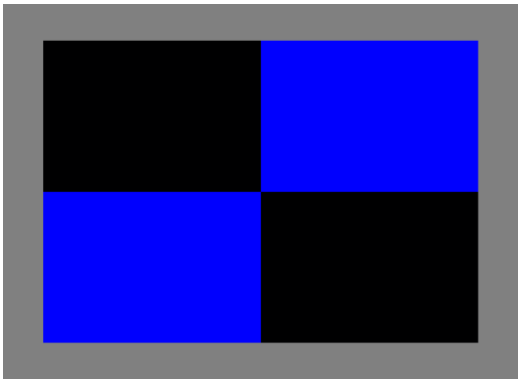

Figure 8: Photometric calibration pattern

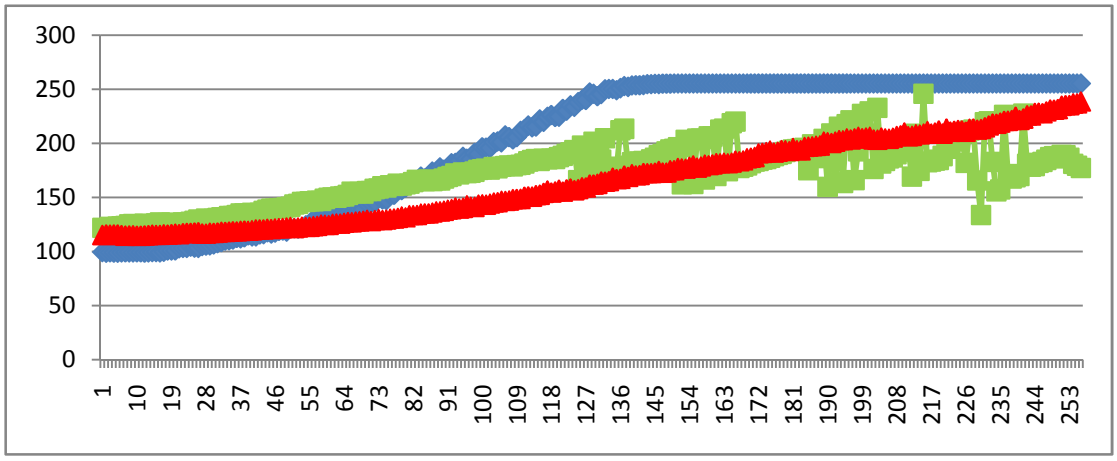

Figure 9: Photometric calibration using full frame color

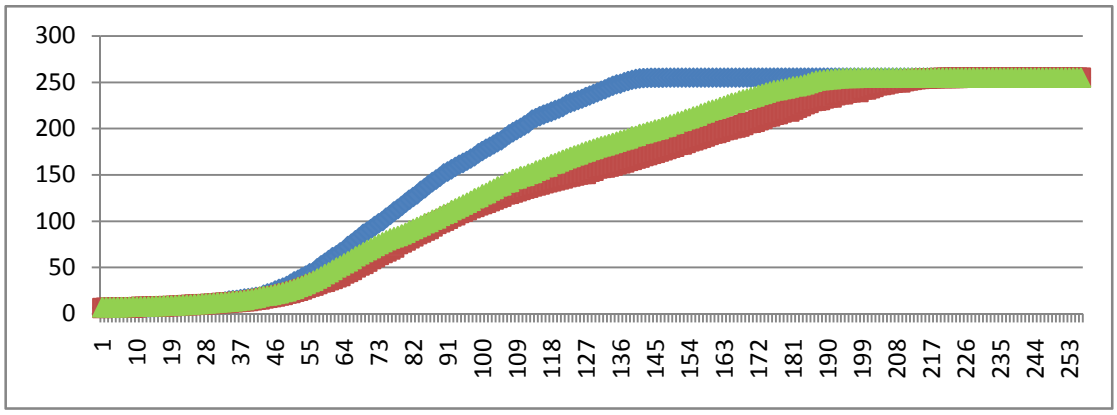

Figure 10: Photometric calibration using pattern

The volume visualization program begins by offering the option to calculate new extrinsics for the camera and projector. The projector and camera can move relative to each other slightly and destroy the operation of the system, so this step allows extrinsics recalculation as needed. Only one view of the calibration pattern for each device is needed, so it can be performed relatively quickly. These values are stored so that subsequent starts of the program can immediately display volume data.

2.2 Feature Matching

The feature matching stage takes calibration information, a camera image, and the oldest geometry from the geometry buffer to generate a set of matched points between the camera frame and the expected camera frame. This includes feature detection, localization, and matching steps. Before features can be detected, however, the constructed camera image must be created.

2.2.1 Expected Camera Image

Once a camera image is acquired it must be paired with an expected camera image to generate match points. At program start, the projector is set to display a number of frames where the planar geometry is assumed to be $z = 0$. This pulls the cross section from the dataset corresponding to that plane and displays it. After the initial frames are up, the algorithm begins forming estimates but not storing them.

This gives enough time for the camera focus and white balance to normalize. When the estimation process has produced a few valid estimates they begin to be stored in the geometry buffer. The expected camera image is constructed by projecting the geometry of interest into the camera frame. This defines the quadrilateral where the data is expected to be in the camera image. The image is rendered in the backbuffer and never swapped to the front buffer. This means that the graphics hardware is used to generate the expected camera image without displaying it. This step increases the amount of time per frame over the baseline implementation because the image is not already in the framebuffer and ready to be warped and used.

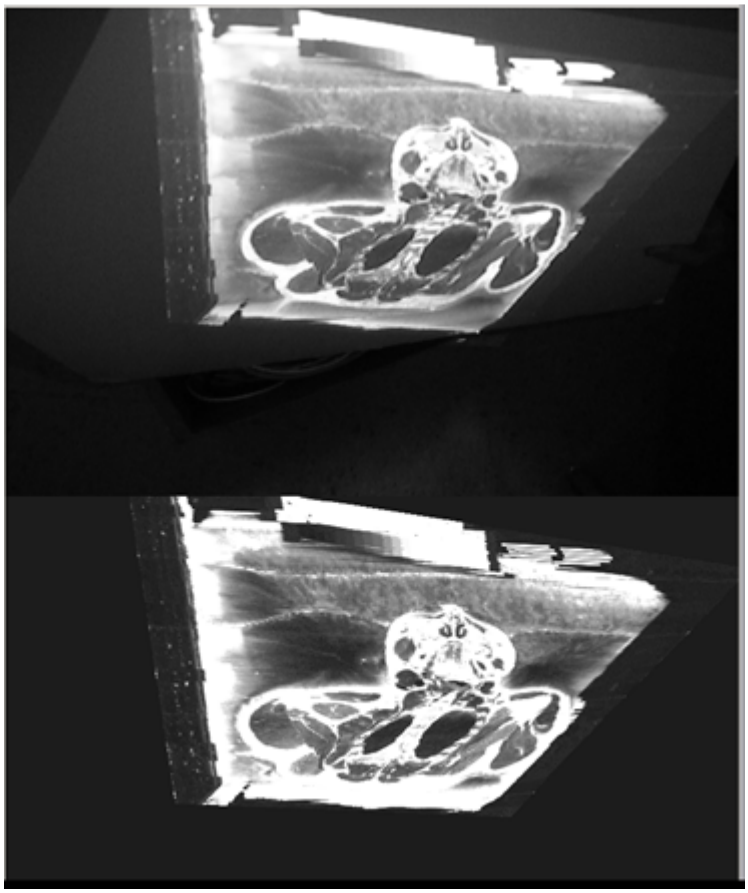

Figure 11: Top: Camera image in grayscale. Bottom: Expected camera image with color correction. By rendering the expected camera image in the backbuffer using a previously calculated geometry it matches nearly perfectly with the camera image.

The reconstructed camera image must be converted to a grayscale image through the data in the look-up table from the calibration phase. Each pixel in the grayscale image is calculated from the color of the original pixel and the values in the look-up table. The look-up table was filled with calibrated camera intensity values reflecting the average intensity seen by the camera for a given projected color and intensity. Equation 7 shows that the expected intensity of a camera pixel is the sum of the calibrated intensity values for all the channels of the projected pixel. A black projected pixel will not correspond to a black camera pixel due to the white projection surface combined with light emanating from the projector when it is projecting black. A sum of three camera channels will thus have two extra black pixel values if the channels since the projector light will only be additive above the threshold of the black camera pixel.

2.2.2 Harris Detector

Points of interest are identified by applying a Harris corner detector [5] and non-maximal suppression [3]. The Harris detector is a simple feature locator that provides higher responses in areas where gradients in multiple directions and appear as corners versus intersecting edges. Non-maximal suppression finds the highest valued element in the Harris image within a 5x5 window and identifies it as a possible feature location. Many possible features can be found this way as a result of noise in relatively flat areas, so a threshold requires a minimum Harris response value. Features are described by useful parameters for normalized correlation: an 11x11 image patch for each feature and its sum, sum squared and a coefficient used in the matching process.

The Harris detector produces a response image that contains peaks where the intensity gradient has more than one direction and appears like a corner. Depending on the tuning of the Harris detector's parameters the response can range from appearing like an edge detector to returning very sharp peaks. Striking a good balance is key to extracting enough quality features to ensure matching and, ultimately, geometry estimation are successful.

Figure 12: Features detected for the camera and expected camera frames.

For each pixel the Harris detector calculates a 2x2 gradient covariance matrix M [5]. This represents how strong of dependency there is between the gradients around the pixel in a certain neighborhood. Smaller neighborhoods require less computation time, so the smallest value of 3 was chosen. M is then used to calculate the response at that pixel by the equation below. If the eigenvalues of M are strong, then a corner is detected. If the eigenvalues are weak, then the image is approximately uniform in this area. Asymmetrical eigenvalues indicate an edge.

$$HarrisPixel = \det(M) - k * trace(M)^2 \quad (8)$$

The parameter k is set to 0.6 as a result of comparing matching results for changing values of k. In a sense, k tunes the strength at which the eigenvalues of M cause the Harris response to start peaking. Lower values of k cause a stronger overall Harris response, but this is not desirable as the responses become far more prolific for the same threshold while also emphasizing image features that are less and less corner-like.

To find feature locations from the Harris image a non-maximal suppression step identifies local maxima in the image. All the pixels in a 5x5 neighborhood are checked and the location for the pixel with maximum values is recorded provided this maximum is above the minimum threshold. Harris responses are rather low in magnitude, so thresholds are set around 0.00001. The expected camera image tends to produce a stronger response than the camera image, so its threshold is a little higher and the camera's is a little lower.

2.2.3 Matching

Two feature sets are now available, one from the camera image and one from the reconstructed camera image. Comparison between features starts by finding features in the reconstructed camera set that lie within a certain disparity window of a feature in the camera set. Normalized correlation scores are calculated for every feature that falls within this disparity window. The feature being checked for a match is then paired with its candidate match: the match with the highest correlation score. A final match is determined by a mutual consistency check [1]. The candidate match (from the expected camera image) is compared with the features from the camera set. If the feature most closely matching it from the camera set is the original feature then a match is verified.

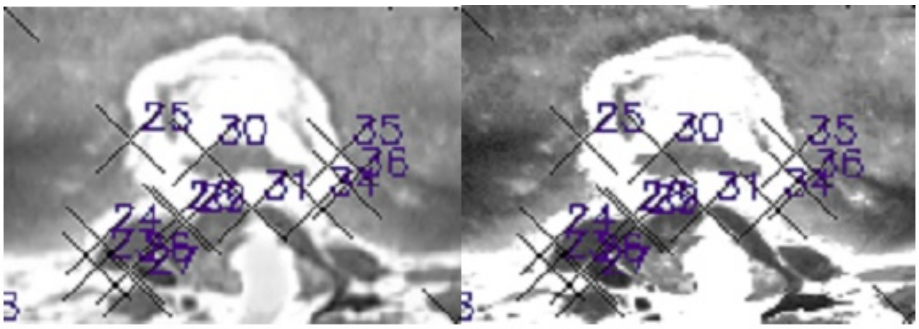

Figure 13: Feature matching result. Nearly all found matches using this algorithm are visually accurate.

The disparity window is used to limit the number of features being scored in each matching attempt. The camera image and expected camera image are not likely to have very different appearances due to the slight deformations of the image that accompany comparatively large changes in the 3D orientation of the display surface in space. This makes it reasonable to assume that valid matched features will be spatially close in the two images. Valuable time is saved by eliminating most of the features from the scoring process and focusing on those features most likely to be matches.

Normalized correlation is a calculation on image patches. In the broadest sense the score could be computed directly from two image patches, but such a scheme would waste enough time to kill any prospect of real time interaction. As described above, the feature description is primarily made to aid and speed up the process of matching. Each feature contains values that are needed every time a normalized correlation score is computed: the sum of the image patch values, the sum of squared patch values and a constant coefficient. To compute normalized correlation with two such well equipped features, one must only compute one additional variable: the sum of the products of the values in the two image patches being tested. The equations below show the details of normalized correlation computation [3].

$$NC = (121 * D - Sum_1 * Sum_2) * C_1 * C_2$$
$$C = \frac{1}{\sqrt{121 * SumSq - Sum^2}}$$
$$D = \sum I_1 * I_2 \qquad (9)$$

The calculation of *D* in the above formula is reserved for features sharing a disparity window. This multiplication is the most computationally expensive part of the matching algorithm as all other values in the *NC* equation were computed as part of feature creation. From this point all the features found within the disparity window are tested with normalized correlation. Once the highest scoring feature is found a mutual consistency check ensures that the match is supported by both features. If both features have the highest normalized correlation score with each other out of their respective disparity windows the probability of a good match is high. As a last check, a threshold is applied to the *NC* values.

Matched features form the backbone of the estimation process. One of the best metrics for determining the success of the algorithm is the quality of the match points. A good match point correctly connects the expected and actual camera images, providing the basis for a geometric estimation. Match points are triangulated to obtain 3D points which represent the location of the display surface in the last part of the algorithm, but it is not possible to maintain interactive framerates by directly estimating from all available match points. Ideally a smaller set of match points could be triangulated to find the surface geometry in 3D while remaining accurate. For this reason match points are refined to a smaller representative set before they are triangulated for making a geometric estimate.

2.3 Geometry Estimation

With a set of matches calculated, the geometry estimation portion of the algorithm seeks to quickly filter the match data into a geometric estimate of the display surface location. The first step is to refine the match points to a representative set through RANSAC. This smaller set of points can then be

triangulated to four 3D points. Plane fitting produces a normal and centroid that can be used to determine a projector image for the newly estimated display surface location.

2.3.1 Preemptive RANSAC

An algorithm must be used to reduce the number of match points used to form an estimate of the plane geometry. The random sample and consensus approach (RANSAC) provides a framework where random samples of the match points can be compared with other match points to find a consensus on which random sampling best represents the overall set. This method has been used time and again to solve problems where robust estimation is needed to find a set of useful inliers among data containing both inliers and outliers.

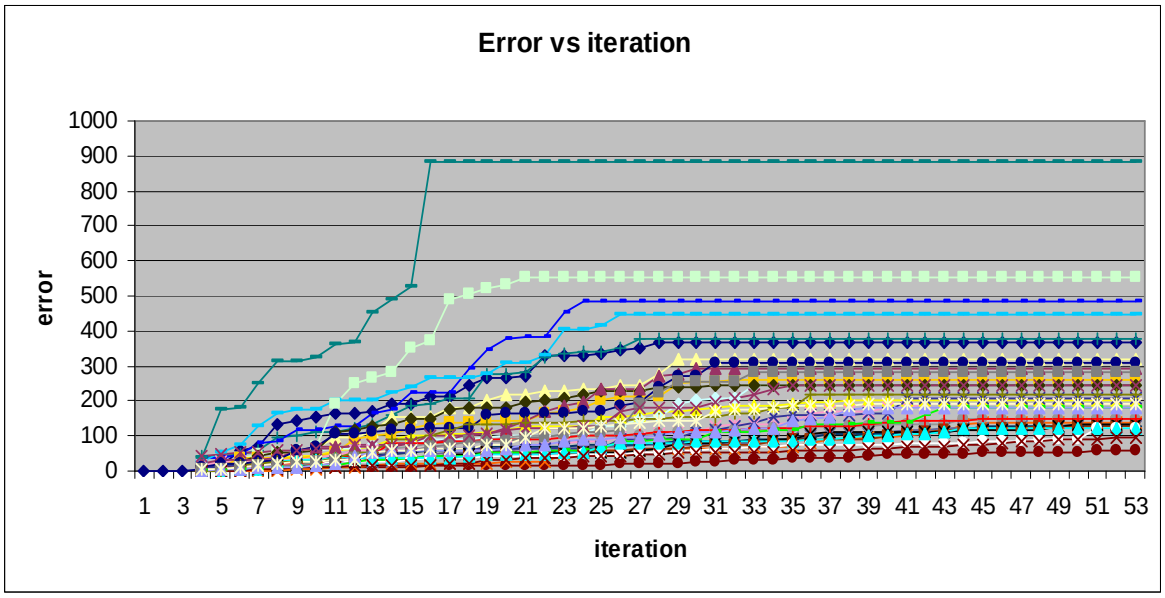

Figure 14: RANSAC error over a large number of iterations. These hypotheses represent the lowest error group of hypotheses for this run. Higher error hypotheses were removed to show separation. When an error track becomes flat and others are below it, it has been preempted by the algorithm.

Each hypothesis is composed of four match points. This makes it possible to calculate a homography from the expected camera image to the camera image and use it to evaluate the performance of the hypothesis. The homography generated by

a set of four such matches can be used to check how well other match points line up with them. This homography represents a perspective transformation from the 2D frame of the expected camera image to the camera image and is invertible to transfer points from the camera image to the expected camera image. Inliers would show low reprojection error when this homography is used to project the match from one frame to another. Any match not used to form the hypothesis can thus be used to evaluate the hypothesis through cumulative reprojection error. A match from the hypothesis set will have a reprojection error of zero. The formulation of the error accumulator is:

$$\sum(x - x_r)^2 + \sum(y - y_r)^2. \quad (10)$$

The Preemptive RANSAC algorithm [4] uses this as a method for finding an optimal estimation of the geometry in a minimal amount of time. After a number of hypotheses have been formed, preemptive RANSAC uses the reprojection error of randomly selected match points to accumulate scores for the hypotheses. After each random match is used to score the hypothesis set, some of the hypotheses with the highest errors are discarded to minimize the amount of calculation needed to arrive at the final estimate. The result is that significant outliers and poor performing hypotheses are rapidly rejected. Figure 13 shows this as flattening out of the error curve due to the algorithm skipping over the hypothesis.

The number of hypotheses H influences the time to execute the RANSAC algorithm more than other factors because each hypothesis requires the generation of a homography. Each round of RANSAC must eliminate N hypotheses, leading to a total of H/N rounds before a final estimate is reached. The hypothesis with the lowest error is determined to be the winner and represents the newest geometric estimate.

First RANSAC produces all the necessary hypotheses. For each hypothesis four random points are selected. Four matches in two frames of reference are enough to generate a homography between the two frames. This homography will indirectly

determine how well other matches fit the plane defined by these four randomly selected matches. Because points must be transferred between both frames, the homography's inverse is also calculated. To begin scoring a random match point is selected and the points from either reference frame are projected to the opposite reference frame (from camera image to expected camera image and vice versa) using the hypothesis homography and inverse. If a hypothesis and random match point line up well then the projections of these points would line up with their match counterpart. This is rare; some error will exist for any match except the ones used to generate the homography. Reprojection error is calculated as the sum of squared differences between the matches and the projected points in their own frames. Large reprojection error indicates that the current match and hypothesis are not compatible. The error for all hypotheses are measured against this single observed match and the *N* hypotheses with the highest error are discarded. Another random match is selected and the process is repeated until *N* hypotheses can no longer be removed from the set, then the lowest error hypothesis is selected as the estimate.

This adaptation of the generic RANSAC algorithm provides the advantage of taking a nearly fixed amount of time each frame. With preemptive RANSAC estimation the system is guaranteed a quick, representative estimate from the available match point data.

2.3.2 Triangulation and Plane Fitting

The final step is to convert the set of four matches into a form representing a new geometric estimate. First the match point locations in the expected camera image are warped to the projector frame. This can be performed by calculating the homography from the camera frame to the projector frame. Since a 3D geometry is given, it can be projected into each frame providing the four correspondences needed to calculate the homography. Triangulation is performed on the correspondences between the camera and projector frames by way of their stored calibration data. With four 3D points a plane can be estimated and a normal and

centroid are found to define the location of the display surface in space. At this point a geometric estimate is finalized by calculating the plane's intersection with the volume extents and the resulting values are stored in the geometry buffer. Geometry buffer elements are smoothed across two frames before being used to display the projector image and are stored in the geometry buffer.

The triangulation method used here is based on singular value decomposition. Using OpenCV's SVD function. The 3D to 2D projection matrices of the camera and projector provide a map to 2D, but as 3x4 matrices they are not simply invertible to go in the other direction. Instead the two matrices are used together to find a path to 3D. A matrix A is formed that composes the information in the camera matrices with the match data and the smallest singular vector of A corresponds to the 3D location of the match point [1].

$$A = \begin{bmatrix} x_1 * p_1^3 - p_1^1 \\ y_1 * p_1^3 - p_1^2 \\ x_2 * p_2^3 - p_2^1 \\ y_2 * p_2^3 - p_2^2 \end{bmatrix} \quad (11)$$

In this formulation of the SVD matrix A, x and y refer to the locations of the match points for two different frames and p represents different rows of the two camera matrices. The smallest singular valued vector of A corresponds to the 3D location of the match point on the display surface.

The next step in this processing chain is to find the normal and centroid of the triangulated 3D points. A standard method is used to calculate the normal and centroid. The centroid is simply the average of the points. The normal is solved from

$$\boldsymbol{n} \cdot (\boldsymbol{X} - \boldsymbol{c}) = \mathbf{0}, \quad (12)$$

where $\boldsymbol{n}$ is the normal, $\boldsymbol{X}$ is the set of points on the plane and $\boldsymbol{c}$ is the centroid of the data. This equation is easily solvable with SVD. The normal found in this step is stored in a buffer to be used for smoothing. This is the raw geometric estimate of the

orientation of the plane in 3D space. The average for all the normals in the smoothing buffer is calculated as the smoothed normal and this vector defines the geometry that will be shown on the display surface. Averaging normals across frames helps to mitigate the effect of intermittent poor estimates. The smoothed normal is now converted into a set of 3D points that are the corners of the dataset where the display surface cuts through the volume in real space. These values are added to the geometry buffer and a new estimation cycle is begun.

2.3.3 Geometry Buffer

By implementing a simple three position buffer, using a cheap DV camera becomes possible. The geometry buffer delays the use of a calculated geometry estimate by three frames. The assumption is that the camera has an internal buffer structure that shifts images from the CMOS imaging array to the firewire bus where the computer reads the images. A three geometry delay works to keep the camera images synced to the geometric estimates even if the per frame time increases or decreases.

Before an estimation of the display surface geometry is made the camera image is acquired and an expected camera image is created from the estimated geometry of the display surface and the projected image. Using the commodity camera causes trouble here as the camera will not return the current image when asked for an image of the scene. Details on the camera's internal hardware specifications and how frames are moved from the CMOS sensor to the firewire bus are not available. From empirical observation, however, the camera has an internal buffer of three images. When the user polls the camera for an image, the camera acquires a new image, but places an older image stored in the last buffer location onto the firewire bus. It is possible to make multiple image polls to force the camera to catch up, but results in a very slow algorithm. To compensate, a geometry buffer is implemented so that every estimation is made using geometry estimated three frames before the currently available camera image. The result is that the camera images line up temporally with the geometries used to construct the data appearing in them. When

an expected camera image is constructed using this geometry it will closely match the actual camera image and produce valid match points.

It is possible for an estimation to fail, and this has consequences in the geometry buffer. If an estimate is attempted on an image pair and for some reason too few matches are found then the algorithm does not record a new geometry in the buffer. The result is an attempt to use the geometry again in the next frame against a camera image that should belong to the next geometry in the buffer. For this reason when an estimate fails no shifts are made in the buffer. If estimation continues to fail the buffer will retain its current contents until enough matches are available to make an estimate. If the orientation can be recaptured by the user, then the system will begin estimating new geometries again.

Table 3: How the geometry buffer works. At startup some frames are used to estimate the geometry without storing the estimate, subsequent estimates are added to the geometry buffer. The latest estimate is used to calculate the current display once new estimates are stored.

Current Estimated Geometry	Current Displayed Geometry	Geometry Camera Sees	Geometry Compared	Buffer Contents
B	0	0	0	[0 0 0]
C	0	0	0	[0 0 0]
D	D	0	0	[0 0 D]
E	E	0	0	[0 D E]
F	F	0	0	[D E F]
G	G	D	D	[E F G]
H	H	E	E	[F G H]

2.4 Data Preparation and OpenGL

This is a volume data visualization system that uses a set of large images to form a 3D dataset. These images must be processed in order to be useful to the OpenGL 3D texture framework. A 14 gigabyte set of numbered images is thus downsampled into a texture object of approximately 100 megabytes. This allows the 3D texture to fit on the video card's memory while leaving enough texture memory for displaying 2D textures in the scene. An OpenGL scene is a virtual 3D environment where

graphics primitives can be positioned and textured to form arbitrary 3D scenes. A scene is rasterized into an image based on the assumption of a specific view orientation and displayed on the screen, giving a view of the 3D environment.

The OpenGL scene for this system is as simple as possible to ensure quick processing. A plane is created orthogonal to the view direction so that it extends beyond the viewable area. This is an OpenGL primitive called a quad. This quad is the only geometric primitive used in the scene, and the scene does not change during the operation of the system. The effect of altering the projected image to fit a specific display surface orientation is achieved by texturing the dataset image on this quad within the bounds of four points defined by the projection of the volume extents in 3D to the projector frame.

The same scene is used to form the expected camera image. As a result of the DV camera and geometry buffering process, it isn't possible to simply take the current framebuffer and warp it to the camera frame and directly compare them as the current camera image is older than the current projector frame. This means that an extra OpenGL display call is made which rasterizes the textured quad into an image.

CHAPTER 3

RESULTS

3.1 Calibration

Geometric calibration can be measured by the mean squared error between the locations of 2D points found in a calibration image and the projection of their corresponding 3D locations into the image frame. Error between points tends to increase toward the outer edges of the image, but remains low for both the camera and projector.

Table 4: MSE of x and y directions for the camera and projector over 3 calibration runs.

Calibration Run		*x* MSE	*y* MSE
Run 1	Camera	0.118228	0.081468
	Projector	0.06583	0.118696
Run 2	Camera	0.124685	0.080831
	Projector	0.057336	0.099057
Run 3	Camera	0.117633	0.083330
	Projector	0.088556	0.176183

Figures 14 and 15 show the absolute error in pixels of the calibration pattern. Each 2D point was compared to the projection of the corresponding 3D point into the image frame. The result is that nearly all the points fall below one pixel of error while mostly remaining within 0.5 pixels of error.

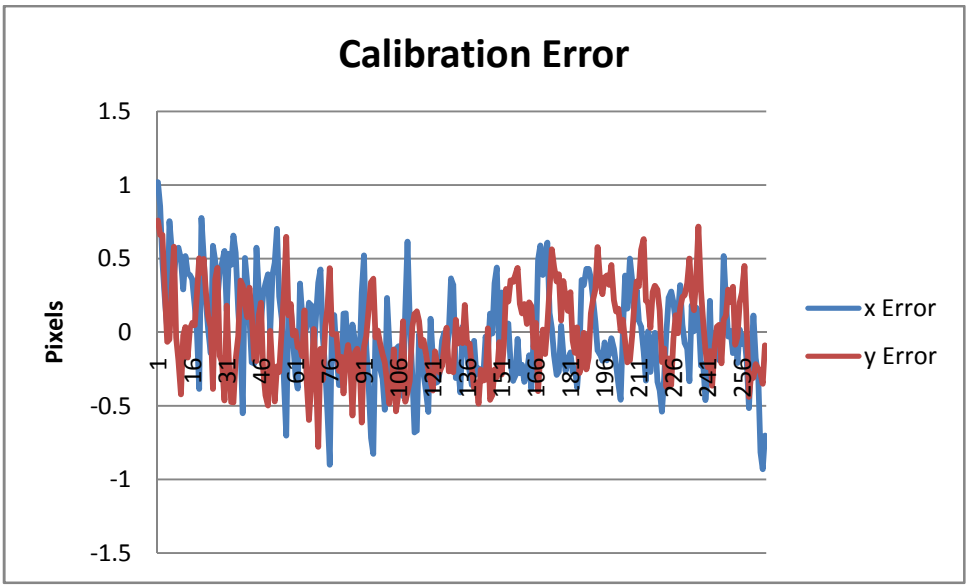

Figure 15: Absolute x and y error from camera calibration.

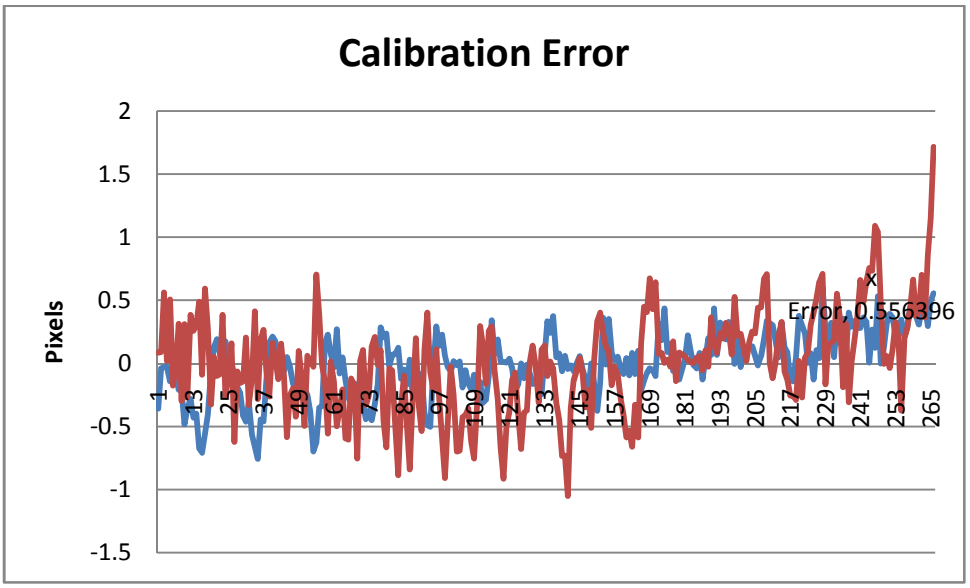

Figure 16: Absolute x and y error from projector calibration.

3.2 Feature Matching

Feature matching accuracy can be visually discerned by taking a sample and individually counting correct matches and incorrect matches. Incorrect matches will be obvious outliers and correct matches will be likely inliers. The ratio of correct to incorrect matches provides a good metric for evaluating performance of the steps leading up the feature matching stage. A minimum of four matches can still produce an estimate, but estimates are not made unless more than 5 matches are available. This provides the ability to estimate the geometry even under occlusion.

Matching is affected by ten parameters which change the makeup of feature locations and how closely the system requires image patches to match. The Harris feature detector has a variable block an aperture size which determine how large of a neighborhood is used to find corners and how far outside of the local neighborhood the image affects the corner response respectively. These parameters are fixed at their minimum requirements for speed. A larger window and aperture allow for more consistent feature locations, but also reduce the number of features with high corner responses at the cost of processing time. A larger number of high corner responses give a more even distribution of feature locations after the non-maximal suppression step. The Harris operator has a parameter *k* that can change the output to resemble an edge detector when its value is low and a strict corner detector when the value is high. Varying this parameter has little effect on match accuracy (except at extreme values), but the number of matches found in a given frame will be lower if the value is set higher. In the UMB, the value for *k* is set at 0.6, as it is in this work. Higher and lower values affect orientations of the display surface farther away from the camera the matches become less accurate despite large numbers of them or the number of matches declines to the point of generating more noisy hypotheses.

Table 5: Total matches and good matches for four values of *k*, taken for four orientations of the display surface.

k =0.01		*k =0.04*		*k =0.06*		*k =0.1*	
Total Matches	Good Matches	Total Matches	Good Matches	Total Matches	Good Matches	Total Matches	Good Matches
135	123	63	59	69	68	26	26
101	89	86	85	86	84	34	34
70	64	37	33	75	73	37	37
83	72	47	39	40	39	20	20

Table 6: Total matches and good matches for four values of the match threshold, taken for four orientations of the display surface.

M_THRESH =0.9		*M_THRESH =0.94*		*M_THRESH =0.97*		*M_THRESH =0.99*	
Total Matches	Good Matches	Total Matches	Good Matches	Total Matches	Good Matches	Total Matches	Good Matches
174	61	88	52	61	59	10	10
169	83	93	48	53	51	8	8
133	54	73	47	41	40	15	15
145	65	99	56	32	32	12	12

3.3 Geometry Estimation

RANSAC must provide a reasonable representative set in the shortest time possible. The number of hypotheses *H* influences the time to execute the RANSAC algorithm more than *N* because each hypothesis requires the generation of a homography. Each round of RANSAC must eliminate *N* hypotheses, leading to a total of *H/N* rounds before a final estimate is reached. The hypothesis with the lowest score is determined to be the winner and represents the newest geometric estimate. This happens in a nearly fixed amount of time each frame due to the specific number of calculations made.

By choosing a large *N*, the total number of scoring operations will be lower, but the results may not be as stable as lower values. If *H* is large, then the amount of time computing the winning hypotheses increases. If *H* is small, then there may not be enough hypotheses generated to guarantee a representative set.

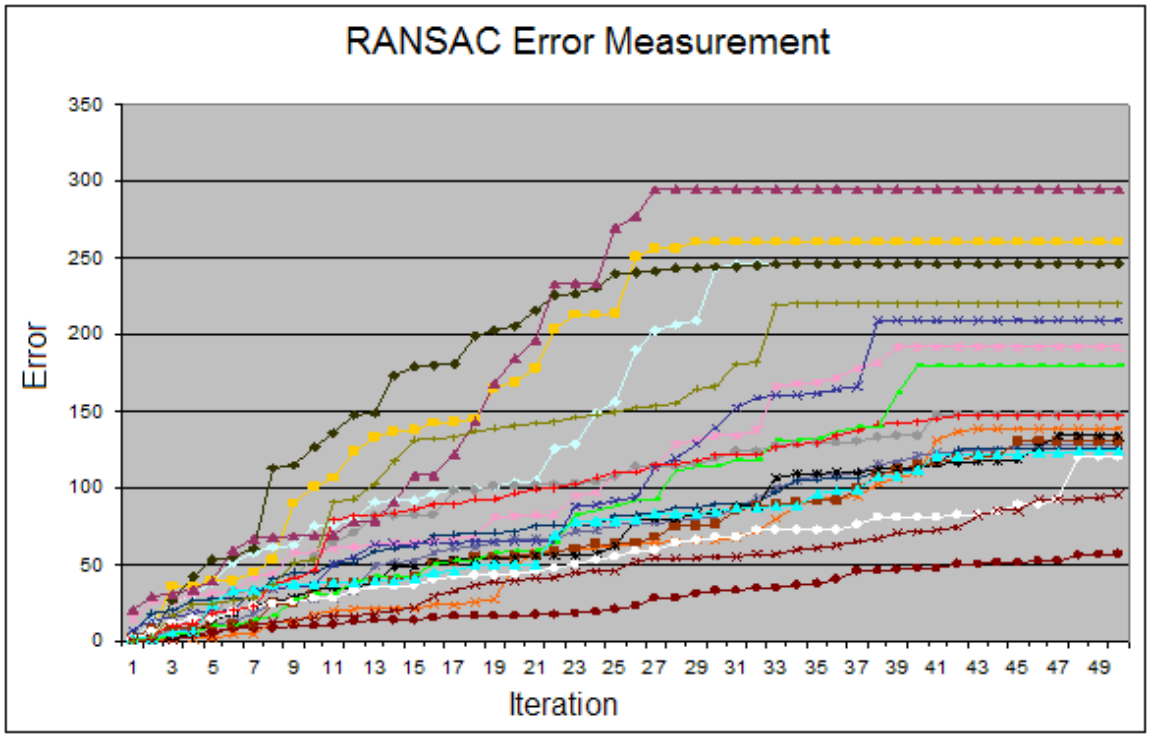

Figure 17: A close-in look at the lowest scoring hypotheses in a RANSAC run. Preempted hypotheses flatten out toward the right end of the graph.

The result of the RANSAC refinement represents the new geometric estimate of the location of the display surface. The first step to translating the match points to the estimate is to triangulate the match points to 3D points. In this stage calibration error is injected. Also the triangulation method is linear. To increase the accuracy of linear triangulation, an undistortion map generated from the distortion coefficients is used to correct for lens distortion before triangulation. The result is that a triangulated 3D point will be conservatively accurate to within one unit of the calibration pattern. The units are measured from one corner to its neighbor on the pattern. Figure 17 shows this and that the z error dominates. Each unit is approximately one inch and the dimensions of the dataset in space are 19 x 14 x 55.

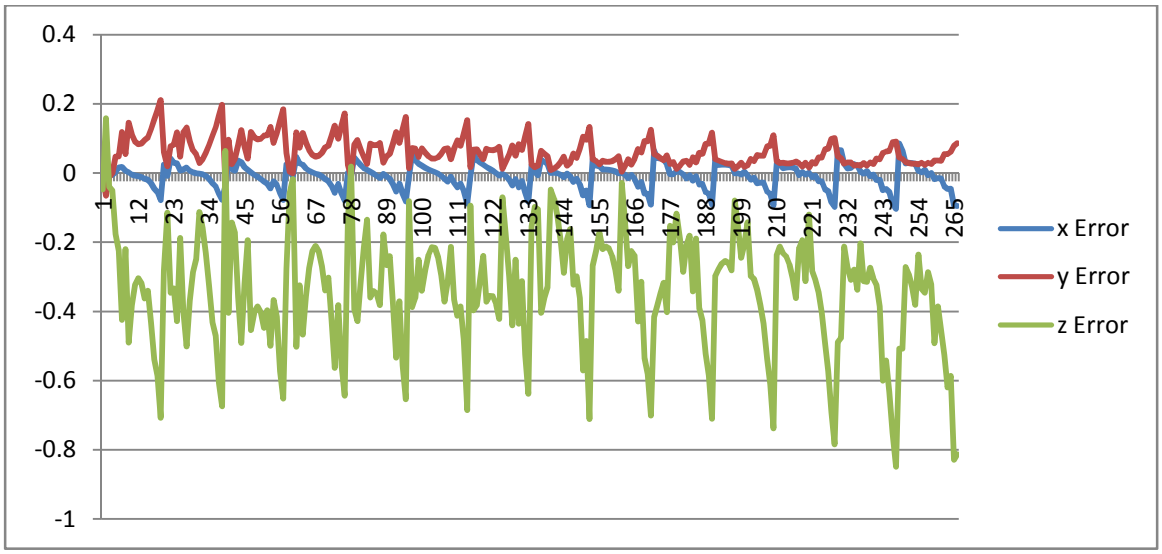

Figure 18: x, y and z error from triangulating the calibration pattern locations from camera and projector frame locations. The vertical axis represents the distance between two corners on the calibration pattern.

The triangulated 3D points are then used to compute a normal and a centroid. From these the coordinates of the intersection of the display surface with the volume extents can be computed which becomes the final geometric estimate. These estimates are smoothed by two frames as this helps to stabalize the display image and does not noticably slow the algorithm.

3.4 Overall Performance

The algorithm runs at between 3 and 7 frames per second. This is sufficient for interacting with the display and navigating to specific locations within it. Table 7 shows a significant startup time of 14 seconds followed by a framerate of somewhat less than 5 fps.

Table 7: Frame times and startup time. (milliseconds)

Frame	0	1	2	3	4	5	6	7	8	9	Ave
Time	13875	203	203	204	203	187	235	202	204	250	210.11

The framerate is allowed to float and can be adjusted somewhat by tuning parameters. The above example of frame times is typical for stable operation with little jitter while the display surface is being held in a particular place. Most runtime framerate variance comes from the number of features generated for a given frame. If the display surface is closer to the camera then both the camera and expected camera images take up more of the frame, leading to more features. With more features it is also more likely that there will be more matches, so the matching phase will also take more time in this situation. Once the algorithm reaches the RANSAC phase there aren't any factors that change execution time per frame. Transient effects can also cause frame times to change. If the camera changes white balance then the system may lose tracking until the balance change is finished due to a washed out camera image. The system is robust to occlusions that occur close to the display surface, even to the point of covering most of the important data, but occlusions between the camera and surface can cause the focus to change enough to lose tracking, also causing the frame time to increase until the focus is stable. Dramatic changes in illumination, such as turning on the room lights, cause similar effects, often resulting in the system losing the ability to track further.

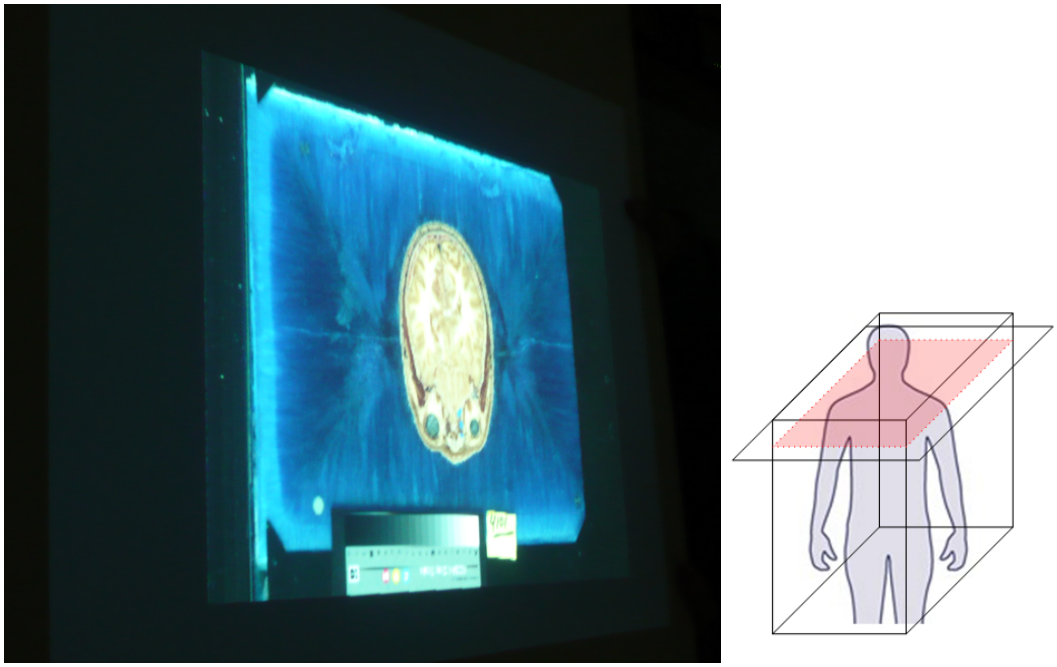

Figure 19: Display surface in default position. Cross section shows the brain and eyes. The figure to the right shows the relative location of the cross section in the volume data if the human is facing to the right.

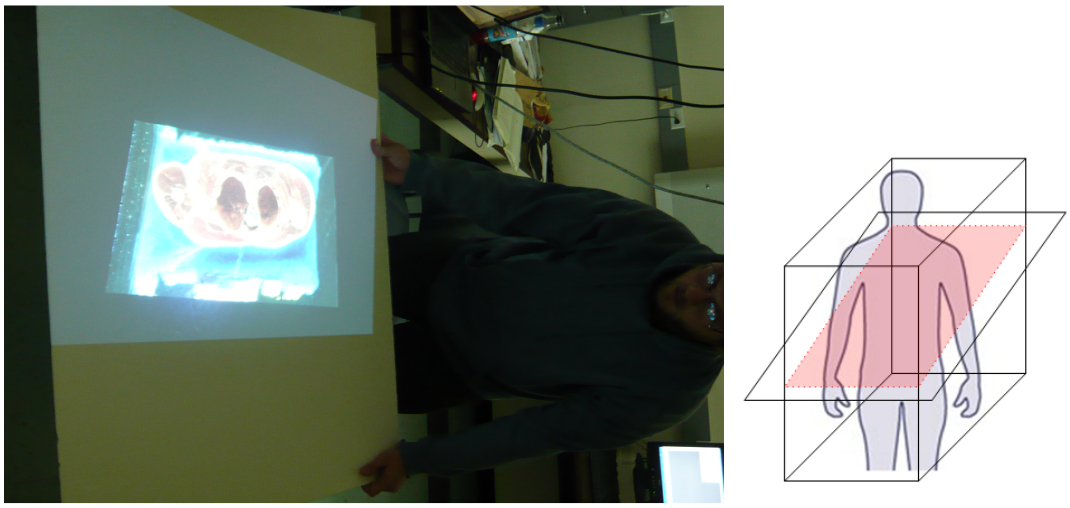

Figure 20: Display surface in action. The cross section shows an angular cut through the left shoulder, lungs, heart and right upper arm.

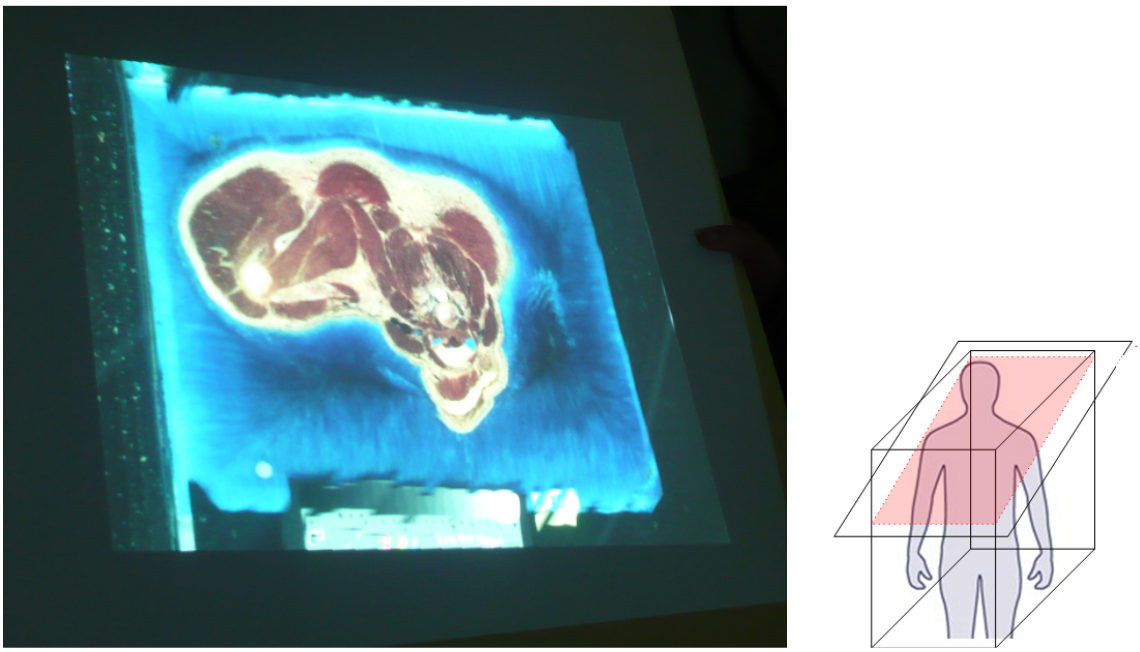

Figure 21: Angular cut through the neck and right shoulder.

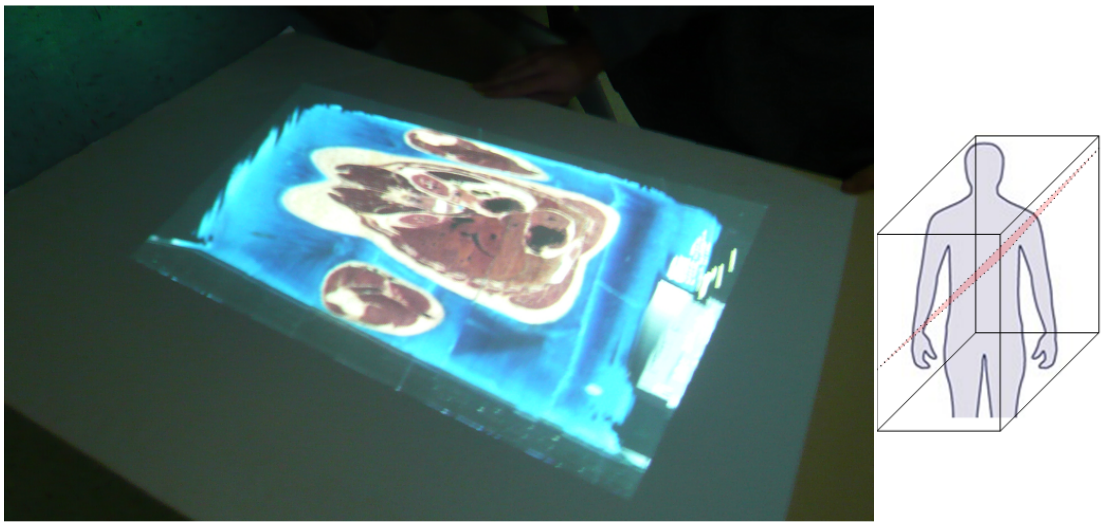

Figure 22: More vertical slice showing the lower back, elbows, liver, kidneys, and heart.

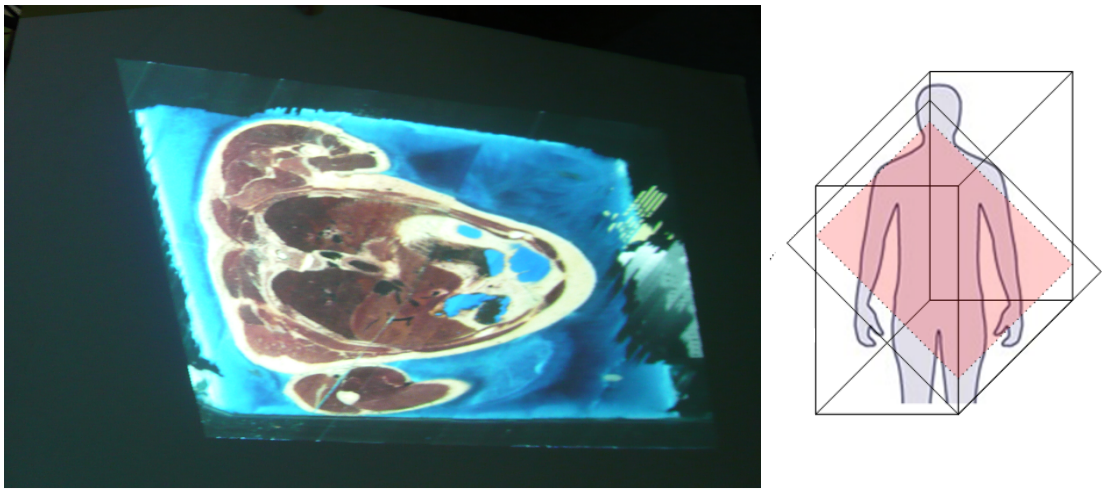

Figure 23: Slice from opposite angle, showing the muscles around the collar bone, lungs and some of the upper intenstine.

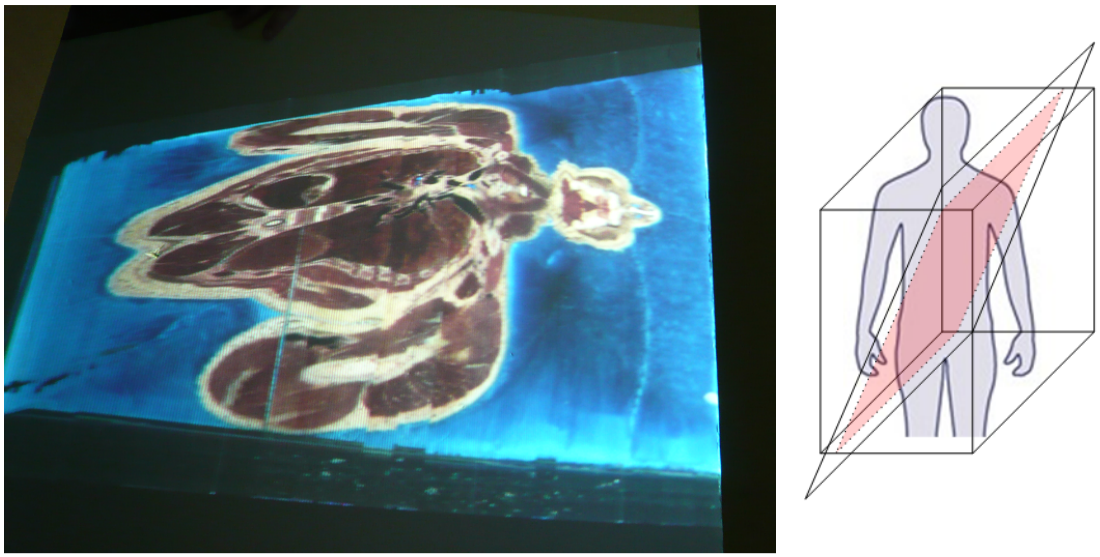

Figure 24: Near vertical slice showing the lower back, spine, kidneys, lungs, neck, jaw and nose.

CHAPTER 4

CONCLUSIONS AND FUTURE RESEARCH

A volume visualization system is implemented allowing a user to interact with data in an intuitive, hands-on way. The system is implemented on commodity hardware: a laptop with an adequate graphics card, a common presentation projector and a DV camera. Publicly available software libraries were used to implement parts of the system under windows using the Visual Studio IDE. Challenges related to the use of a DV camera are explored and overcome using a geometry buffer to fix temporal mismatch and a new color pattern for photometric calibration. Implementation details and the effects of implementation decisions on performance in the context of the overall system goals are explored, illuminating the most critical aspects.

4.1. Conclusion

There are two measures to the quality of the volume visualization system. First is accurate estimation and second is framerate. If estimation is too noisy then the user will be unable to gain any useful experiences from using the system as it may not appear to even function correctly. If the estimation process runs too slowly then the user will find it frustrating and difficult to use for visualizing the dataset. The system nominally operates at between 3.5 and 7 frames per second on a 2 GHz Intel Core Duo laptop with an NVidia Quadro 110M graphics accelerator. The

display can be moved at interactive rates to navigate the dataset. The overall noise level depends on environmental factors such as room illumination, but the system remains robust and able to track reasonably paced motion in most conditions while located in a room subject to variable daylight.

4.2 Future Work

This system could be modified to provide more tools, such as the ability to save frames of data from the set or to change the orientation of the volume in 3D space. With these a user could move the volume to a desirable position and navigate to an interesting area where images can be acquired. Locking the display in a specific position would be useful for adding features like the ability to draw on the data. Implementing such tools would require an additional input method. If the display surface is in the hands then the interface should use the display surface through some form of occlusion detection. Currently occlusion only hurts estimations when the occluded area covers most of the complex portions of the image, so the main idea would be to find a way to use what is already available in the algorithm and minimize the processing cost of detecting the input.

Frame rate performance could be increased slightly through small changes in implementation details, but an interesting option for a major boost would be to swap the Harris detector for the FAST feature detector [11]. This could dramatically decrease the amount of time spent per frame finding feature locations. The authors claim that their method is many times faster than the Harris detector specifically and can function the same way, returning the locations of features in the image. It would be interesting to see if matching the FAST features with the normalized correlation method would return as good of results as the Harris detector.

Another interesting route to follow in improving the system would be to model the motion of the display surface over time. This can be useful for two reasons: speed and smooth frame transitions. The motion model would be used to refine the

estimates produced by adding information about the expected location of the display surface given its historical location. If the motion model is accurate, then it could be expected that each frame would have a larger number of matches. The data area processed could be reduced under these conditions, reducing the amount of time required per frame. If the number of matches found is lower than a threshold then the area could be increased again as the motion model updates to increase accuracy. Adding the motion model means the system would be predicting the location of the display surface where before it was simply an observation. This creates the possibility for over and undershoot and the system becomes more like a control system. The estimator provides input to the motion model that predicts the variable user input before producing output. The output is then used as input data for the estimator. This implies that the system could over and undershoot on the estimate if the display surface is moved too quickly or at a frequency beyond half the frame rate. These weaknesses would be the tradeoff for the possible performance gain of adding the motion model.

REFERENCES

[1] S. Gupta and C. Jaynes, "The Universal Media Book: Tracking and Augmenting Moving Surfaces with Projected Information," 2006.

[2] S. Gupta and C. Jaynes, "Active Pursuit Tracking in a Projector-Camera System with Application to Augmented Reality," 2005, pp. 111-111.

[3] D. Nister, O. Naroditsky, and J. Bergen, "Visual odometry," 2004, pp. I-652-I-659 Vol.1.

[4] D. Nister, "Preemptive RANSAC for live structure and motion estimation," 2003, pp. 199-206 vol.1.

[5] C. Harris and M. Stephens, "A Combined Corner and Edge Detector," in Alvey Vision Conference, 1988.

[6] D. Nister, "An efficient solution to the five-point relative pose problem," Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 26, pp. 756-770, 2004.

[7] R. Tsai, "A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses," Robotics and Automation, IEEE Journal of [legacy, pre - 1988], vol. 3, pp. 323-344, 1987.

[8] M. B. Vieira, L. Velho, S. Asla, and P. C. Carvalho, "A Camera-Projector System for Real-Time 3D Video," 2005, pp. 96-96.

[9]Z. Zhang, "A flexible new technique for camera calibration," Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 22, pp. 1330-1334, 2000.

[10] T. A. DeFanti, G. Dawe, D. J. Sandin, J. P. Schulze, P. Otto, J. Girado, F. Kuester, L. Smarr, and R. Rao, "The StarCAVE, a third-generation CAVE and virtual reality OptIPortal," Future Generation Computer Systems, vol. 25, pp. 169-178, 2009.

[11] Rostin, Edwared and Drummond, Tom. “Machine Learning for High-Speed Corner Detection,” Springer Berlin, vol. 3951, 2006.

VITA

Jason Graham

Candidate for the Degree of

Master of Science

Thesis: IMPLEMENTATION OF AN INTERACTIVE REAL TIME VOLUME VISUALIZATION SYSTEM USING COMMODITY HARDWARE

Major Field: Electrical Engineering

Biographical:

Personal Data:

Education:
Bachelor of Science in Electrical Engineering, Oklahoma State University, Stillwater, May 2005.
Completed the requirements for the Master of Science in the Electrical Engineering Department of Oklahoma State University, Stillwater, Oklahoma in July, 2009.

Experience:

Fall 2007 – Spring 2008 : Laboratory Teaching Assistant for Experimental Methods 3 at Oklahoma State University
Fall 2007 – Fall 2008 : Research Assistant MSVLSI
Spring 2009 – Summer 2009 : Research Assistant VCIPL

Professional Memberships:

Student member of IEEE

Name: Jason Graham Date of Degree: July, 2009

Institution: Oklahoma State University Location: Stillwater, Oklahoma

Title of Study: IMPLEMENTATION OF AN INTERACTIVE VOLUME VISUALIZATION SYSTEM USING COMMODITY HARDWARE

Pages in Study: 56 Candidate for the Degree of Master of Science

Major Field: Electrical Engineering

Scope and Method of Study: System implementation

Findings and Conclusions:

An implementation for a volume data visualization system is provided. User interaction occurs through a markerless planar surface onto which images from a dataset are projected. The surface is moved through an area of 3D space where camera and projector views intersect and have been calibrated to a world coordinate system. A 3D dataset is defined as residing in this space and cross sections of the dataset are projected onto the display surface based on the position of the surface in the dataset area. A robust estimation algorithm provides the location of the surface based on the acquired camera image and expected camera image. This represents an overall real-time image acquisition, processing and update feedback loop. Only commodity hardware is used to implement this system. Using a common digital video camera complicates the important task of image acquisition, delaying the calculation of estimates. A method to overcome crippling slowness due to DV camera acquisition is described.

ADVISER'S APPROVAL: Dr. Guoliang Fan