# CARNEGIE MELLON UNIVERSITY

## ITERATIVE X-RAY/CT REGISTRATION USING ACCELERATED VOLUME RENDERING

A DISSERTATION
SUBMITTED TO THE GRADUATE SCHOOL
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

for the degree

DOCTOR OF PHILOSOPHY
in
ELECTRICAL AND COMPUTER ENGINEERING

by

David A. LaRose

May, 2001

**Keywords:** 2D-3D registration, frameless stereotaxy, computer assisted surgery, volume rendering, hardware accumulation

# Abstract

Recent years have seen exciting advances in Computer Assisted Surgery (CAS). CAS systems are currently in use which provide data to the surgeon, provide passive feedback and motion constraint, and even automate parts of the surgery by manipulating cutters and endoscopic cameras.

For most of these systems, accurate registration between the patient's anatomy and the CAS system is crucial: if the position of the surgical target is not known with sufficient accuracy, therapies cannot be applied precisely, and treatment efficacy falls.

This thesis presents a system for recovering the position and orientation of the target anatomy in 3D space based on iterative comparison of 2D planar radiographs with preoperative CT data. More specifically, this system uses X-ray images acquired at the time of treatment, and iteratively compares them with synthetic images, known as Digitally Reconstructed Radiographs (DRRs), in order to estimate the position and orientation of the target anatomy.

An intermediate data representation called a Transgraph is presented. The Transgraph is similar to the Lumigraph, or Light Field, and extends the computer graphics field called *image-based rendering* to transmission imaging. This representation speeds up computation of DRRs by over an order of magnitude compared to ray-casting techniques, without the use of special graphics hardware.

A hardware based volume rendering technique is also presented. This approach is based on new texture mapping techniques which enable DRR generation using off the shelf consumer grade computer graphics hardware. These techniques permit computation of full resolution (512x512) DRRs based on 256x256x256 CT data in roughly 70 ms.

The registration system is evaluated for application to frameless stereotactic radiosurgery, and phantom studies are presented demonstrating accuracy comparable to current immobilization-based systems. Additional phantom studies are presented in which the registration system is used to measure implant orientation following total hip replacement surgery, improving on current practice by a more than factor of two.

# Acknowledgements

The first thank-you goes to my advisor, Takeo Kanade, for years of support, for having confidence in my abilities, and for countless insights. It has been a privilege to work so closely with such a truly effective person.

I am deeply grateful to the members of my thesis committee, John Bayouth, David Casasent, and Russell Taylor, for their time and guidance through the course of my research. I owe a special thanks to Russell Taylor and John Bayouth. Russ literally started my technical career in 1991, and has been a role model ever since. I greatly admire his dedication and honesty. John provided hours of discussion, late night data gathering help, and encouragement, and has shown me that even in a world full of compromises it is possible to be happy and ethical at the same time.

Many people have contributed to my technical development at CMU. I am particularly grateful to Chris Paredis, Gary Ellis, and C. J. Taylor, without whom I would never have found my feet. Teck Khim Ng and Mei Han deserve a special thank you for their unconditional friendship and support, as well as for their outstanding technical insights. Daniel Huber, Sundar Vedula, Daniel Morris, Henry Schneiderman, and Devin Amin have been excellent colleagues through the last few years, and have added immeasurably to my experience at CMU, filling the large shoes which were left empty when Rahul Sukthankar, Mike Sipe, and Dirk Langer graduated.

Jim Hoburg has always been willing to share his considerable technical expertise, but more importantly has been a shining example of honesty and integrity. I aspire to be more like him.

I am indebted to the Mary Hillman Jennings Cancer Center at UPMC Shadyside for the use of data and facilities. I am similarly indebted to Branko Jaramaz, Tony DiGioia, and Jim Moody from the Center for Orthopaedic Research at UPMC Shadyside, who have been sources of inspiration and encouragement, and who provided all of the data for my hip implant experiments. Rich LaBarca and Costa Nikou have always been generous with time, expertise, and software. Laura Cassenti has been a strong and steady collaborator, consistently doing more than her share, and always ready with a smile. Thank you, Laura.

I especially thank David Simon for hours of technical and not-so-technical discussion, and also for restoring my confidence when it was at an all time low. Lynn Philibin, Elaine Lawrence, and Louise Ditmore have been my surrogate family at CMU, for the most part keeping me out of trouble.

Finally, it is important to recognize the people who, although not direct contributors to my work, have shaped my life so profoundly that it shows in everything I do. Very special thanks to to Tracy Logan, who is the standard by which I measure my own integrity, Edna Neivert, who has been the single most important and wonderful part of my life all these years, my brother Gavin, who has blazed the trail for me on so many occasions, and my parents, Albert and Barbara. Thanks, and I love you.

# Contents

# List of Figures

# Chapter 1

# Introduction

Recent years have seen many exciting advances in Computer Assisted Surgery (CAS). Maturing technologies in robotics, computer graphics, and data visualization have become directly applicable in the operating room. Successful CAS systems are currently in use which provide data to the surgeon through virtual fluoroscopy and virtual endoscopy. Other systems provide passive feedback by constraining the position of drill guides and other surgical tools, and still other systems actually automate parts of the surgery by manipulating cutters and endoscopic cameras.

In spite of these advances, registration between the patient's anatomy and the CAS system has remained a difficult task. Accurate registration is crucial: if the position of the surgical target is not known with sufficient accuracy, therapies cannot be applied precisely, and treatment efficacy falls.

This dissertation presents an algorithm for recovering the position and orientation of the target anatomy in 3D space based on iterative comparison of 2D planar radiographs with preoperative CT data. More specifically, this system uses X-ray transmission images acquired at the time of treatment, and iteratively compares them with synthetic images, known as Digitally Reconstructed Radiographs (DRRs) [50]. The DRRs are generated based on an estimate of the position and orientation of the patient's anatomy, and this estimate is progressively updated throughout the course of the iteration. In this respect, our system is similar to several existing systems [18] [42] [56].

DRR generation is essentially a computer graphics problem, and involves rendering images based on volumetric CT data. This is a computationally expensive process. The effectiveness of the iterative registration algorithm depends on both the speed with which DRRs can be generated and the methods by which they are compared to the input X-ray images. Our method differs from the others in that we compute DRRs on the fly using novel adaptations of computer graphics techniques, and in the method of image comparison.

This document presents an intermediate data representation called a Transgraph. The Transgraph is similar to the Lumigraph [20], or Light Field [36], and extends the computer graphics field

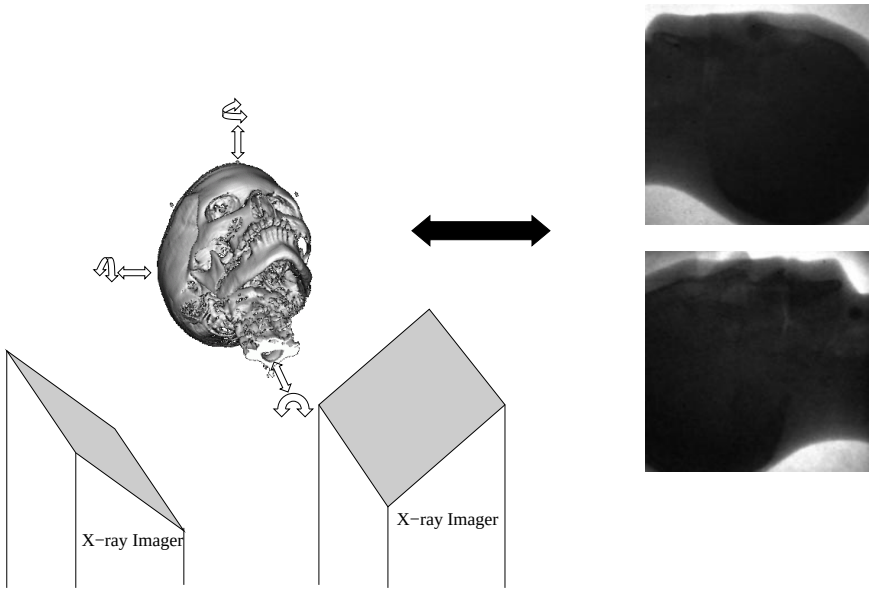



Figure 1.1: The goal of X-ray/CT registration is to recover patient pose based on information from one or more 2D radiographs and a preoperative CT scan.

called *view-based rendering* to transmission imaging. This representation speeds up computation of DRRs by over an order of magnitude compared to ray-casting techniques, without the use of special graphics hardware.

We also present new texture mapping techniques which enable DRR generation using off the shelf consumer grade computer graphics hardware. These techniques permit computation of full resolution (512x512) DRRs based on 256x256x256 CT data in roughly 70 ms. We anticipate a further reduction of at least 10ms with the next vendor supplied driver release.

## 1.1 Iterative X-ray/CT registration

Figure 1.1 illustrates the idea of X-ray/CT registration. The patient's anatomy, in this case a head, is in an unknown position and orientation between a pair of X-ray imagers. Each imager generates an image, shown at the right side of the figure. The goal is to deduce the pose of the surgical target using information from the pair of X-ray images and from a preoperative CT scan.

We assume an initial guess of the patient pose which is accurate to within a few centimeters of translation and approximately 10 degrees of rotation. Using this initial pose estimate, we compute a series of DRRs, which we compare with the input X-ray images. Based on the results of each comparison, we update the pose estimate. We repeat this until the image similarity reaches a maximum, and then return the updated pose estimate. This process is described in more detail in chapter 2.

We anticipate that volume based X-ray/CT registration will have many applications in computer assisted surgery, manufacturing, and product inspection. In this thesis, however, we consider only computer assisted surgery applications. We specifically evaluate the registration algorithm for application to frameless stereotactic intracranial radiosurgery, and to postoperative evaluation of acetabular implant placement for total hip replacement surgery. The assumption of an approximate initial estimate of patient pose is justified for radiosurgical applications by the existence of sufficiently accurate non-stereotactic registration procedures [25]. We obtain initial pose estimates in post-operative measurement of acetabular cup orientation using an interactive procedure which is described in chapter 8.

In frameless stereotaxy experiments using an anthropomorphic head phantom, we currently perform volume based 6D registration using full patient anatomy and images similar to those in figure 1.1 with RMS 3D registration errors of approximately 1.25 mm, and worst case 3D registration errors of under 3 mm. These results are comparable to the accuracy of stereotactic systems using immobilization devices [25]. Similar experiments for postoperative measurement of acetabular cup placement show RMS orientation errors on the order of $2^o$, significantly improving on the current state of the art [27] [28].

## 1.2 Prior Work

Current research in non- and minimally- invasive registration techniques applicable to CAS can be roughly divided into three groups: techniques which rely on external fiducials or surface information; techniques which measure the position of patient anatomy using X-ray imaging; and techniques which measure the position of patient anatomy using ultrasound imaging. The X-ray based registration algorithms can be further divided in two categories: techniques which perform feature based registration using local features such as curves, contours, or implanted fiducials; and techniques which perform pixel based matching using intensity information from larger areas of the image.

Several existing registration systems work by directly measuring external features or contours, and matching the external shape to precomputed models. Grimson presents a registration system which works by aligning a surface model of the patient's face with data collected from a laser scanner during treatment [21]. A conceptually similar technique is presented by Simon [51]. Taylor describes a technique in which registration is established by matching the position of anatomical features to corresponding coordinates in a preoperative CT scan [53]. We propose X-ray/CT registration as a supplement to these techniques, for those situations in which external anatomical features do not reliably indicate the position of internal structures.

Other authors describe registration algorithms in which ultrasound data are acquired and matched

with preoperative models of underlying anatomy [13] [26]. Ultrasound based registration is an emerging technology, and shows a lot of promise for clinical use. Each registration, however, depends on several ultrasound images, and current systems require manual acquisition of these images, making the registration procedure relatively slow.

Still other techniques rely on implanted fiducials which can be detected at the time of surgery. Gall measures the position of a tumor during proton beam treatment of intra-cranial lesions by tracking implanted radio-opaque fiducials[17], and Balter proposes the use of similar fiducials to track prostate motion [6]. Fiducials are located using direct physical measurement by Taylor [54]. All of these methods require that the fiducials be placed surgically, increasing the invasiveness of the procedure and in some cases requiring multiple surgical interventions.

Lavallée constructs a preoperative surface model of the relevant internal anatomy, which is localized using intensity contours from X-ray images [34]. Other feature-based registration techniques are described by Joskowicz [30] and Guéziec [22]. These techniques work well when the relevant anatomical structures contrast well with surrounding tissue. When noisy, low contrast images are used, however reliable contours and features are difficult to extract.

In many cases, resistance to image noise can be increased by considering large areas of the recovered images during the registration process. Adler implements a registration algorithm in which live images are compared to a library of precomputed DRRs [4]. Each pair of DRRs in the library corresponds to a known patient pose, and a pose estimate is recovered by interpolating in the neighborhood of the best matches. The drawback of this approach is that the size of the required DRR library becomes prohibitively large when 6 degree-of-freedom registration is attempted. Because of this limitation, the Adler's algorithm recovers only translational motion; rotation of the patient is not measured.

A specialized system for registering 2D and 3D angiography images is presented by Kerrien [31]. In this work, synthetic 2D images are generated from a 3D volume using maximum intensity projection. Images are compared using normalized correlation, and pose estimates are updated using a modified optical flow algorithm. Another specialized system which measures artificial joint implant position using using X-ray fluoroscopy images is described by Sarojak [48]. This system uses computer graphics hardware to generate silhouettes of the implant, and these silhouettes are compared with the input image in a pose estimation procedure based on simulated annealing. This system measures implant pose with respect to the imaging hardware. No attempt is made to recover the implant pose with respect to the surrounding anatomy.

Lemieux proposes a method based on iterative optimization which is provides the foundation for the work presented here [35]. We extend Lemieux's work by providing fast ways of computing DRRs, by proposing alternate image comparison metrics, and by using different search algorithms in the optimization. These extensions significantly increase the speed of registration. Other mod-

ifications to Lemieux's approach have been proposed by Gilhuijs [18] and Weese [56]. Gilhuijs' work describes an algorithm for fast DRR computation which restricts attention to specific regions of the CT dataset, while Weese performs registration using sub-volumes corresponding to one or more vertebral body. Murphy presents an algorithm where computation is performed only on small areas of the X-ray image [42]. Our work builds on all of these by enabling X-ray CT registration using full CT data over the entire image.

## 1.3 Dissertation Overview

The remainder of this dissertation is organized as follows:

Chapter 2 presents the iterative registration algorithm. Parameterizations of patient pose are introduced, and the corresponding coordinate transformations are developed. Image comparison functions are presented for measuring the similarity between input X-ray images and DRRs, and methods are presented for computing the gradients of the similarity functions with respect to patient pose parameters. Optimization routines are presented for finding the poses at which image similarity is greatest.

The iterative registration algorithm involves repeatedly synthesizing DRRs for comparison with the input X-ray images, leading to significant computational cost. Chapter 3 describes the DRR generation process in more detail, and then introduces a software based method for accelerated DRR generation based on a precomputed representation of the CT data. Implementation details are presented, as well as a procedure for generating DRRs which correspond to a specific patient poses.

Chapter 4 describes hardware accelerated texture mapping operations, and proves that DRR generation can be accelerated by simply accumulating the results of 2D texture operations. An algorithm is presented for using graphics hardware to generate DRRs.

The algorithm presented in chapter 4 depends on a graphics feature called *hardware accelerated accumulation buffering*, which is very rarely implemented in PC graphics hardware. Chapter 5 presents a way of emulating hardware accelerated accumulation buffer operations using the texture hardware of a very consumer level graphics card. The advantages and limitations of this approach are discussed.

Chapter 6 describes the imaging hardware used in the experiments, and presents parameterized models for the imaging process. Calibration routines are presented recover these parameters for two specific types of imager.

Chapter 7 is the first experimental chapter. An existing image-guided radiosurgery system is described, and an experiment is presented in which our registration algorithm was tested against an independent measurement of patient pose. Registration errors are computed in several ways, and the system accuracy is discussed.

Chapter 8 discusses the use of X-ray/CT registration in measurement of acetabular cup orientation following total hip replacement surgery. An algorithm is developed in which the pelvis pose is estimated using X-ray/CT registration, and then acetabular cup position is recovered with respect to the pelvis using a contour-based registration. An experiment is presented, and results are compared with an independent measurement of cup orientation.

Chapter 9 summarizes the contributions of this thesis, and discusses future research directions.

# Chapter 2

# Iterative Registration

We estimate patient pose by iteratively comparing synthetic images, known as Digitally Reconstructed Radiographs (DRRs), with actual X-ray images of the patient. With each comparison, the estimate of patient pose is updated and a new set of DRRs is generated. This cycle repeats until the real and synthetic images are maximally similar, or until some convergence criterion is satisfied. This approach is illustrated in figure 2.1, and is described by the following steps:

1. Input images are acquired from one or more X-ray imagers and preprocessed, if necessary, to remove geometric and intensity distortions. We associate an index $j$ with each imager, and represent the corresponding un-distorted image as a 2D array of floating point numbers, $U_j$.

2. An initial pose estimate, $\gamma$, is generated based on user input, knowledge of the application, or a pre-registration procedure. The initial estimates used in our experiments are generally accurate to within 1.5 cm translation and $10^o$ rotation around the center of the CT volume.

3. A set of one or more DRRs is generated based on the pose estimate, $\gamma$. Each DRR corresponds to one of the input images. We represent the DRR corresponding to $U_j$ as a 2D array of floating point numbers, $U'_j$.

4. Each DRR is compared with the corresponding input image.

5. If significant differences exist, and convergence criteria are not satisfied, the pose estimate $\gamma$ is modified and the process continues with step 3, above.

Methods for efficiently generating DRRs are presented in chapters 3 and 4, and we defer discussion of the image preprocessing in step 1 until chapter 6. Step 2 calls for a pose estimate, $\gamma$, which we represent using a vector of pose parameters. This chapter presents two possible pose parameterizations in section 2.1. Section 2.2 corresponds to step 4, above, and introduces ways of comparing

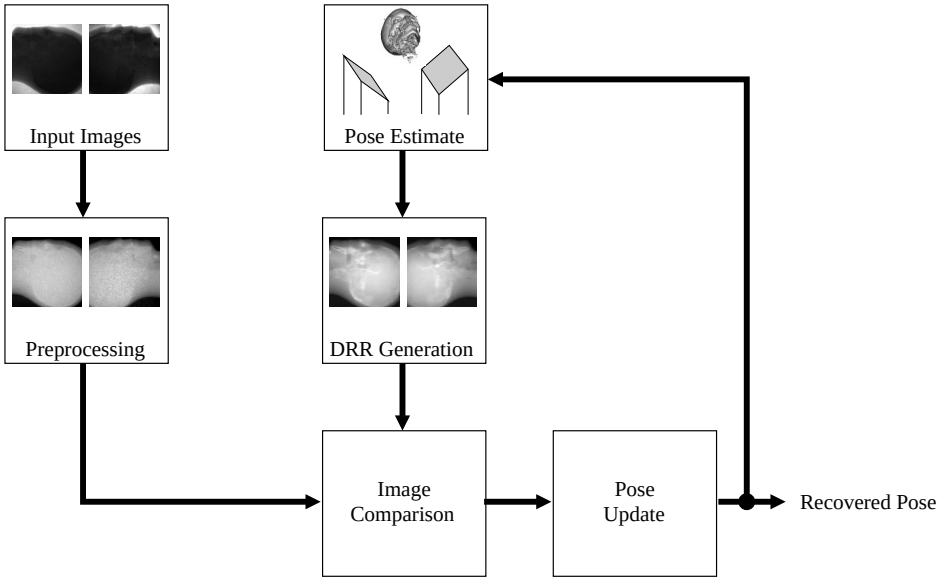



Figure 2.1: Registration is established by iteratively comparing DRRs with the input images. After each set of comparisons, the patient pose estimate is updated.

DRRs with input images. Finally, section 2.3 introduces nonlinear optimization routines which update the pose estimate, $\gamma$, in such a way that the space of pose parameters is searched efficiently.

## 2.1 Parameterization of Patient Pose

This section introduces the pose parameter vector $\gamma$, which describes the position and orientation of the patient's anatomy. In other words, the elements of $\gamma$ specify a coordinate transformation which maps coordinates from a stationary world coordinate system into a coordinate system associated with the patient. Since our 3D representation of the patient is the preoperative CT volume, we generally define the the patient coordinate system to be coincident with the coordinate system of the CT volume. It is convenient to write this coordinate transformation as a 4x4 transformation matrix, ${}^{\mathrm{ct}}T_{\mathrm{w}}(\gamma)$. This chapter presents two parameterizations of ${}^{\mathrm{ct}}T_{\mathrm{w}}$ and describes how each parameterization defines the elements of the matrix. Please refer to appendix A for a brief introduction to 4x4 coordinate transformation matrices.

### 2.1.1 Euler Angles

One convenient representation of the rigid body transformation between the CT coordinate system and the world coordinate system is the six parameter vector $[t_{\mathrm{x}},\, t_{\mathrm{y}},\, t_{\mathrm{z}},\, \theta_{\mathrm{x}},\, \theta_{\mathrm{y}},\, \theta_{\mathrm{z}}]^T$, where $t_{\mathrm{x}}$, $t_{\mathrm{y}}$, and $t_{\mathrm{z}}$ are orthogonal translations and $\theta_{\mathrm{x}}$, $\theta_{\mathrm{y}}$, and $\theta_{\mathrm{z}}$ represent consecutive rotations around each of

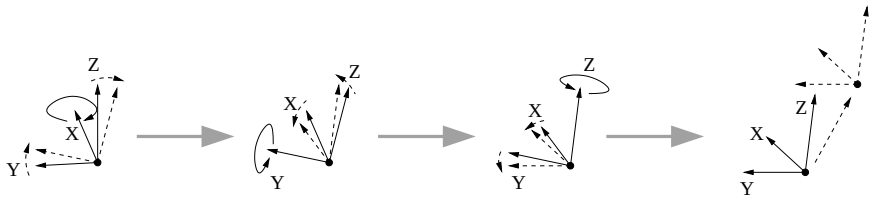



Figure 2.2: The 6-element parameterization of patient pose comprises three consecutive rotations around the three coordinate axes, followed by a 3D translation.

the three coordinate axes. Figure 2.2 illustrates the application of these rotations and translations. This parameterization is minimal in the sense that the rigid body transformation, which has six degrees of freedom, is represented using only six parameters. The relationship between this 6 parameter representation and the matrix ${}^{\mathrm{ct}}T_{\mathrm{w}}$ can be seen by writing the translation and each of the rotations as a matrix, and then composing these matrices. The matrix representation, $T$, of the translation $[t_{\mathrm{x}},\, t_{\mathrm{y}},\, t_{\mathrm{z}}]$ is

$$T(t_{\mathrm{x}}, t_{\mathrm{y}}, t_{\mathrm{z}}) = \begin{bmatrix} 1 & 0 & 0 & t_{\mathrm{x}} \\ 0 & 1 & 0 & t_{\mathrm{y}} \\ 0 & 0 & 1 & t_{\mathrm{z}} \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2.1}$$

Similarly, the matrices $R_{\mathrm{x}}$, $R_{\mathrm{y}}$, and $R_{\mathrm{z}}$, which represent rotations around the X, Y, and Z axes, can be written

$$R_{\mathrm{x}}(\theta_{\mathrm{x}}) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta_{\mathrm{x}}) & -\sin(\theta_{\mathrm{x}}) & 0 \\ 0 & \sin(\theta_{\mathrm{x}}) & \cos(\theta_{\mathrm{x}}) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

$$R_{\mathrm{y}}(\theta_{\mathrm{y}}) = \begin{bmatrix} \cos(\theta_{\mathrm{y}}) & 0 & \sin(\theta_{\mathrm{y}}) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\theta_{\mathrm{y}}) & 0 & \cos(\theta_{\mathrm{y}}) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

$$R_{\mathrm{z}}(\theta_{\mathrm{z}}) = \begin{bmatrix} \cos(\theta_{\mathrm{z}}) & -\sin(\theta_{\mathrm{z}}) & 0 & 0 \\ \sin(\theta_{\mathrm{z}}) & \cos(\theta_{\mathrm{z}}) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.4}$$

Composing these four matrices to find ${}^{\mathrm{ct}}T_{\mathrm{w}}$ gives

$$ {}^{\mathrm{ct}}T_{\mathrm{w}} = T(t_{\mathrm{x}}, t_{\mathrm{y}}, t_{\mathrm{z}}) * R_{\mathrm{z}}(\theta_{\mathrm{z}}) * R_{\mathrm{y}}(\theta_{\mathrm{y}}) * R_{\mathrm{x}}(\theta_{\mathrm{x}}) \tag{2.5} $$

$$ = \begin{bmatrix} c_{\mathrm{y}}c_{\mathrm{z}} & (s_{\mathrm{x}}s_{\mathrm{y}}c_{\mathrm{z}} - c_{\mathrm{x}}s_{\mathrm{z}}) & (c_{\mathrm{x}}s_{\mathrm{y}}c_{\mathrm{z}} + s_{\mathrm{x}}s_{\mathrm{z}}) & t_{\mathrm{x}} \\ c_{\mathrm{y}}s_{\mathrm{z}} & (s_{\mathrm{x}}s_{\mathrm{y}}s_{\mathrm{z}} + c_{\mathrm{x}}c_{\mathrm{z}}) & (c_{\mathrm{x}}s_{\mathrm{y}}s_{\mathrm{z}} - s_{\mathrm{x}}c_{\mathrm{z}}) & t_{\mathrm{y}} \\ -s_{\mathrm{y}} & s_{\mathrm{x}}c_{\mathrm{y}} & c_{\mathrm{x}}c_{\mathrm{y}} & t_{\mathrm{z}} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.6} $$

$s_{\mathrm{x}} = \sin(\theta_{\mathrm{x}})$, $c_{\mathrm{x}} = \cos(\theta_{\mathrm{x}})$, $s_{\mathrm{y}} = \sin(\theta_{\mathrm{y}})$, $c_{\mathrm{y}} = \cos(\theta_{\mathrm{y}})$, $s_{\mathrm{z}} = \sin(\theta_{\mathrm{z}})$, $c_{\mathrm{z}} = \cos(\theta_{\mathrm{z}})$.

A more detailed description of projective geometry and homogeneous coordinates is presented in [14].

Referring to equations 2.6 and A.7, we see that a 3D point $[{}^{\mathrm{w}}x,\ {}^{\mathrm{w}}y,\ {}^{\mathrm{w}}z]^T$ in world coordinates corresponds to the CT coordinate $[{}^{\mathrm{ct}}x,\ {}^{\mathrm{ct}}y,\ {}^{\mathrm{ct}}z]^T$ as follows:

$$ \begin{bmatrix} {}^{\mathrm{ct}}x \\ {}^{\mathrm{ct}}y \\ {}^{\mathrm{ct}}z \end{bmatrix} = \begin{bmatrix} c_{\mathrm{y}}c_{\mathrm{z}}({}^{\mathrm{w}}x) + (s_{\mathrm{x}}s_{\mathrm{y}}c_{\mathrm{z}} - c_{\mathrm{x}}s_{\mathrm{z}})({}^{\mathrm{w}}y) + (c_{\mathrm{x}}s_{\mathrm{y}}c_{\mathrm{z}} + s_{\mathrm{x}}s_{\mathrm{z}})({}^{\mathrm{w}}z) + t_{\mathrm{x}} \\ c_{\mathrm{y}}s_{\mathrm{z}}({}^{\mathrm{w}}x) + (s_{\mathrm{x}}s_{\mathrm{y}}s_{\mathrm{z}} + c_{\mathrm{x}}c_{\mathrm{z}})({}^{\mathrm{w}}y) + (c_{\mathrm{x}}s_{\mathrm{y}}s_{\mathrm{z}} - s_{\mathrm{x}}c_{\mathrm{z}})({}^{\mathrm{w}}z) + t_{\mathrm{y}} \\ -s_{\mathrm{y}}({}^{\mathrm{w}}x) + s_{\mathrm{x}}c_{\mathrm{y}}({}^{\mathrm{w}}y) + c_{\mathrm{x}}c_{\mathrm{y}}({}^{\mathrm{w}}z) + t_{\mathrm{z}} \end{bmatrix} \tag{2.7} $$

The disadvantage of this parameterization is that it suffers from degeneracies in certain areas of the parameter space. To see this, consider what happens when $\theta_{\mathrm{x}} = \pi/2$ radians, and $\theta_{\mathrm{y}} = -\pi/2$ radians. In this case, the rotations $\theta_{\mathrm{x}}$ and $\theta_{\mathrm{z}}$ are about the same physical axis. The degeneracy can be seen by computing the derivatives of equation 2.7 with respect to the rotation parameters $\theta_{\mathrm{x}}$, $\theta_{\mathrm{y}}$, and $\theta_{\mathrm{z}}$.

$$ \frac{\partial}{\partial\theta_{\mathrm{x}}} \begin{bmatrix} {}^{\mathrm{ct}}x \\ {}^{\mathrm{ct}}y \\ {}^{\mathrm{ct}}z \end{bmatrix} = \begin{bmatrix} (c_{\mathrm{x}}s_{\mathrm{y}}c_{\mathrm{z}} + s_{\mathrm{x}}s_{\mathrm{z}})({}^{\mathrm{w}}y) + (-s_{\mathrm{x}}s_{\mathrm{y}}c_{\mathrm{z}} + c_{\mathrm{x}}s_{\mathrm{z}})({}^{\mathrm{w}}z) \\ (c_{\mathrm{x}}s_{\mathrm{y}}s_{\mathrm{z}} - s_{\mathrm{x}}c_{\mathrm{z}})({}^{\mathrm{w}}y) + (-s_{\mathrm{x}}s_{\mathrm{y}}s_{\mathrm{z}} - c_{\mathrm{x}}c_{\mathrm{z}})({}^{\mathrm{w}}z) \\ c_{\mathrm{x}}c_{\mathrm{y}}({}^{\mathrm{w}}y) - s_{\mathrm{x}}c_{\mathrm{y}}({}^{\mathrm{w}}z) \end{bmatrix} \tag{2.8} $$

$$ \frac{\partial}{\partial\theta_{\mathrm{y}}} \begin{bmatrix} {}^{\mathrm{ct}}x \\ {}^{\mathrm{ct}}y \\ {}^{\mathrm{ct}}z \end{bmatrix} = \begin{bmatrix} -s_{\mathrm{y}}c_{\mathrm{z}}({}^{\mathrm{w}}x) + (s_{\mathrm{x}}c_{\mathrm{y}}c_{\mathrm{z}})({}^{\mathrm{w}}y) + (c_{\mathrm{x}}c_{\mathrm{y}}c_{\mathrm{z}})({}^{\mathrm{w}}z) \\ -s_{\mathrm{y}}s_{\mathrm{z}}({}^{\mathrm{w}}x) + (s_{\mathrm{x}}c_{\mathrm{y}}s_{\mathrm{z}})({}^{\mathrm{w}}y) + (c_{\mathrm{x}}c_{\mathrm{y}}s_{\mathrm{z}})({}^{\mathrm{w}}z) \\ -c_{\mathrm{y}}({}^{\mathrm{w}}x) - s_{\mathrm{x}}s_{\mathrm{y}}({}^{\mathrm{w}}y) - c_{\mathrm{x}}s_{\mathrm{y}}({}^{\mathrm{w}}z) \end{bmatrix} \tag{2.9} $$

$$ \frac{\partial}{\partial\theta_{\mathrm{z}}} \begin{bmatrix} {}^{\mathrm{ct}}x \\ {}^{\mathrm{ct}}y \\ {}^{\mathrm{ct}}z \end{bmatrix} = \begin{bmatrix} -c_{\mathrm{y}}s_{\mathrm{z}}({}^{\mathrm{w}}x) - (s_{\mathrm{x}}s_{\mathrm{y}}s_{\mathrm{z}} + c_{\mathrm{x}}c_{\mathrm{z}})({}^{\mathrm{w}}y) - (c_{\mathrm{x}}s_{\mathrm{y}}s_{\mathrm{z}} - s_{\mathrm{x}}c_{\mathrm{z}})({}^{\mathrm{w}}z) \\ c_{\mathrm{y}}c_{\mathrm{z}}({}^{\mathrm{w}}x) + (s_{\mathrm{x}}s_{\mathrm{y}}c_{\mathrm{z}} - c_{\mathrm{x}}s_{\mathrm{z}})({}^{\mathrm{w}}y) + (c_{\mathrm{x}}s_{\mathrm{y}}c_{\mathrm{z}} + s_{\mathrm{x}}s_{\mathrm{z}})({}^{\mathrm{w}}z) \\ 0 \end{bmatrix}. \tag{2.10} $$

When $\theta_{\mathrm{x}} = \pi/2$ radians, and $\theta_{\mathrm{y}} = -\pi/2$ radians, the derivatives in equations 2.8 and 2.10

become identical, and the three rotation parameters no longer represent independent rotations.

$$\frac{\partial}{\partial\theta_{\mathrm{x}}}\left[\begin{array}{c}{}^{\mathrm{ct}}x\\ {}^{\mathrm{ct}}y\\ {}^{\mathrm{ct}}z\end{array}\right]\Bigg|_{\theta_{\mathrm{x}}=\frac{\pi}{2},\theta_{\mathrm{y}}=-\frac{\pi}{2}} = \left[\begin{array}{c}s_{\mathrm{z}}({}^{\mathrm{w}}y)+c_{\mathrm{z}}({}^{\mathrm{w}}z)\\ -c_{\mathrm{z}}({}^{\mathrm{w}}y)+s_{\mathrm{z}}({}^{\mathrm{w}}z)\\ 0\end{array}\right] \tag{2.11}$$

$$\frac{\partial}{\partial\theta_{\mathrm{z}}}\left[\begin{array}{c}{}^{\mathrm{ct}}x\\ {}^{\mathrm{ct}}y\\ {}^{\mathrm{ct}}z\end{array}\right]\Bigg|_{\theta_{\mathrm{x}}=\frac{\pi}{2},\theta_{\mathrm{y}}=-\frac{\pi}{2}} = \left[\begin{array}{c}s_{\mathrm{z}}({}^{\mathrm{w}}y)+(c_{\mathrm{z}})({}^{\mathrm{w}}z)\\ -c_{\mathrm{z}}({}^{\mathrm{w}}y)+s_{\mathrm{z}}({}^{\mathrm{w}}z)\\ 0\end{array}\right]. \tag{2.12}$$

This type of degeneracy is known as *gimbal lock* because it mimics a physical limitation is the mechanical rotational device known as a gimbal. Gimbal lock results in the loss of one rotational degree of freedom at the degeneracy, and causes the parameterization to be unstable in the neighborhood of the degeneracy.

The degeneracies in the $[t_{\mathrm{x}},\, t_{\mathrm{y}},\, t_{\mathrm{z}},\, \theta_{\mathrm{x}},\, \theta_{\mathrm{y}},\, \theta_{\mathrm{z}}]^T$ parameter space are of no consequence, provided the actual pose of the patient is known not to lie in the neighborhood of a degeneracy. This is guaranteed whenever all three rotation angles are small, less than $\pi/4$, say. This is not always the case, so we present an alternative parameterization which does not suffer from degeneracies.

### 2.1.2 Unit Quaternion

For those situations in which the the parameter space must be free of degeneracies, it is often convenient to use another representation of rotation known as a unit quaternion. A unit quaternion can be thought of as a four-element vector having unit magnitude. Quaternions, and how they can be used to represent rotation, are discussed in [33].

In this parameterization, we simply replace the three consecutive rotations of the previous parameterization with the four elements of a unit quaternion, and write the resulting seven-element parameter vector $[t_{\mathrm{x}},\, t_{\mathrm{y}},\, t_{\mathrm{z}},\, s,\, i,\, j,\, k]^T$. The translational component of ${}^{\mathrm{ct}}T_{\mathrm{w}}$ is again represented by the three orthogonal translations $t_{\mathrm{x}}$, $t_{\mathrm{y}}$, and $t_{\mathrm{z}}$.

We adopt the convention that a unit quaternion $[s,\, i,\, j,\, k]^T$ has a corresponding rotation matrix, $R_q$:

$$R_q(s,i,j,k) = \left[\begin{array}{cccc}(1-2j^2-2k^2) & 2(ij-sk) & 2(ik+sj) & 0\\ 2(ij+sk) & (1-2i^2-2k^2) & 2(jk-si) & 0\\ 2(ik-sj) & 2(jk+si) & (1-2i^2-2j^2) & 0\\ 0 & 0 & 0 & 1\end{array}\right]. \tag{2.13}$$

This equation is valid only when the quaternion has unit magnitude. In practice, it is often inconve-

nient to enforce this constraint during optimization, so we include an explicit normalization in our parameterization of $R_q$.

$$R'_q(s,i,j,k) = \begin{bmatrix} (1-2j'^2-2k'^2) & 2(i'j'-s'k') & 2(i'k'+s'j') & 0 \\ 2(i'j'+s'k') & (1-2i'^2-2k'^2) & 2(j'k'-s'i') & 0 \\ 2(i'k'-s'j') & 2(j'k'+s'i') & (1-2i'^2-2j'^2) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{2.14}$$

where

$$s' = \frac{s}{\sqrt{s^2+i^2+j^2+k^2}}, \qquad i' = \frac{i}{\sqrt{s^2+i^2+j^2+k^2}}, \tag{2.15}$$

$$j' = \frac{j}{\sqrt{s^2+i^2+j^2+k^2}}, \qquad k' = \frac{k}{\sqrt{s^2+i^2+j^2+k^2}}. \tag{2.16}$$

Equivalently,

$$R'_q(s,i,j,k) = \begin{bmatrix} \frac{s^2+i^2-j^2-k^2}{s^2+i^2+j^2+k^2} & \frac{2(ij-sk)}{s^2+i^2+j^2+k^2} & \frac{2(ik+sj)}{s^2+i^2+j^2+k^2} & 0 \\ \frac{2(ij+sk)}{s^2+i^2+j^2+k^2} & \frac{s^2-i^2+j^2-k^2}{s^2+i^2+j^2+k^2} & \frac{2(jk-si)}{s^2+i^2+j^2+k^2} & 0 \\ \frac{2(ik-sj)}{s^2+i^2+j^2+k^2} & \frac{2(jk+si)}{s^2+i^2+j^2+k^2} & \frac{s^2-i^2-2j^2+k^2}{s^2+i^2+j^2+k^2} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2.17}$$

Using this rotation matrix, we can write

$${}^{\mathrm{ct}}T_{\mathrm{w}}(\gamma) = T(t_{\mathrm{x}},t_{\mathrm{y}},t_{\mathrm{z}}) * R'_q(s,i,j,k) = \begin{bmatrix} \frac{s^2+i^2-j^2-k^2}{s^2+i^2+j^2+k^2} & \frac{2(ij-sk)}{s^2+i^2+j^2+k^2} & \frac{2(ik+sj)}{s^2+i^2+j^2+k^2} & t_{\mathrm{x}} \\ \frac{2(ij+sk)}{s^2+i^2+j^2+k^2} & \frac{s^2-i^2+j^2-k^2}{s^2+i^2+j^2+k^2} & \frac{2(jk-si)}{s^2+i^2+j^2+k^2} & t_{\mathrm{y}} \\ \frac{2(ik-sj)}{s^2+i^2+j^2+k^2} & \frac{2(jk+si)}{s^2+i^2+j^2+k^2} & \frac{s^2-i^2-2j^2+k^2}{s^2+i^2+j^2+k^2} & t_{\mathrm{z}} \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2.18}$$

Transforming the world coordinate $[{}^{\mathrm{w}}x,\ {}^{\mathrm{w}}y,\ {}^{\mathrm{w}}z]^T$ to a point $[{}^{\mathrm{ct}}x,\ {}^{\mathrm{ct}}y,\ {}^{\mathrm{ct}}z]^T$ in the CT coordinate system using this matrix gives

$$\begin{bmatrix} {}^{\mathrm{ct}}x \\ {}^{\mathrm{ct}}y \\ {}^{\mathrm{ct}}z \end{bmatrix} = \begin{bmatrix} \frac{(s^2+i^2-j^2-k^2)({}^{\mathrm{w}}x)+2(ij-sk)({}^{\mathrm{w}}y)+2(ik+sj)({}^{\mathrm{w}}z)}{s^2+i^2+j^2+k^2} + t_{\mathrm{x}} \\ \frac{2(ij+sk)({}^{\mathrm{w}}x)+(s^2-i^2+j^2-k^2)({}^{\mathrm{w}}y)+2(jk-si)({}^{\mathrm{w}}z)}{s^2+i^2+j^2+k^2} + t_{\mathrm{y}} \\ \frac{2(ik-sj)({}^{\mathrm{w}}x)+2(jk+si)({}^{\mathrm{w}}y)+(s^2-i^2-j^2+k^2)({}^{\mathrm{w}}z)}{s^2+i^2+j^2+k^2} + t_{\mathrm{z}} \end{bmatrix}. \tag{2.19}$$

For a discussion of how unit quaternions relate to matrix transformations, and the computational aspects of each, please refer to [16].

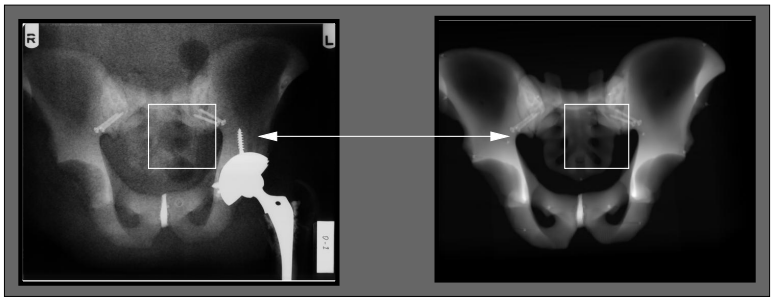


Figure 2.3: Often, it is useful to compute the normalized correlation over a specific region in a pair of images.

## 2.2 Image Comparison Functions

As discussed in the introduction of this chapter, a key part of our registration algorithm is the comparison between X-ray images and DRRs. The image comparison metric must reach an extremum when the pose parameter estimate matches the actual pose of the patient. In other words, the image comparison metric must measure how well the rendered images match the input X-ray images. The image comparison metric must not be confused by image noise, unanticipated variations in image brightness, and clutter from unmodeled anatomy, tools, or other structures in the field of view. In our work we use two measures of image similarity: the *sum of local normalized correlation* (SLNC) metric is described in section 2.2.2; and the *variance-weighted sum of local normalized correlation* (VLNC) metric is described in section 2.2.3. Both of these similarity measures are extensions of the *normalized correlation* similarity measure, which is described in section 2.2.1.

### 2.2.1 Normalized Correlation

*Normalized correlation* is traditionally used in computer vision for applications such as template comparison and stereo matching. In this context, the normalized correlation between two images is often called the *correlation coefficient* [14] [11]. A principle advantage of normalized correlation is that it is invariant to linear changes in image intensity. That is, the normalized correlation between two images is unchanged even if the pixel intensities in one or both of the images are multiplied by a positive constant, or are increased or decreased by a constant.

Often we are concerned not with the normalized correlation of two complete images, but rather with the normalized correlation between a smaller image and a specific region of a larger image, or the normalized correlation between two image regions as shown in figure 2.3.

The normalized correlation of two image regions can be computed by first normalizing each image region to have zero mean and unit variance, then multiplying each pixel in one image region

by the corresponding pixel in the other image region, and summing the products. To express this more precisely, we represent the two images using the 2D functions $I_0(\boldsymbol{p})$ and $I_1(\boldsymbol{p})$, where the parameter $\boldsymbol{p} = [r, s]^T$ is a point in 2D image coordinates. We describe the image region by defining a set, $P$, of 2D points such that a point $[r, s]^T$ is included in $P$ if and only if it corresponds to a pixel location within the image region. Normalizing each image region is straightforward:

$$\overline{I_i}(\boldsymbol{p}) = \frac{I_i(\boldsymbol{p}) - \frac{1}{|P|}\sum_{\boldsymbol{q}\in P} I_i(\boldsymbol{q})}{\sqrt{\frac{1}{|P|}\sum_{\boldsymbol{q}\in P} I_i(\boldsymbol{q})^2 - \frac{1}{|P|^2}\left(\sum_{\boldsymbol{q}\in P} I_i(\boldsymbol{q})\right)^2}}, \tag{2.20}$$

where $I_i(\boldsymbol{p})$ is the original image value at pixel location $\boldsymbol{p}$, $|P|$ is the number of pixels in image region $P$, and $\overline{I_i}(\boldsymbol{p})$ is the normalized value at pixel location $\boldsymbol{p}$. The quantity $\frac{1}{|P|}\sum_{\boldsymbol{q}\in P} I_i(\boldsymbol{q})$ is the mean pixel value within region $P$, and the quantity $\left(\frac{1}{|P|}\sum_{\boldsymbol{q}\in P} I_i(\boldsymbol{q})^2 - \frac{1}{|P|^2}\left(\sum_{\boldsymbol{q}\in P} I_i(\boldsymbol{q})\right)^2\right)$ is the variance of the pixel values in region $P$. The normalized correlation coefficient between the region $P$ in the two images is

$$\mathrm{NC}(I_0, I_1, P) = \sum_{\boldsymbol{p}\in P} \overline{I_0}(\boldsymbol{p})\overline{I_1}(\boldsymbol{p}). \tag{2.21}$$

Combining equations 2.20 and 2.21, the normalized correlation coefficient can be written directly

$$\mathrm{NC}(I_0, I_1, P) = \frac{\sum_{\boldsymbol{p}\in P} I_0(\boldsymbol{p})I_1(\boldsymbol{p}) - \frac{1}{|P|}\sum_{\boldsymbol{p}\in P} I_0(\boldsymbol{p})\sum_{\boldsymbol{p}\in P} I_1(\boldsymbol{p})}{\sqrt{\left(\sum_{\boldsymbol{p}\in P} I_0(\boldsymbol{p})^2 - \frac{1}{|P|}\left(\sum_{\boldsymbol{p}\in P} I_0(\boldsymbol{p})\right)^2\right)\left(\sum_{\boldsymbol{p}\in P} I_1(\boldsymbol{p})^2 - \frac{1}{|P|}\left(\sum_{\boldsymbol{p}\in P} I_1(\boldsymbol{p})\right)^2\right)}}. \tag{2.22}$$

### 2.2.2 Sum of Local Normalized Correlation

Although normalized correlation is invariant to linear changes in image intensity, our experience is that spatially varying intensity distortions, such as those introduced by image vignetting and non-uniformity in the imager response, can significantly bias the result. To overcome this problem, we present a modified image comparison method.

We assume that the intensity distortions can be described using two bias functions which vary slowly over the image, and write

$$I_i(\boldsymbol{p}) = W_i(\boldsymbol{p})\widetilde{I}_i(\boldsymbol{p}) + B_i(\boldsymbol{p}), \tag{2.23}$$

where $W_i(\boldsymbol{p})$ is a spatially varying multiplicative bias, $B_i(\boldsymbol{p})$ is a spatially varying additive bias, and $\widetilde{I}_i(\boldsymbol{p})$ is the underlying unbiased signal. Approximating $W_i$ and $B_i$ using a Taylor series expansion

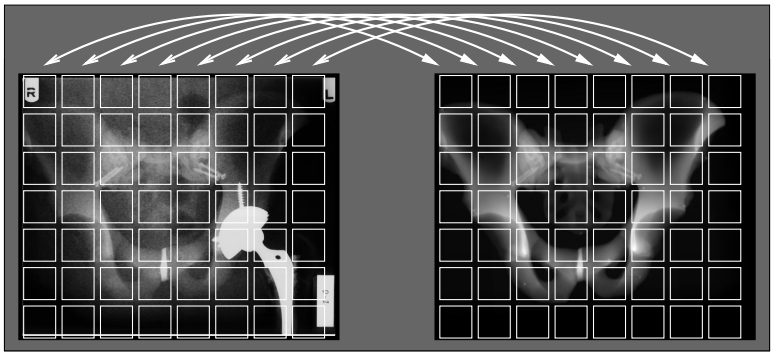


Figure 2.4: The Sum of Local Normalized Correlation image comparison metric is computed by adding the normalized correlation scores from many small image regions. The regions can be non-overlapping, as shown here, or overlapping. In the limit, a region can be centered on each image pixel.

[40], we write $I_i$ in the neighborhood of $\boldsymbol{p}_0$

$$\begin{aligned} I_i(\boldsymbol{p}_0 + \Delta\boldsymbol{p}) &= (W_i(\boldsymbol{p}_0) + (\nabla_{\boldsymbol{p}} W_i(\boldsymbol{p}_0))\,\Delta\boldsymbol{p} + \ldots)\,\widetilde{I}_i(\boldsymbol{p}) && (2.24) \\ &\quad + (B_i(\boldsymbol{p}_0) + (\nabla_{\boldsymbol{p}} B_i(\boldsymbol{p}_0))\,\Delta\boldsymbol{p} + \ldots)\,, && (2.25) \end{aligned}$$

where the ellipses represent higher order terms in $\Delta\boldsymbol{p}$. If the spatial frequency of the bias functions is low, and if we restrict our attention to a small neighborhood of $\boldsymbol{p}_0$, we can neglect all but the first term of each Taylor series

$$I_i(\boldsymbol{p}_0 + \Delta\boldsymbol{p}) \approx W_i(\boldsymbol{p}_0)\widetilde{I}_i(\boldsymbol{p}) + B_i(\boldsymbol{p}_0). \tag{2.26}$$

Equation 2.26 states that under this approximation the effects of non-uniform image intensity look locally like a simple linear scaling. This suggests that local application of normalized correlation is appropriate. Accordingly, we define an image comparison metric which evaluates the normalized correlation in a series of small neighborhoods which collectively span the images as shown in figure 2.4.

$$\mathrm{SLNC}(I_0, I_1) = \frac{1}{|Q|} \sum_{\boldsymbol{p} \in Q} \mathrm{NC}(I_0, I_1, P(\boldsymbol{p})), \tag{2.27}$$

where $Q$ is a set of 2D pixel locations which span the region over which SLNC is to be computed, $P(\boldsymbol{p})$ is neighborhood surrounding the point $\boldsymbol{p}$, and the function $\mathrm{NC}()$ is defined in equation 2.22. We divide the sum by $|Q|$, the number of points in the set $Q$, to obtain SLNC values which range from 1 (perfectly correlated) to $-1$ (perfectly anticorrelated).

In our work we choose $P(\boldsymbol{p})$ to be a square 7 pixel by 7 pixel or 11 pixel by 11 pixel window

surrounding $\boldsymbol{p}$, and $Q$ to be the set of all pixel locations for which the corresponding $P()$ does not extend past the edge of the image. Consequently, we end up computing the normalized correlation between the two images over a dense grid of overlapping windows. This choice lends itself to a particularly efficient implementation, since all of the necessary summations can be computed using recursive filters [9]. Our current implementation computes the SLNC between a pair of 256x256 images, using an 11x11 pixel window centered at every pixel, in under 170 ms on a 933 MHz Pentium III machine. During registration, SLNC computation is somewhat faster, since summations involving only the input image can be cached between iterations, and do not need to be recomputed.

In summary, we compute the SLNC between a DRR and the corresponding input image as follows:

1. For each pixel in the DRR, we define a surrounding region of interest, for example a window of 11x11 pixels or 7x7 pixels.

2. We compute the normalized correlation between each window and the corresponding region of the input image. We do this for each pixel in the image, excepting those at which the region of interest extends past the border of the image.

3. We compute the mean of these correlation values over all pixels.

Occasionally, one or more of the regions of interest in the input image has all pixels at the same intensity. When this happens, the normalized correlation coefficient for that region is undefined, and a correlation value of 0 is arbitrarily assigned.

The situation is more complicated when part of the DRR has uniform intensity. Since the DRR changes from iteration to iteration, simply assigning a zero normalized correlation to these patches would cause the SLNC function to be evaluated over different regions of the image at each iteration, leading to discontinuities in the image comparison measure. This is avoided by generating a bias image of small magnitude i.i.d. Gaussian noise and adding this bias to each DRR.

### 2.2.3 Variance-Weighted Sum of Local Normalized Correlation

The local normalized correlation metric presented above has one significant disadvantage in our application. This disadvantage is that the normalized correlation values for all neighborhoods are weighted equally. Figure 2.5 shows an input radiograph of a pelvis phantom and a corresponding DRR. Two regions of interest are labeled in each image: region A overlaps the pelvis, while region B does not. Clearly, region A provides more information about the pose of the pelvis than region B, yet the SLNC image comparison weights these two regions equally. To overcome this disadvantage we introduce the *variance-weighted sum of local normalized correlation* (VLNC) function.

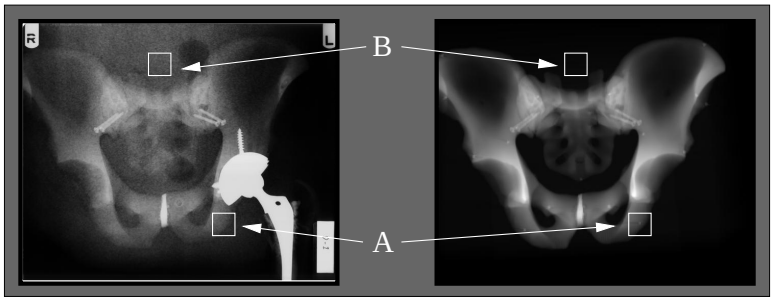



Figure 2.5: The variance-weighted sum of local normalized correlation function gives more weight to region A, which contains part of the pelvis, than to region B, which does not.

We define VLNC to be similar to SLNC, except that normalized correlation value for each neighborhood is scaled by the local variance of one of the two images. We call this image the *control image.* This scaling effectively concentrates attention in those regions of the image where the control image has high signal strength. Also, rather than simply computing the mean of these scaled normalized correlation values, we compute a weighted average. Assuming that $I_1$ is the control image, We write

$$\mathrm{VLNC}(I_0, I_1) = \frac{\sum_{\boldsymbol{p}\in Q} \mathrm{C}(I_1, I_1, P(\mathbf{p})) * \mathrm{NC}(I_0, I_1, P(\boldsymbol{p}))}{\sum_{\boldsymbol{p}\in Q} \mathrm{C}(I_1, I_1, P(\mathbf{p}))} \tag{2.28}$$

$$\mathrm{C}(I_i, I_j, P(\mathbf{p})) = \frac{1}{|P(\boldsymbol{p})|} \sum_{\boldsymbol{q}\in P(\boldsymbol{p})} I_i(\boldsymbol{q}) I_j(\boldsymbol{q}) - \frac{1}{|P(\boldsymbol{p})|^2} \sum_{\boldsymbol{q}\in P(\boldsymbol{p})} I_i(\boldsymbol{q}) \sum_{\boldsymbol{q}\in P(\boldsymbol{p})} I_j(\boldsymbol{q}), \tag{2.29}$$

where the function $\mathrm{C}(I_1, I_1, P(\mathbf{p}))$ computes the variance of the control image within the neighborhood $P(\boldsymbol{p})$, and all terms are defined as in equation 2.27, above. As before, the VLNC function value ranges from 1 (perfectly correlated) to $-1$ (perfectly anticorrelated).

As with SLNC, we choose $P(\boldsymbol{p})$ to be a square 7 pixel by 7 pixel or 11 pixel by 11 pixel window surrounding $\boldsymbol{p}$, and $Q$ to be the set of all pixel locations for which the corresponding $P()$ does not extend past the edge of the image. Computation times for the VLNC error function are nearly identical to those of the SLNC function. Our implementation requires just under 170 ms to compare two 256x256 images on a 933MHz Pentium III test machine. Because the summations are implemented using recursive filters, the computation time is nearly independent of the size of the correlation windows.

In summary, we compute the VLNC between a DRR and the corresponding input image as follows:

1. For each pixel in the synthetic image, we define a surrounding region of interest. for example a window of 11x11 pixels or 7x7 pixels.

2. We choose the DRR to be the control image, and compute its variance over each window, excepting those windows which extend past the border of the image.

3. We compute the normalized correlation between each of the windows from step 2 and the corresponding region of the input image. Note that the variance from step 2 can be used in this computation, and need not be recomputed.

4. We scale each correlation value by the associated variance, and compute the weighted average of the set.

Computation times for the VLNC error function are nearly identical to those of the SLNC function. Our implementation requires just under 170 ms to compare two 256x256 images on our 933MHz Pentium III test machine.

Occasionally, one or more of the regions of interest in the input image has all pixels at the same intensity. When this happens, the normalized correlation coefficient for that window is undefined, and a value of zero is substituted.

It is not necessary to bias the DRR with small magnitude noise as we did when computing SLNC. We can see this by rewriting the numerator of equation 2.28

$$\sum_{\boldsymbol{p}\in Q} \mathrm{C}(I_1, I_1 P(\mathbf{p})) * \mathrm{NC}(I_0, I_1, P(\boldsymbol{p})) = \sum_{\boldsymbol{p}\in Q} \left( \frac{C(I_0, I_1, P(\boldsymbol{p}))\sqrt{C(I_1, I_1, P(\boldsymbol{p}))}}{\sqrt{C(I_0, I_0, P(\boldsymbol{p}))}} \right), \quad (2.30)$$

where we have substituted for $\mathrm{NC}(I_0, I_1, P(\boldsymbol{p}))$ using equation 2.22, and then simplified using equation 2.29. Substituting equation 2.30 into equation 2.28, we see that the VLNC is well defined as long as at the DRR has at least one non-uniform neighborhood.

### 2.2.4 Performance of Image Comparison Functions

We illustrate the differences between the three image comparison metrics with an example. Figure 2.6 shows four images of a pelvis. In each image, the pelvis is in the same position and orientation with respect to the X-ray imager. Figure 2.6(a) is a very clean synthetic image showing only the pelvis. Figure 2.6(b) is a copy of the image in figure 2.6(a) to which additive and multiplicative bias has been added following equation 2.23. Figure 2.6(c) is a real image of a high density Sawbones phantom. The phantom pelvis is surrounded by simulated soft tissue as described in chapter 8. Figure 2.6(d) is a copy of the image in figure 2.6(c) to which image noise and clutter have been added. The added noise consists of independent, zero mean, identically distributed uniform noise at

each pixel, and a slowly varying additive bias. The clutter consists of other pelvis images, both AP and Lateral, which were simply added to image.

For these images, the actual position and orientation of the pelvis are known with good precision. We call this position and orientation the *target pose*. A series of 101 DRRs were generated corresponding to poses in the neighborhood of the target pose. The DRR series starts with the pelvis a shifted 0.5 cm to the patient's left of the target pose. The pelvis was shifted back toward the patient's right by 0.099 mm prior to each successive DRR, so that the final DRR was generated with the pelvis shifted 0.5 cm to the patient's right. Displayed in sequence, these DRRs look like a movie of the pelvis translating across the screen, with the DRR at the very middle of the movie corresponding to the target pose. Figure 2.7 illustrates the direction of this motion.

Each DRR was compared to each of the four images using each of the three similarity measures, and graphs were generated showing how the image similarity value changed throughout the series. Figure 2.8 shows these graphs for the normalized correlation image comparison function. Figure 2.8(a) shows the normalized correlation results between the DRR sequence to the clean synthetic image. This graph has a clear correlation peak as the pelvis moves past the target pose. Figure 2.8(b) shows a similar graph for the normalized correlation between the DRR sequence and the image in figure 2.6(b). In this graph, the correlation peak has shifted off to the side by just over 1 mm. This shift would lead to an inaccuracy in registration. Figures 2.8(c) and 2.8(d) show even more significant deviations from the target pose.

Figure 2.9 shows the results from image comparisons using the SLNC image comparison function. As before, figure 2.9(a) shows SLNC values between the DRR sequence and the clean synthetic image. This graph has a clear peak at the target pose. SLNC values between the DRR sequence and the biased image of figure of figure 2.6(b) are a significant improvement over the normalized correlation values, as the peak now occurs within 0.2 mm of the target pose. This is to be expected, since the SLNC function was designed to handle exactly this type of bias in the image. Figure 2.9(c) shows a similar result for the phantom image of figure 2.6(c). The result deteriorates somewhat for the cluttered image in figure 2.6(d), with a similarity peak almost 0.4 mm from the target pose. Again, this is to be expected, since the image clutter violates the assumptions of equation 2.26.

Fortunately, the attention focusing characteristics of the VLNC image comparison function help to filter out much of the clutter. Figure 2.10 shows the results from image comparisons using VLNC. This figure shows similarity peaks within 0.15 mm of the target pose for both the phantom image of figure 2.6(c) and the cluttered image of figure 2.6(d).

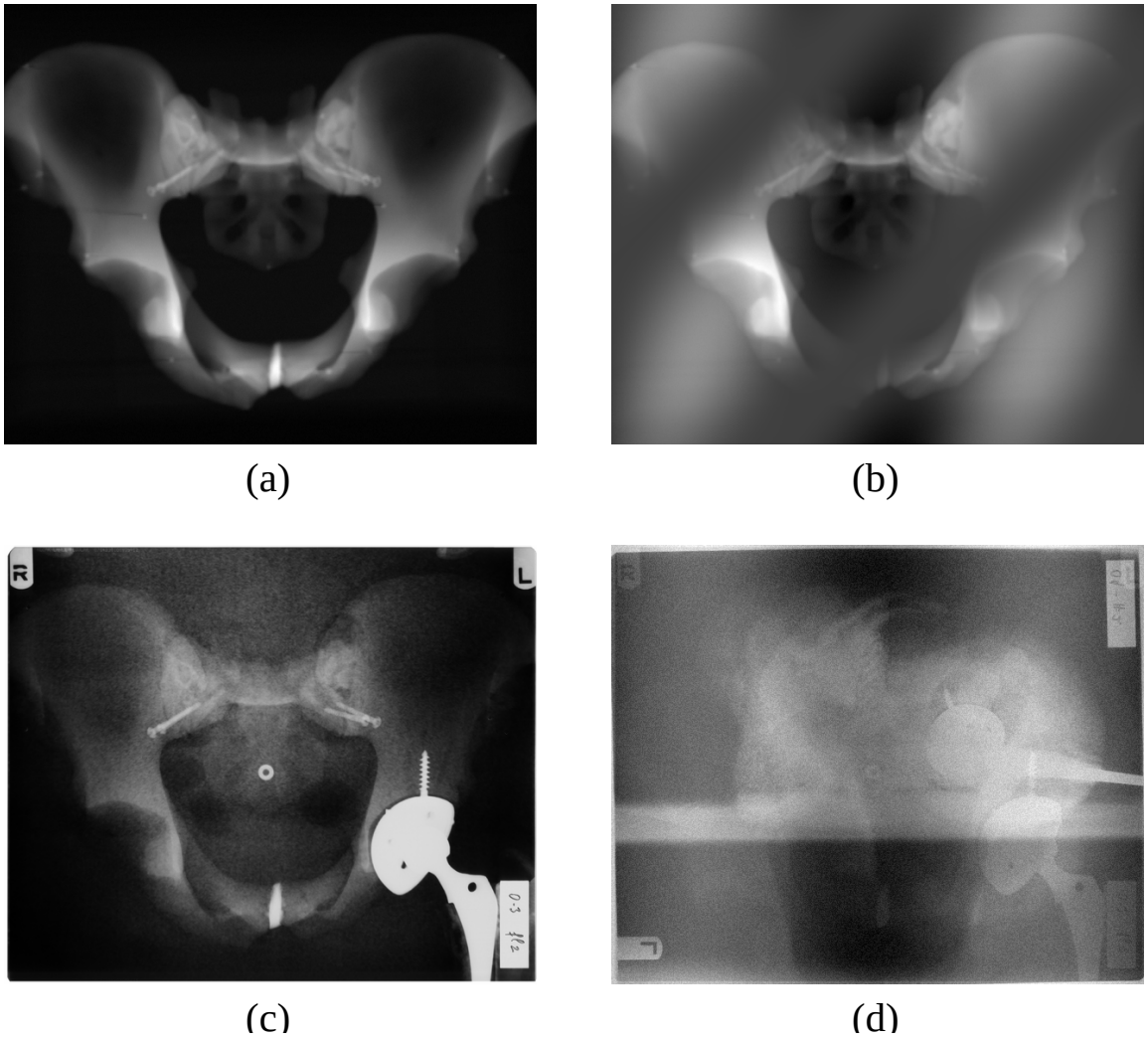



Figure 2.6: Four test images were used to illustrate the performance differences between the three image comparison metrics: image (a) is simply a DRR; image (b) is the same as image (a), except that a spatially varying bias has been applied; image (c) is a real input image from a phantom study; and image (d) is the same as image (c), except that noise, clutter, and a spacially varying bias have been added, almost completely obscuring the original view of the pelvis.

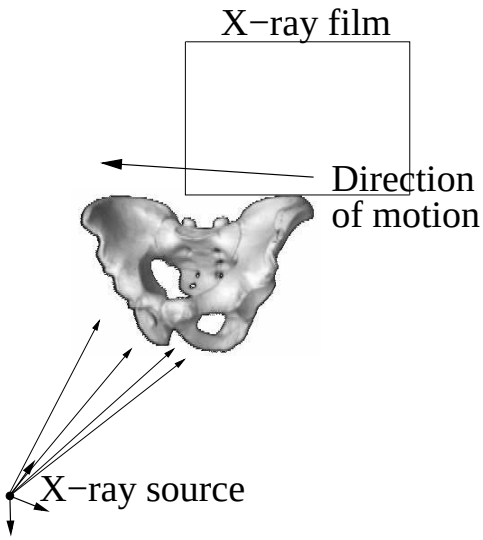



Figure 2.7: A series of DRRs were generated. Prior to each DRR, the pelvis was shifted slightly, so that, viewed in sequence, the entire series looks like a movie of the pelvis translating across the field of view.

## 2.3 Optimization

Once a parameterization of patient pose and an image comparison function have been selected, an algorithm must be chosen for finding the patient pose which maximizes image similarity. Exhaustively sampling the 6 or 7 dimensional pose space is out of the question, since the number of samples required would be prohibitively high. Accordingly we conduct the search using iterative nonlinear optimization routines. These routines work by iteratively adjusting a vector of parameters in order to minimize a scalar valued *objective function*. The objective function takes the vector of parameters as an argument, and returns a single floating point value. In our case, the vector of parameters is simply the pose parameter vector, $\gamma$, and the scalar return value is simply an indication of how well the pose parameter vector matches the input images. We define scalar valued objective functions based on the SLNC and VLNC image comparison metrics in section 2.3.1, while section 2.3.2 presents the actual optimization routines.

### 2.3.1 Objective Functions

The image comparison metrics described in section 2.2 measure the similarity between pairs of images. During registration, we may need to combine the information from more than one image pair. For example, the image-guided radiosurgery system presented in chapter 7 has two X-ray imagers, and the images from both are used simultaneously to determine patient position. Accordingly, we

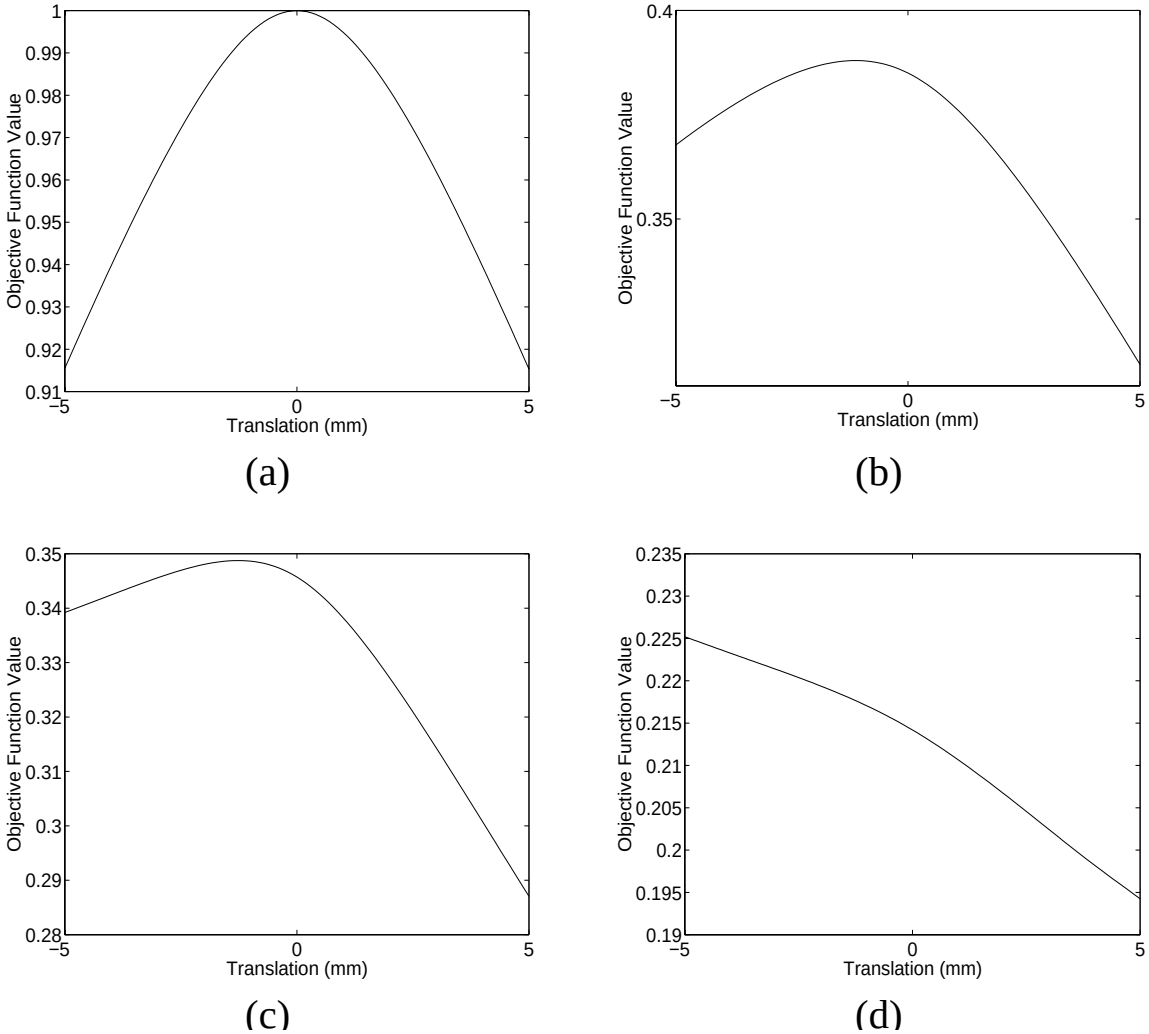



Figure 2.8: These graphs show how the normalized correlation value changes as the pelvis pose estimate is translated from left to right. The four graphs correspond to the four images in figure 2.6. The correlation peak diverges significantly from the ideal position (0 mm translation) for all except the clean synthetic image shown in figure 2.6(a).

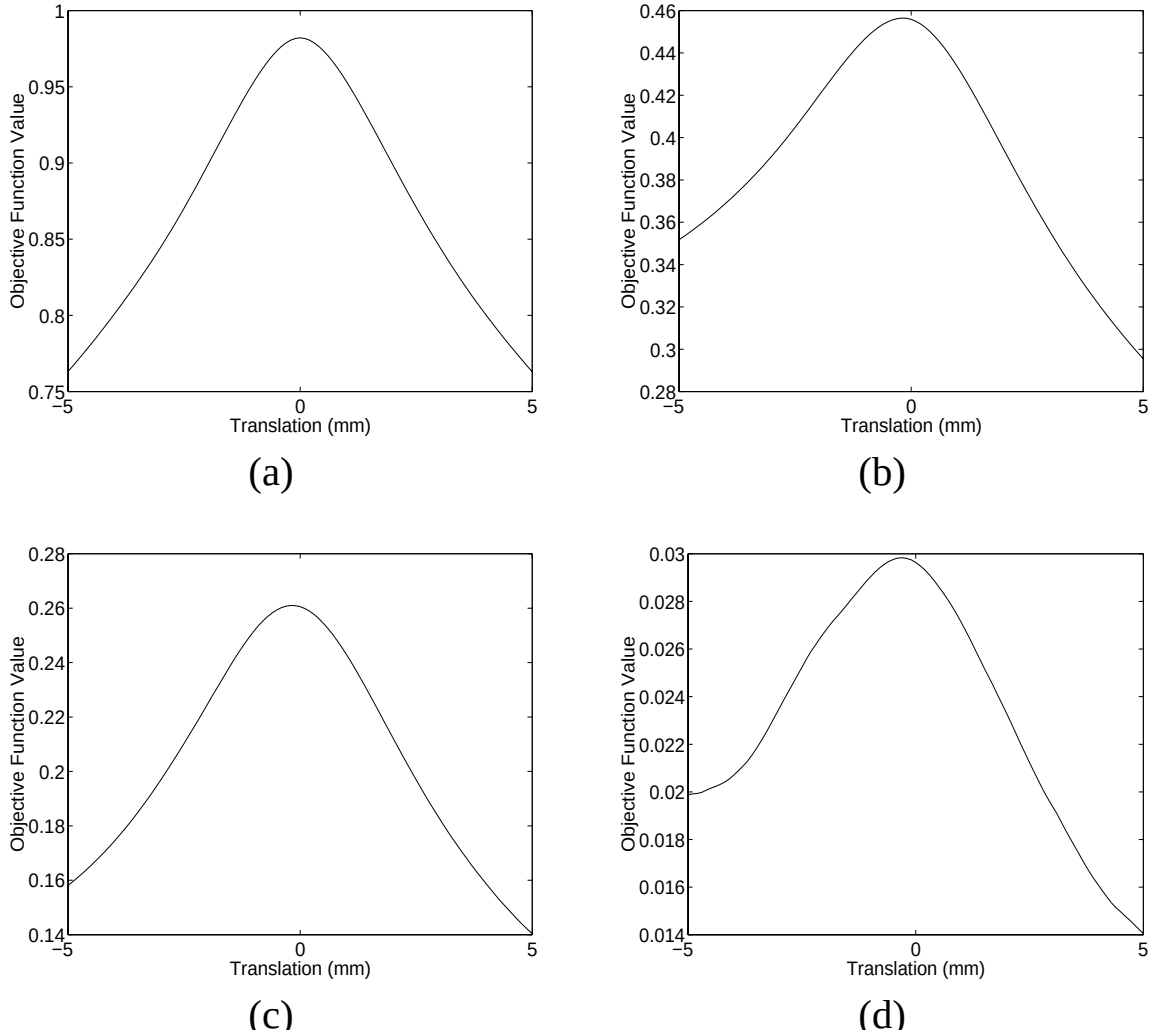



Figure 2.9: These graphs show how the sum of local normalized correlation value changes as the pelvis pose estimate is translated from left to right. The four graphs correspond to the four images in figure 2.6. The similarity peak diverges significantly from the ideal position only for the cluttered image shown in figure 2.6(d).

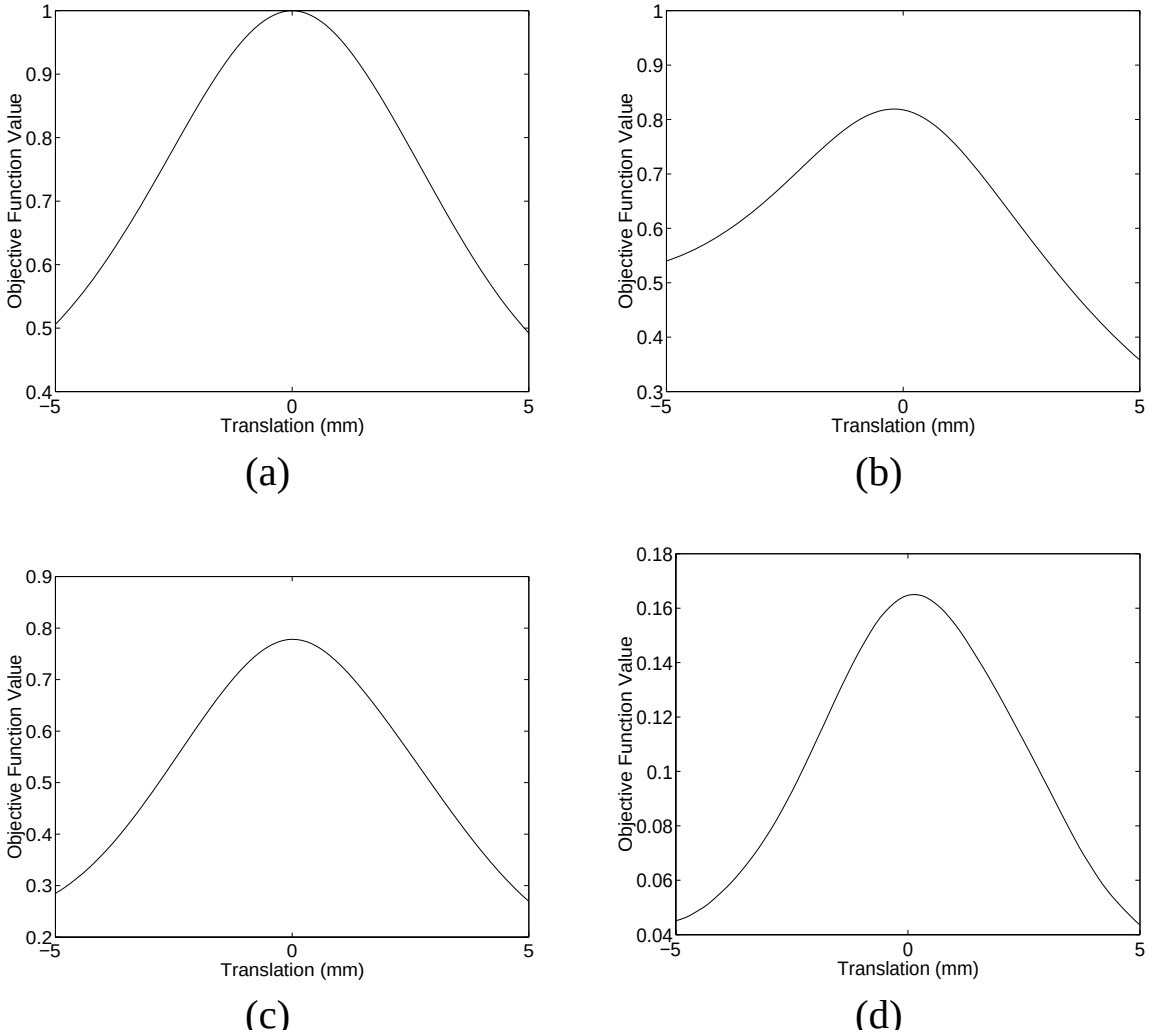



Figure 2.10: These graphs show how the variance-weighted sum of local normalized correlation value changes as the pelvis pose estimate is translated from left to right. The four graphs correspond to the four images in figure 2.6. The similarity peak matches the ideal position (0 mm translation) well for all four images.

define objective functions which can combine the SLNC or VLNC values from more than one image pair.

Prior to registration, input images are processed to remove geometric and intensity distortions so that at the processed image matches the actual X-ray attenuation as closely as possible, resulting in one or more processed images. We represent the $j^{\text{th}}$ processed image using the 2D function $U_j$. The values of each image $U_j$ reflect the attenuation of X-rays as they pass from the radiation source to the surface of the imager. For a given pose estimate $\gamma$, we denote the DRR corresponding to the $j^{\text{th}}$ input image by $U'_j(\gamma)$.

We define the SLNC objective function

$$E_{\text{SLNC}}(\gamma) = 1 - \frac{1}{R}\sum_{j=0}^{R-1} \text{SLNC}(U_j, U'_j(\gamma)), \tag{2.31}$$

where $R$ is the number of input X-ray images. The value of this objective function lies in the range $[0, 2]$, with smaller values indicating better matching between image pairs. Similarly, we define the VLNC objective function

$$E_{\text{VLNC}}(\gamma) = 1 - \frac{1}{R}\sum_{j=0}^{R-1} \text{VLNC}(U_j, U'_j(\gamma)). \tag{2.32}$$

As before, this function takes on values in the range $[0, 2]$, with smaller values indicating better matching between image pairs.

### 2.3.2 Specific Optimization Algorithms

Once an objective function is defined, we apply nonlinear optimizations methods from the literature to find its minimum. The most straightforward way to conduct this search is to use Brent's method, nonlinear simplex search, or some other non-gradient-based optimization routine to minimize the objective function directly. This is the approach chosen by Lemieux [35], Gilhuijs [19], and others. The principal disadvantage of this strategy is that non-gradient-based nonlinear optimization routines typically require many function evaluations in order to converge to a minimum. In our experiments, 6D registrations using the downhill simplex method of Nelder and Mead (as described in [45]) often require as many as three hundred function evaluations to converge, leading to unacceptably long registration times.

Fortunately, it is often possible to compute the first derivative of DRR pixel intensity with respect to the patient pose parameters, either by symbolic computation or by finite differences. By symbolically differentiating the objective function with respect to the DRR pixel values, and then

applying the chain rule, it is possible to compute the gradient of the objective function with respect to the patient pose parameters. This gradient information greatly speeds up the optimization by permitting the use of gradient based optimization routines.

Essentially, having gradient information makes it possible to update the pose estimate more intelligently, and decreases the number of function evaluations necessary before convergence. In our experiments, we minimize both the SLNC objective function and the VLNC objective function using the quasi-Newton method of Broyden, Fletcher, Goldfarb, and Shanno [45]. Using this method it is common for the 7D optimization to converge after fewer than 60 function evaluations and 20 gradient computations.

Quasi-Newton minimization requires that the first derivative of the objective function be computed. Repeated application of the chain rule to equations 2.22, 2.27, 2.29, and 2.31 gives us the first derivative of the SLNC objective function.

$$\frac{\partial E_{\mathrm{SLNC}}}{\partial \gamma_i} = -\frac{1}{R}\sum_{j=0}^{R-1}\frac{\partial}{\partial \gamma_i}\mathrm{SLNC}(U_j, U'_j(\gamma)) \tag{2.33}$$

$$\frac{\partial}{\partial \gamma_i}\mathrm{SLNC}(U_j, U'_j(\gamma)) = \frac{1}{|Q|}\sum_{\boldsymbol{p}\in Q}\frac{\partial}{\partial \gamma_i}\mathrm{NC}(U_j, U'_j(\gamma), P(\boldsymbol{p})) \tag{2.34}$$

$$\begin{aligned}\frac{\partial}{\partial \gamma_i}\mathrm{NC}(U_j, U'_j(\gamma), P(\boldsymbol{p})) \;=\; & \frac{\frac{\partial}{\partial \gamma_i}C\left(U_j, U'_j(\gamma), P(\boldsymbol{p})\right)}{\sqrt{C(U_j, U_j, P(\boldsymbol{p}))C(U'_j(\gamma), U'_j(\gamma), P(\boldsymbol{p}))}} \\ & -\frac{C\left(U_j, U'_j(\gamma), P(\boldsymbol{p})\right)C\left(U_j, U_j, P(\boldsymbol{p})\right)\frac{\partial}{\partial \gamma_i}C\left(U'_j(\gamma), U'_j(\gamma), P(\boldsymbol{p})\right)}{2\left(C(U_j, U_j, P(\boldsymbol{p}))C(U'_j(\gamma), U'_j(\gamma), P(\boldsymbol{p}))\right)^{\frac{3}{2}}}\end{aligned} \tag{2.35}$$

$$\frac{\partial}{\partial \gamma_i}C\left(U_j, U'_j(\gamma), P(\boldsymbol{p})\right) = \frac{1}{|P(\boldsymbol{p})|}\sum_{\boldsymbol{q}\in P(\boldsymbol{p})}U_j(\boldsymbol{q})\frac{\partial}{\partial \gamma_i}U'_j(\boldsymbol{q},\gamma) - \frac{1}{|P(\boldsymbol{p})|^2}\sum_{\boldsymbol{q}\in P(\boldsymbol{p})}U_j(\boldsymbol{q})\sum_{\boldsymbol{q}\in P(\boldsymbol{p})}\frac{\partial}{\partial \gamma_i}U'_j(\boldsymbol{q},\gamma), \tag{2.36}$$

where $\gamma_i$ is the $i^{\mathrm{th}}$ element of the pose parameter vector $\gamma$, $U_j(\boldsymbol{q})$ is the value of image $U_j$ at pixel location $\boldsymbol{q}$, $U'_j(\boldsymbol{q},\gamma)$ is the value of image $U'_j(\gamma)$ at pixel location $\boldsymbol{q}$, and all other variables are defined as in equations 2.20, 2.27, and 2.31.

To aid in differentiating $E_{\mathrm{VLNC}}$, we refer to equation 2.30 and define

$$S\left(I_0, I_1, P(\boldsymbol{p})\right) = \frac{C(I_0, I_1, P(\boldsymbol{p}))\sqrt{C(I_1, I_1, P(\boldsymbol{p}))}}{\sqrt{C(I_0, I_0, P(\boldsymbol{p}))}}. \tag{2.37}$$

Repeated application of the chain rule to equations 2.32, 2.37, and 2.28 gives

$$\frac{\partial E_{\mathrm{VLNC}}}{\partial \gamma_i} = -\frac{1}{R}\sum_{j=0}^{R-1}\frac{\partial}{\partial \gamma_i}\mathrm{VLNC}(U_j, U_j'(\gamma)) \tag{2.38}$$

$$\begin{aligned}\frac{\partial}{\partial \gamma_i}\mathrm{VLNC}(U_j, U_j'(\gamma)) &= \frac{\sum_{\boldsymbol{p}\in Q}\frac{\partial}{\partial \gamma_i}S(U_j, U_j'(\gamma), P(\mathbf{p}))}{\sum_{\boldsymbol{p}\in Q}\mathrm{C}(U_j, U_j'(\gamma), P(\mathbf{p}))} \\ &\quad - \frac{\sum_{\boldsymbol{p}\in Q} S(U_j, U_j'(\gamma), P(\mathbf{p}))\sum_{\boldsymbol{p}\in Q}\frac{\partial}{\partial \gamma_i}\mathrm{C}(U_j, U_j'(\gamma), P(\mathbf{p}))}{\left(\sum_{\boldsymbol{p}\in Q}\mathrm{C}(U_j, U_j'(\gamma), P(\mathbf{p}))\right)^2}\end{aligned} \tag{2.39}$$

$$\begin{aligned}\frac{\partial}{\partial \gamma_i}S(U_j, U_j'(\gamma), P(\mathbf{p})) &= \frac{\left(\frac{\partial}{\partial \gamma_i}C(U_j, U_j'(\gamma), P(\boldsymbol{p}))\right)\sqrt{C(U_j'(\gamma), U_j'(\gamma), P(\boldsymbol{p}))}}{\sqrt{C(U_j, U_j, P(\boldsymbol{p}))}} \\ &\quad + \frac{C(U_j, U_j'(\gamma), P(\boldsymbol{p}))\frac{\partial}{\partial \gamma_i}C(U_j'(\gamma), U_j'(\gamma), P(\boldsymbol{p}))}{2\sqrt{C(U_j, U_j, P(\boldsymbol{p}))C(U_j'(\gamma), U_j'(\gamma), P(\boldsymbol{p}))}},\end{aligned} \tag{2.40}$$

where, as before, all variables are defined as in equations 2.20, 2.27, and 2.31.

## 2.4 Discussion

This chapter describes how the X-ray/CT registration process is cast as an iterative nonlinear optimization problem. Finding the optimal patient pose is reduced to a problem of finding the vector of pose parameters which minimize an objective function.

Sections 2.1.1 and 2.1.2 present ways of representing patient pose as a vector of parameters, section 2.2 presents image comparison metrics which are efficiently computable and accurate even when image noise is quite high, and section 2.3.1 presents objective functions which take pose parameters as arguments and return scalar values indicating how well the pose parameters describe the input X-ray images. These objective functions are computed by rendering DRRs and then comparing the DRRs to the input images using image comparison metrics described in section 2.2. Section 2.3.2 presents the specific optimization functions used to do the minimization.

# Chapter 3

# Volume Rendering Using Transgraph

In our system, on-line DRRs are iteratively compared with X-ray images in order to estimate the position of the patient. Since DRRs must be recomputed at each iteration, the speed of the registration algorithm depends directly on how quickly DRRs can be generated. This chapter presents a software-only method of accelerated DRR generation based on a data structure which we call a Transgraph. The Transgraph is itself based on a data structure called a Lumigraph [20] or Light Field [36], which is part of the computer graphics field called *view-based rendering*. The Lumigraph was originally conceived to allow fast generation of reflectance images and we extend this idea to transmission imaging. This representation permits rapid generation of DRRs using data from the entire CT volume. One further advantage is that using the Transgraph permits easy differentiation of DRR pixel intensity with respect to patient pose parameters. These derivatives often permit differentiation of the image comparison metric. This, in turn, allows the use of gradient-based optimization routines in our registration algorithm, greatly speeding convergence.

This chapter begins by describing a ray-casting algorithm for DRR generation in section 3.1. Section 3.2 introduces the Transgraph, and section 3.3 presents a more detailed description of how the Transgraph is organized and used.

## 3.1 Computing DRRs by Ray Casting

We can model the diagnostic energy X-ray imaging process as a linear attenuation of X-rays as they pass through the patient's body. Under the linear attenuation model, each type of tissue has an associated *linear attenuation coefficient*, $\mu$, which describes the likelihood that a photon will be attenuated as it passes through the tissue. Imagine that number of photons, $N_{\text{in}}$, enters a uniformly thick slab of tissue as shown in figure 3.1. If the tissue has uniform linear attenuation coefficient, $\mu$,

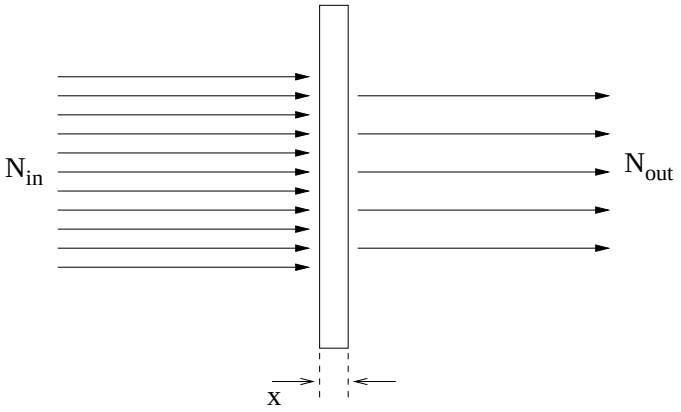



Figure 3.1: Only some of the photons which enter a slab of attenuating tissue continue on their path. In this illustration a number of photons, $N_{\text{in}}$, enters a slab of attenuating mater having thickness $x$. Some of the photons are attenuated, and the remainder, $N_{\text{out}}$, continue on their path.

we can describe the attenuation through the slab

$$N_{\text{out}} = N_{\text{in}} \text{e}^{\mu x}, \tag{3.1}$$

where $N_{\text{out}}$ is the number of unattenuated photons, and $x$ is the thickness of the slab [29]. In general, patients are not made of uniformly thick slabs of tissue, and so the quantity $\mu x$ in equation 3.1 will be replaced with a more complicated expression, such as a line integral. We call this quantity the *log total attenuation*, and represent it with the symbol $U$

$$N_{\text{out}} = N_{\text{in}} \text{e}^{U}. \tag{3.2}$$

When generating a DRR, we know (or hypothesize) the geometry of the X-ray imaging system, the patient pose, and other imaging parameters. The DRR is intended to answer the question "if we were to take an actual X-ray image, what would it look like?" We think of each ray as starting at the radiation source, and passing through space to a particular point on the imager as shown in figure 3.2. In other words, if we could trace a line from the radiation source to a point on the imager, and "add up" the attenuation of the ray at each point along the line, then we could predict the total attenuation of the radiation incident on that part of the imager. We assume here that the effects of scatter and differential absorption across the energy spectrum of the X-rays (beam hardening) are insignificant.

For now, assume that the geometry of the imaging system and the specifics of the imaging radiation are known. Assume also that the position and orientation of the patient are specified by a parameter vector, $\gamma$. The 3D structure of the patient, and the approximate linear attenuation

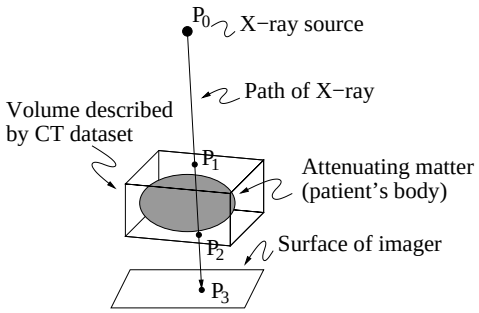



Figure 3.2: Path of a single ray from radiation source to imager. The box indicates the volume in space which is represented by the CT dataset. $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ represent the points at which the ray enters and exits this volume.

coefficients of the component tissues, are available from preoperative CT. We predict an entire x-ray image by considering each pixel independently and computing the log total attenuation along the ray which joins the corresponding point on the surface of the imager to the x-ray source. This process can be broken into steps as follows.

1. The point on the surface of the imager which corresponds to the center of the current pixel is found. This point is labeled $\boldsymbol{p}_3$ in figure 3.2.

2. The ray is computed which connects the location of the X-ray source, $\boldsymbol{p}_0$, with $\boldsymbol{p}_3$.

3. Calculations are performed to find the points $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$, at which the ray from step 2 enters and exits the volume described by the CT dataset. Note that these points depend on $\gamma$, $\boldsymbol{p}_0$, and $\boldsymbol{p}_3$, and could be written $\boldsymbol{p}_1(\boldsymbol{p}_0, \boldsymbol{p}_3, \gamma)$ and $\boldsymbol{p}_2(\boldsymbol{p}_0, \boldsymbol{p}_3, \gamma)$. We omit the parameterization simply to make the expression easier to write.

4. Numerical integration is performed along the length of the ray. The quantity integrated is the linear attenuation coefficient at each point along the ray. The linear attenuation coefficient at a given point depends on both the type of tissue at that part of the patient and the energy of the X-rays emitted from the X-ray source [24], however it can be closely approximated by linearly scaling the CT value. For points outside the CT volume, the linear attenuation coefficient of air should be used.

$$U_{\text{tot}}(\boldsymbol{p}_3, \gamma) = \|\boldsymbol{p}_1 - \boldsymbol{p}_0\| \, \mu_{\text{air}} + \|\boldsymbol{p}_3 - \boldsymbol{p}_2\| \, \mu_{\text{air}} + U_{\text{ct}}(\boldsymbol{p}_1, \boldsymbol{p}_2), \tag{3.3}$$

$$U_{\text{ct}}(\boldsymbol{p}_1, \boldsymbol{p}_2) = \int_0^{||\boldsymbol{p}_2 - \boldsymbol{p}_1||} \mu_{\text{ct}}(p(s, \boldsymbol{p}_1, \boldsymbol{p}_2)) ds. \tag{3.4}$$

$$p(s, \boldsymbol{p}_1, \boldsymbol{p}_2) = \boldsymbol{p}_1 + s \frac{\boldsymbol{p}_2 - \boldsymbol{p}_1}{||\boldsymbol{p}_2 - \boldsymbol{p}_1||}, \tag{3.5}$$

where $U_{\mathrm{tot}}(\boldsymbol{p}_3, \gamma)$ is the log total attenuation (see equations 3.1 and 3.2) along the path from the X-ray source to $\boldsymbol{p}_3$, give pose parameters $\gamma$. The constant $\mu_{air}$ is the linear attenuation coefficient of air, and $\mu_{ct}(\boldsymbol{p})$ is the linear attenuation coefficient derived from the CT value at point $\boldsymbol{p}$. The function $p(s, \boldsymbol{p}_1, \boldsymbol{p}_2)$ is a parameterization of the line segment which passes from $\boldsymbol{p}_1$ to $\boldsymbol{p}_2$. $U_{tot}(\boldsymbol{p}_3, \gamma)$ determines the reduction in beam intensity along the ray which passes from $\boldsymbol{p}_0$ to $\boldsymbol{p}_3$. For nearly all practical cases, $\mu_{air}$ is equal to zero [24], so we have

$$U_{tot}(\boldsymbol{p}_3, \gamma) = U_{ct}(\boldsymbol{p}_1, \boldsymbol{p}_2), \tag{3.6}$$

5. The photon fluence at the surface of the imager is computed by substituting the result of step 4 into an exponential attenuation rule [29][18]:

$$f(\boldsymbol{p}_3, \gamma) = f_0 \left( \frac{r_0}{\|\boldsymbol{p}_3 - \boldsymbol{p}_0\|} \right)^2 exp(-U_{tot}(\boldsymbol{p}_3, \gamma)), \tag{3.7}$$

where $f(\boldsymbol{p}_3, \gamma)$ is the photon fluence at point $\boldsymbol{p}_3$, and $f_0$ is the unattenuated photon fluence at a known distance $r_0$ from the X-ray source.

6. Using the photon fluence from step 5, the output pixel value is computed according the the characteristics of the imaging system.

## 3.2 The Transgraph

In practice, 3D datasets based on CT are often quite large, and computing one DRR may involve integrating through millions of voxels. This process can be very slow, especially if an attempt is made to accurately interpolate the sampled CT data. In our experiments, the computation of a 256x256 DRR using a 512x512x100 voxel CT volume requires 10-15 seconds on an SGI O2 R10000, even under the assumption of uniform linear attenuation coefficient within each voxel. The ray tracing and numerical integration for this computation were implemented using the fast voxel traversal algorithm of Amanatides and Woo [5]. Because of its high computational cost, the naive DRR generation procedure outlined above is too slow for interactive computation of DRRs, when computation times of a fraction of a second may be required. This is especially true in the context of our iterative registration algorithm, which requires many DRRs to be generated.

If we could precompute the line integrals described in step 4, above, then DRR generation could be much faster. The integration and interpolation of CT values could be done off-line, and the result stored. Later, during DRR generation, the precomputed values could be rapidly assembled to produce the desired image.

In other words, equation 3.4 defines a scalar function whose parameters are related to the entry and exit points on the surface of the CT volume. We can speed up the generation of DRRs by densely sampling this function, and recording its value at each sample point. When a value is needed for DRR generation, we simply interpolate among the stored values. Since this interpolation can be executed much more quickly than the actual line integral, we speed up DRR generation tremendously. We call this database of function values a Transgraph. For comparison, the DRR mentioned above, which requires 10-15 seconds to compute by ray-casting, can be generated in roughly 0.2 seconds using the Transgraph.

Note that storing a database of line integral values is very different from storing a library of precomputed DRRs. In order to be useful for registration, a DRR library must contain entries reflecting the entire range of expected patient motion. For six degrees-of-freedom registration, this requires an unreasonably large library. For example, a uniformly sampled six-dimensional DRR library having only ten sample points along each axis would require $10^6$ DRRs, and use many gigabytes of storage. For this reason, it is not practical to precompute a 6D database of images.

We will show in section 3.2.1 that a line integral database of only 4 dimensions is sufficient to reconstruct the full set DRRs required for registration.

### 3.2.1 A 4D Database

In equation 3.4, $U_{ct}(\boldsymbol{p}_1, \boldsymbol{p}_2)$ is parameterized by two points in 3D space, $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$. Each point has three degrees of freedom, so the total dimensionality of the function $U_{ct}$ is six. The fact that both $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ are constrained to lie on the boundary of the CT volume is not reflected in this parameterization. For example, points $\boldsymbol{p}_0$ and $\boldsymbol{p}_3$ lie on the same line as points $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$, and therefore correspond to the same path through the CT volume (and the same value of $U_{ct}$) yet the pairs $(\boldsymbol{p}_1, \boldsymbol{p}_2)$ and $(\boldsymbol{p}_0, \boldsymbol{p}_3)$ represent different points in this six-dimensional parameter space. If we are to sample and reconstruct $U_{ct}$ efficiently, we must not over-parameterize in this way.

In fact, we can represent the database using a lookup table of only four dimensions. To see this, consider figure 3.3, where we have defined two parallel coordinate planes, $C_0$ and $C_1$. Any trajectory through the CT volume, with the exception of trajectories which are parallel to $C_0$ and $C_1$, can be represented by $\boldsymbol{q}_0$ and $\boldsymbol{q}_1$, its points of intersection with these coordinate planes. Since $\boldsymbol{q}_0$ and $\boldsymbol{q}_1$ are 2D points in the coordinate systems of $C_0$ and $C_1$ respectively, the total dimensionality of this parameterization is 4. Horizontal rays cannot be represented using this particular parameterization, but this could easily be remedied by defining a more sophisticated indexing scheme, and is of no consequence in our application, where incident rays fall over a well defined set of orientations. This indexing scheme is attractive because of its simplicity, and because it is computationally inexpensive to compute the intersection of a given ray with each indexing plane. This makes it easy to compute

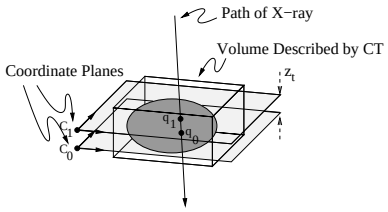



Figure 3.3: Two coordinate planes can be used to parameterize the Transgraph.

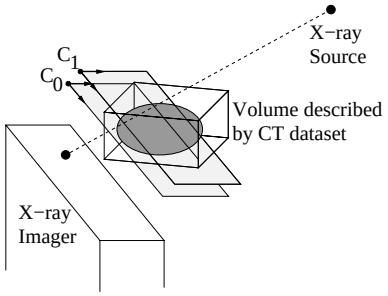



Figure 3.4: One possible Transgraph coordinate plane configuration

Transgraph indices during DRR generation, when time is at a premium.

Since we cannot represent rays which lie parallel to the indexing planes, it is important to choose the indexing planes carefully. The choice of indexing planes depends on the expected range patient poses with respect to the imaging hardware.

We select the indexing planes by first defining a nominal patient pose. One convenient way to generate this nominal pose is to represent the extrema of the expected pose parameters as points in parameter space, and choose the mean of these points. Once the nominal pose has been selected, we note the orientation of the imager surface with respect to the CT volume, and choose indexing plane $C_0$ to be parallel to the imager surface. We also constrain $C_0$ to pass through the exact center of the CT volume, although this choice is somewhat arbitrary. The second indexing plane, $C_1$, is chosen to lie parallel to $C_0$, but offset by a small amount, which we denote $z_\mathrm{t}$. In our work we choose $z_\mathrm{t} = 1\,\mathrm{mm}$, although this choice is again arbitrary. For illustration, possible positions of $C_0$ and $C_1$ are shown in figure 3.4.

Note that it is not necessary to precompute every possible line integral. It is only necessary to precompute those which lie in regions of the Transgraph that we expect to use for DRR generation. The Transgraphs used in this study contain roughly $2 \times 10^7$ samples, and consume about $40\,\mathrm{MB}$ of memory.

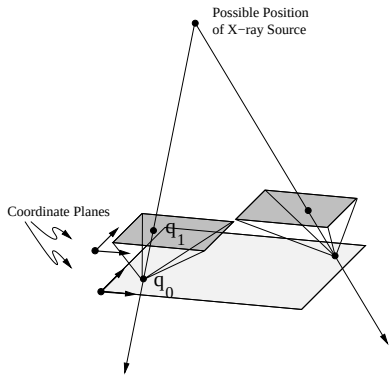



Figure 3.5: The Transgraph is implemented as a 2D array of 2D arrays. Each element of the first array corresponds to a point $\boldsymbol{q}_0$ in the $C_0$ coordinate plane, and contains a 2D sub-array which describes a region of the $C_1$ coordinate plane.

## 3.3 Implementation Details

The Transgraph is implemented as a nested data structure. The top-level structure is a 2D array. The indices of this array correspond to coordinates in the $C_0$ coordinate frame (see figures 3.3 and 3.4). Each element in this array is itself a 2D array, and the indices in each sub-array correspond to coordinates in the $C_1$ coordinate frame. Each element of a particular sub-array is a numerical value representing the total attenuation along the corresponding ray.

Put another way, each element of the top-level 2D array corresponds to a point $\boldsymbol{q}_0$ in $C_0$, and is itself a 2D array describing the set of ray trajectories which pass through $\boldsymbol{q}_0$. Each element of the sub-array corresponds to some point $\boldsymbol{q}_1$ in $C_1$, and contains a numeric value representing the attenuation of an X-ray as it passes through the CT volume along the line which intersects $C_0$ at $\boldsymbol{q}_0$ and intersects $C_1$ at $\boldsymbol{q}_1$. This is illustrated in figure 3.5.

The total attenuation along the path from the X-ray source to the imager surface depends on both the attenuation due to to the CT volume and the attenuation due to air. In most cases, the linear attenuation coefficient of air is effectively zero, and this second term can be disregarded. When the linear attenuation coefficient of air is not zero, the total attenuation due to air can be conveniently computed by subtracting the distance traveled through the CT volume from the total distance between the X-ray source and the imaging surface (see equation 3.3). In these cases, it is useful to precompute the "distance through the CT volume" for each ray in the Transgraph, and store this value along with the numerical attenuation value.

### 3.3.1 Minimizing Storage Space

The effectiveness of the Transgraph depends entirely on which samples are included in the database. These samples must be chosen so that they completely cover the regions of 4D ray-space which will be needed during image generation. Furthermore, the sampling density must be sufficient to accurately reproduce the 4D function throughout this range.

The relevant region of the 4D space can be calculated based on the expected range of patient poses. To help with this calculation, we define a 3D coordinate system associated with the Transgraph. The origin of this coordinate system is coincident with the origin of the $C_0$ coordinate system, and its $X$ and $Y$ axes are parallel to the $X$ and $Y$ axes of the $C_0$ coordinate system. The position and orientation of this coordinate system are fixed with respect to the coordinate system of the CT volume, and we define a 4x4 transformation matrix ${}^{\mathrm{tg}}T_{\mathrm{ct}}$ which transforms coordinates in the coordinate system of the CT volume to coordinates in the coordinate system of the Transgraph.

Any given pose of the patient with respect to the imaging system corresponds to a transformation matrix ${}^{\mathrm{ct}}T_{\mathrm{im}}$ which transforms coordinates in the coordinate system of the imager to coordinates in the coordinate system of the CT volume.

The two coordinate transformations, ${}^{\mathrm{tg}}T_{\mathrm{ct}}$ and ${}^{\mathrm{ct}}T_{\mathrm{im}}$ can be composed to find the coordinate transformation ${}^{\mathrm{tg}}T_{\mathrm{im}}$ which relates the Transgraph coordinate system to the X-ray imager coordinate system. Under this coordinate transformation, the rectangular imager surface projects to a tetragonal region of the $C_0$ coordinate plane. This tetragon is always convex, and each of its vertices is the projection of one of the four corners of the rectangular imaging surface. Similarly, the volume described by the CT dataset projects into a convex polygonal region of the $C_0$ coordinate plane. These projections are illustrated in figure 3.6. As the CT volume and Transgraph traverse the 6D space of patient poses, these two projections change position and shape. The intersection of these two polygons is the region of $C_0$ where which contributes to DRR computation at that particular pose. The union of these regions over all possible poses defines the region of $C_0$ which must be populated with samples. We call this the *active region* of $C_0$.

In the current Transgraph implementation, we represent the active region of the $C_0$ coordinate plane not as an arbitrary polygon, but rather as a rectangular area. This choice is convenient because the rectangular $C_0$ region is easily represented using a 2D array. The active region is chosen to be a rectangle which bounds the convex hull of the projected CT corners, where the projection is taken over the set of possible CT poses. We compute this rectangle by coarsely sampling the space of pose parameters, projecting each CT corner at each sample pose, and computing the minimum and maximum $C_0$ coordinates of the projected vertices over all of the sample poses. Finally, a safety margin is added to the minimum and maximum coordinates to account for poses which were not included in the sample set.

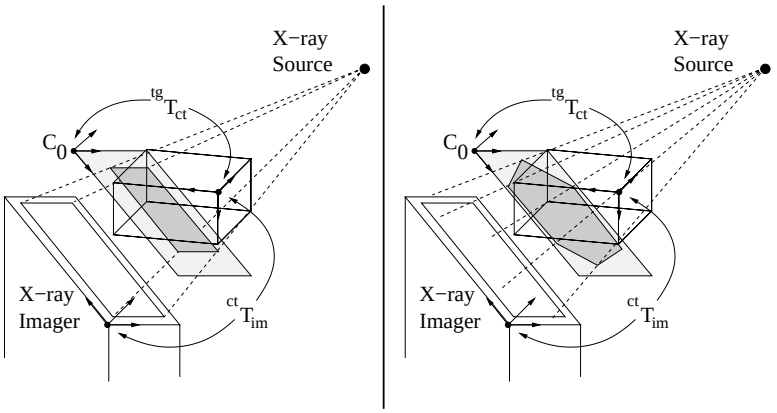



Figure 3.6: The the imaging surface and the volume described by the CT both project into convex polygons in the $C_0$ coordinate plane. The shape and location of these polygons depend on the pose of the CT with respect to the imager, ${}^{\mathrm{ct}}T_{\mathrm{im}}$, and the pose of the Transgraph with respect to the CT volume, ${}^{\mathrm{tg}}T_{\mathrm{ct}}$.

In addition to selecting the active region of the $C_0$ coordinate plane, a similar determination must be made for each of the $C_1$ sub-planes. In the current implementation, after the $C_0$ active region has been selected, the imager surface is reprojected at each of the sample poses. The minimum and maximum $C_1$ coordinates over the set of poses are recorded for each $C_1$ sub-plane, and the $C_1$ coordinate sub-plane bounding boxes are chosen accordingly.

### 3.3.2 Quadrilinear Interpolation

During image synthesis, the values drawn from the Transgraph are recovered by interpolating among the precomputed samples. In the interest of computational speed, we currently use quadrilinear interpolation.

Quadrilinear interpolation is straightforward to implement. We define the 4D discrete function $T$, which represents the sampled values in the Transgraph. $T$ is defined only for integer indices. That is, $T[0,0,0,0]$, $T[0,0,0,1]$, and $T[4,3,6,2]$ represent valid samples in the Transgraph, while $T[2,3,1.2,0]$ does not. We can write the interpolated value at general coordinates $(u,v,s,t)$ as follows:

$$
\begin{aligned}
f(s,t,u,v) \;=\; & (1-a)(1-b)(1-c)(1-d) * T[\lfloor s \rfloor, \lfloor t \rfloor, \lfloor u \rfloor, \lfloor v \rfloor] \\
& +(1-a)(1-b)(1-c)d * T[\lfloor s \rfloor, \lfloor t \rfloor, \lfloor u \rfloor, \lfloor v \rfloor + 1] \\
& +(1-a)(1-b)c(1-d) * T[\lfloor s \rfloor, \lfloor t \rfloor, \lfloor u \rfloor + 1, \lfloor v \rfloor] \\
& +(1-a)(1-b)cd * T[\lfloor s \rfloor, \lfloor t \rfloor, \lfloor u \rfloor + 1, \lfloor v \rfloor + 1] \\
& +(1-a)b(1-c)(1-d) * T[\lfloor s \rfloor, \lfloor t \rfloor + 1, \lfloor u \rfloor, \lfloor v \rfloor] \\
& +(1-a)b(1-c)d * T[\lfloor s \rfloor, \lfloor t \rfloor + 1, \lfloor u \rfloor, \lfloor v \rfloor + 1] \\
& +(1-a)bc(1-d) * T[\lfloor s \rfloor, \lfloor t \rfloor + 1, \lfloor u \rfloor + 1, \lfloor v \rfloor] \\
& +(1-a)bcd * T[\lfloor s \rfloor, \lfloor t \rfloor + 1, \lfloor u \rfloor + 1, \lfloor v \rfloor + 1] \\
& +a(1-b)(1-c)(1-d) * T[\lfloor s \rfloor + 1, \lfloor t \rfloor, \lfloor u \rfloor, \lfloor v \rfloor] \\
& +a(1-b)(1-c)d * T[\lfloor s \rfloor + 1, \lfloor t \rfloor, \lfloor u \rfloor, \lfloor v \rfloor + 1] \\
& +a(1-b)c(1-d) * T[\lfloor s \rfloor + 1, \lfloor t \rfloor, \lfloor u \rfloor + 1, \lfloor v \rfloor] \\
& +a(1-b)cd * T[\lfloor s \rfloor + 1, \lfloor t \rfloor, \lfloor u \rfloor + 1, \lfloor v \rfloor + 1] \\
& +ab(1-c)(1-d) * T[\lfloor s \rfloor + 1, \lfloor t \rfloor + 1, \lfloor u \rfloor, \lfloor v \rfloor] \\
& +ab(1-c)d * T[\lfloor s \rfloor + 1, \lfloor t \rfloor + 1, \lfloor u \rfloor, \lfloor v \rfloor + 1] \\
& +abc(1-d) * T[\lfloor s \rfloor + 1, \lfloor t \rfloor + 1, \lfloor u \rfloor + 1, \lfloor v \rfloor] \\
& +abcd * T[\lfloor s \rfloor + 1, \lfloor t \rfloor + 1, \lfloor u \rfloor + 1, \lfloor v \rfloor + 1].
\end{aligned}
\tag{3.8}
$$

The interpolation coefficients $a$, $b$, $c$, and $d$ are defined simply

$$
a = s - \lfloor s \rfloor\,,\; b = t - \lfloor t \rfloor\,,\; c = u - \lfloor u \rfloor\,,\; d = v - \lfloor v \rfloor\,. \tag{3.9}
$$

This formulation of quadrilinear interpolation is simple to write, but is computationally inefficient. As written, each interpolation requires 48 multiplications and 46 additions in addition to the overhead is required to compute the 16 sets of indices (such as $[\lfloor s \rfloor, \lfloor t \rfloor + 1, \lfloor u \rfloor, \lfloor v \rfloor + 1]$), and to actually index into the Transgraph data structure at each of the 16 locations. By making use of the identity

$$
(1-a)x + ay = a(y-x) + x, \tag{3.10}
$$

equation 3.8 can be rearranged to give

$$f(s,t,u,v) = a(r_0 - r_1) + r_1 \tag{3.11}$$

$$r_i = b(q_{2i+1} - q_{2i}) + q_{2i}, \qquad 0 \le i < 2 \tag{3.12}$$

$$\boldsymbol{q}_i = c(q_{2i+1} - q_{2i}) - q_{2i}, \qquad 0 \le i < 4 \tag{3.13}$$

$$\boldsymbol{p}_i = d(o_{2i+1} - o_{2i}) + o_{2i}, \qquad 0 \le i < 8, \tag{3.14}$$

where the $o_i$ are the actual Transgraph elements, as follows:

$$\begin{aligned}
o_0 &= T[\lfloor s \rfloor, \lfloor t \rfloor, \lfloor u \rfloor, \lfloor v \rfloor] & o_1 &= T[\lfloor s \rfloor, \lfloor t \rfloor, \lfloor u \rfloor, \lfloor v \rfloor + 1] \\
o_2 &= T[\lfloor s \rfloor, \lfloor t \rfloor, \lfloor u \rfloor + 1, \lfloor v \rfloor] & o_3 &= T[\lfloor s \rfloor, \lfloor t \rfloor, \lfloor u \rfloor + 1, \lfloor v \rfloor + 1] \\
o_4 &= T[\lfloor s \rfloor, \lfloor t \rfloor + 1, \lfloor u \rfloor, \lfloor v \rfloor] & o_5 &= T[\lfloor s \rfloor, \lfloor t \rfloor + 1, \lfloor u \rfloor, \lfloor v \rfloor + 1] \\
o_6 &= T[\lfloor s \rfloor, \lfloor t \rfloor + 1, \lfloor u \rfloor + 1, \lfloor v \rfloor] & o_7 &= T[\lfloor s \rfloor, \lfloor t \rfloor + 1, \lfloor u \rfloor + 1, \lfloor v \rfloor + 1] \\
o_8 &= T[\lfloor s \rfloor + 1, \lfloor t \rfloor, \lfloor u \rfloor, \lfloor v \rfloor] & o_9 &= T[\lfloor s \rfloor + 1, \lfloor t \rfloor, \lfloor u \rfloor, \lfloor v \rfloor + 1] \\
o_{10} &= T[\lfloor s \rfloor + 1, \lfloor t \rfloor, \lfloor u \rfloor + 1, \lfloor v \rfloor] & o_{11} &= T[\lfloor s \rfloor + 1, \lfloor t \rfloor, \lfloor u \rfloor + 1, \lfloor v \rfloor + 1] \\
o_{12} &= T[\lfloor s \rfloor + 1, \lfloor t \rfloor + 1, \lfloor u \rfloor, \lfloor v \rfloor] & o_{13} &= T[\lfloor s \rfloor + 1, \lfloor t \rfloor + 1, \lfloor u \rfloor, \lfloor v \rfloor + 1] \\
o_{14} &= T[\lfloor s \rfloor + 1, \lfloor t \rfloor + 1, \lfloor u \rfloor + 1, \lfloor v \rfloor] & o_{15} &= T[\lfloor s \rfloor + 1, \lfloor t \rfloor + 1, \lfloor u \rfloor + 1, \lfloor v \rfloor + 1].
\end{aligned} \tag{3.15}$$

Equation 3.11 can be evaluated with only 15 multiplications and 30 additions, plus the same indexing and lookup overhead.

### 3.3.3 Computing Derivatives

During registration (see chapter 2), it will be useful to evaluate the first derivative of $f()$ with respect to the parameters $u$, $v$, $s$, and $t$. For quadrilinear interpolation, these derivatives can be found by computing additional linear combinations of the same 16 neighboring samples in the Transgraph. Specifically, we write:

$$\frac{\partial f}{\partial s} = (r_0 - r_1)\frac{\partial a}{\partial s} \tag{3.16}$$

$$\frac{\partial a}{\partial s} = 1, \qquad a \neq \lfloor a \rfloor$$

$$\begin{aligned} \frac{\partial f}{\partial t} &= a(\frac{\partial r_0}{\partial t} - \frac{\partial r_1}{\partial t}) + \frac{\partial r_0}{\partial t} \\ \frac{\partial r_i}{\partial t} &= (q_{2i+1} - q_{2i})\frac{\partial b}{\partial t}, \qquad 0 \le i < 2 \\ \frac{\partial b}{\partial t} &= 1, \qquad b \neq \lfloor b \rfloor \end{aligned} \tag{3.17}$$

$$\begin{aligned} \frac{\partial f}{\partial u} &= a(\frac{\partial r_0}{\partial u} - \frac{\partial r_1}{\partial u}) + \frac{\partial r_0}{\partial u} \\ \frac{\partial r_i}{\partial u} &= b(\frac{\partial q_{2i+1}}{\partial u} - \frac{\partial q_{2i}}{\partial u}) + \frac{\partial q_{2i+1}}{\partial u}, \qquad 0 \le i < 2 \\ \frac{\partial q_i}{\partial u} &= (p_{2i+1} - p_{2i})\frac{\partial c}{\partial u}, \qquad 0 \le i < 4 \\ \frac{\partial c}{\partial u} &= 1, \qquad c \neq \lfloor c \rfloor \end{aligned} \tag{3.18}$$

$$\begin{aligned} \frac{\partial f}{\partial v} &= a(\frac{\partial r_0}{\partial v} - \frac{\partial r_1}{\partial v}) + \frac{\partial r_0}{\partial v} \\ \frac{\partial r_i}{\partial v} &= b(\frac{\partial q_{2i+1}}{\partial v} - \frac{\partial q_{2i}}{\partial v}) + \frac{\partial q_{2i+1}}{\partial v}, \qquad 0 \le i < 2 \\ \frac{\partial q_i}{\partial v} &= b(\frac{\partial p_{2i+1}}{\partial v} - \frac{\partial p_{2i}}{\partial v}) + \frac{\partial p_{2i+1}}{\partial v}, \qquad 0 \le i < 4 \\ \frac{\partial p_i}{\partial v} &= (o_{2i+1} - o_{2i})\frac{\partial d}{\partial v}, \qquad 0 \le i < 8 \\ \frac{\partial d}{\partial v} &= 1, \qquad d \neq \lfloor d \rfloor \end{aligned} \tag{3.19}$$

These quantities can be efficiently calculated with an additional computational cost of only 11 multiplications and 22 additions. We ignore the degenerate cases $a = \lfloor a \rfloor$, $b = \lfloor b \rfloor$, $c = \lfloor c \rfloor$, and $d = \lfloor d \rfloor$.

### 3.3.4 Optimizing Access to Transgraph Elements

The interpolation equations above require access to 16 Transgraph elements for each recovered attenuation value. A typical 2D indexing operation requires one multiplication and one addition. Since the Transgraph is implemented as a 2D array of 2D arrays, each 4D indexing operation involves two multiplications, two additions, and some dereferencing overhead. The straightforward implementation incurs this cost for each of the 16 samples. We can reduce this overhead by observing that elements are always accessed in a 2D neighborhood of four 2D neighborhoods, each comprising four adjacent Transgraph elements. Consequently, it is not necessary to independently locate each element. Our current implementation spends 5 multiplications and 20 additions, plus

overhead for pointer dereferencing and memory access, to recover all 16 values. We currently make no attempt to model and optimize cache performance.

## 3.4 Generating DRRs using the Transgraph

Once a parameterization for patient pose has been selected, the position and orientation of the CT volume with respect to the X-ray imagers can be written as a function of the pose parameters. Since the position and orientation of the indexing planes are known with respect to the CT volume, the position of the X-ray source can be found relative to the two indexing planes.

Similarly, an array of pixel locations on the imaging surface can be defined, and the position of each pixel location can be found relative to the two indexing planes. Line segments are defined which connect the X-ray source to each pixel location, just as a line segment is defined connecting points $\boldsymbol{p}_0$ and $\boldsymbol{p}_3$ in figure 3.2. The points where this line segment intersects the two indexing planes correspond to the points labeled $\boldsymbol{q}_0$ and $\boldsymbol{q}_1$ in figure 3.3. These points determine a unique 4D point in the parameter space of the Transgraph. The DRR generation process can be broken into steps, as follows.

1. Find the point, $\boldsymbol{p}_3$, on the surface of the imager which corresponds to the center of the current pixel.

2. Find the line segment which connects the location of the X-ray source, $\boldsymbol{p}_0$, with $\boldsymbol{p}_3$.

3. Calculate the 2D points $\boldsymbol{q}_0$ and $\boldsymbol{q}_1$, at which the ray from step 2 intersects the $C_0$ and $C_1$ planes of the Transgraph.

4. Using the points $\boldsymbol{q}_0$ and $\boldsymbol{q}_1$ from step 3, compute the corresponding indices $(u,\, v,\, s,\, t)$ into the 4D Transgraph.

5. Find the total attenuation along the ray by quadrilinear interpolation.

$$U_{\mathrm{ct}}(\boldsymbol{q}_0, \boldsymbol{q}_1) = f(u, v, s, t), \tag{3.20}$$

   where $f(u, v, s, t)$ is the value recovered from the Transgraph as defined in equation 3.11.

6. If desired, apply further processing as described in steps 5 and 6 of section 3.1.

We address steps 1 and 2 in section 3.4.1, and describe the remaining steps in section 3.4.2.

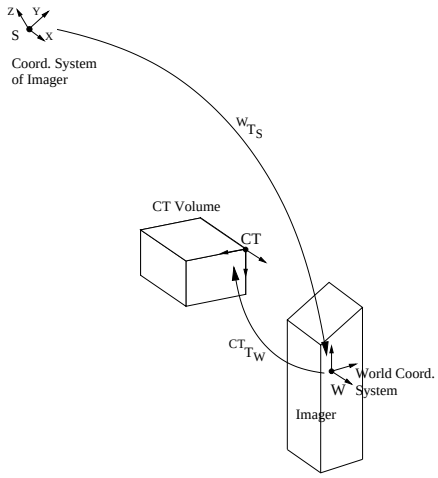



Figure 3.7: The patient pose parameters specify the position and orientation of the CT volume with respect to the world coordinate system, W. The world coordinate system which is defined with respect to the coordinate system of the imager.

### 3.4.1 Defining Line Segments in Transgraph Coordinates

Typically, we specify the position and orientation of the CT volume with respect to a stationary *world coordinate* system, $W$. The coordinate transformation relating the world coordinate system with the 3D coordinate system of each imager is assumed to be known, so that the position and orientation of the CT volume can be found with respect to each imager as shown in figure 3.7.

Following step 1 above, we define an array of pixel positions in the 3D coordinate system each imager. Referring to figure 3.7, we see that the 3D pixel positions in each imager are easily transformed into the world coordinate system. We write these transformed points

$$ {}^{\mathrm{w}}\boldsymbol{p}_{3,i} = \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix}, \tag{3.21} $$

where the additional subscript $i$ indicates that this is the $i^{\mathrm{th}}$ pixel location, and the left superscript $\mathrm{w}$ indicates that the point is expressed in world coordinates. Similarly, we represent the position of the imager X-ray source in world coordinates

$$ {}^{\mathrm{w}}\boldsymbol{p}_{0} = \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix}. \tag{3.22} $$

In order to recover attenuation values from the Transgraph, we need to find the points at which the line segment connecting ${}^{\mathrm{w}}\boldsymbol{p}_{3,i}$ and ${}^{\mathrm{w}}\boldsymbol{p}_0$ intersects the Transgraph coordinate planes $C_0$ and $C_1$. We write this line segment parametrically

$$ {}^{\mathrm{w}}l_i(\lambda) = {}^{\mathrm{w}}\,\boldsymbol{p}_0 + \lambda\left({}^{\mathrm{w}}\boldsymbol{p}_{3,i} - {}^{\mathrm{w}}\,\boldsymbol{p}_0\right) = \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} + \lambda \begin{bmatrix} x_i - x' \\ y_i - y' \\ z_i - z' \end{bmatrix}, \tag{3.23} $$

$$ 0 \leq \lambda \leq 1 $$

where the line line segment $l$ is parameterized by $\lambda$. This equation can be rewritten in homogeneous coordinates

$$ {}^{\mathrm{w}}l_i(\lambda) = \begin{bmatrix} x' + \lambda(x_i - x') \\ y' + \lambda(y_i - y') \\ z' + \lambda(z_i - z') \\ 1 \end{bmatrix}. \tag{3.24} $$

### 3.4.2 Recovering Transgraph Coordinates

The coordinate transformation ${}^{\mathrm{tg}}T_{\mathrm{ct}}$ is defined in section 3.3. Here, we represent each row of the 4x4 matrix representation of ${}^{\mathrm{tg}}T_{\mathrm{ct}}$ as a four element vector:

$$ {}^{\mathrm{tg}}T_{\mathrm{ct}} = \begin{bmatrix} & R_0^T & \\ & R_1^T & \\ & R_2^T & \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.25} $$

Composing ${}^{\mathrm{ct}}T_{\mathrm{w}}(\gamma)$ of from section 2.1 with ${}^{\mathrm{tg}}T_{\mathrm{ct}}$, we have

$$ {}^{\mathrm{tg}}T_{\mathrm{w}}(\gamma) = {}^{\mathrm{tg}}\,T_{\mathrm{ct}} * {}^{\mathrm{ct}}\,T_{\mathrm{w}}(\gamma) \tag{3.26} $$

For the remainder of this chapter, we assume that patient pose is described by the seven element parameterization $[t_{\mathrm{x}},\, t_{\mathrm{y}},\, t_{\mathrm{z}},\, s,\, i,\, j,\, k]^T$ (see section 2.1.2). The following discussion is, however, easily extended to other parameterizations. Under this assumption, equation 2.18 can be substituted

into equation 3.26, to give

$$ {}^{\mathrm{tg}}T_{\mathrm{im}} = \begin{bmatrix} & R_0^T & \\ & R_1^T & \\ & R_2^T & \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} A_0 & A_1 & A_2 & A_3 \end{bmatrix} = \begin{bmatrix} R_0^T A_0 & R_0^T A_1 & R_0^T A_2 & R_0^T A_3 \\ R_1^T A_0 & R_1^T A_1 & R_1^T A_2 & R_1^T A_3 \\ R_2^T A_0 & R_2^T A_1 & R_2^T A_2 & R_2^T A_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.27} $$

$$ A_0 = \begin{bmatrix} 1 - \frac{2j^2+2k^2}{s^2+i^2+j^2+k^2} \\ \frac{2(ij+sk)}{s^2+i^2+j^2+k^2} \\ \frac{2(ik-sj)}{s^2+i^2+j^2+k^2} \\ 0 \end{bmatrix} \quad A_1 = \begin{bmatrix} \frac{2(ij-sk)}{s^2+i^2+j^2+k^2} \\ 1 - \frac{2i^2+2k^2}{s^2+i^2+j^2+k^2} \\ \frac{2(jk+si)}{s^2+i^2+j^2+k^2} \\ 0 \end{bmatrix} \tag{3.28} $$

$$ A_2 = \begin{bmatrix} \frac{2(ik+sj)}{s^2+i^2+j^2+k^2} \\ \frac{2(jk-si)}{s^2+i^2+j^2+k^2} \\ 1 - \frac{2i^2+2j^2}{s^2+i^2+j^2+k^2} \\ 0 \end{bmatrix} \quad A_3 = \begin{bmatrix} t_{\mathrm{x}} \\ t_{\mathrm{y}} \\ t_{\mathrm{z}} \\ 1 \end{bmatrix}. \tag{3.29} $$

Where the four vectors $A_0$, $A_1$, $A_2$, and $A_3$ are introduced to simplify the notation. Note that the $A_i$ implicitly depend on the pose parameter vector $\gamma$, although this dependence is not reflected in the notation.

Transforming ${}^{\mathrm{w}}l_i(\lambda)$ into the Transgraph coordinate system gives the homogeneous equation

$$ {}^{\mathrm{tg}}l_i(\lambda,\gamma) = {}^{\mathrm{tg}}T_{\mathrm{w}}(\gamma) * {}^{\mathrm{w}}l_i(\lambda) \tag{3.30} $$

$$ = \begin{bmatrix} R_0^T A_0 x' + R_0^T A_1 y' + R_0^T A_2 z' + \lambda(R_0^T A_0(x_i - x') + R_0^T A_1(y_i - y') + R_0^T A_2(z_i - z')) + R_0^T A_3 \\ R_1^T A_0 x' + R_1^T A_1 y' + R_1^T A_2 z' + \lambda(R_1^T A_0(x_i - x') + R_1^T A_1(y_i - y') + R_1^T A_2(z_i - z')) + R_1^T A_3 \\ R_2^T A_0 x' + R_2^T A_1 y' + R_2^T A_2 z' + \lambda(R_2^T A_0(x_i - x') + R_2^T A_1(y_i - y') + R_2^T A_2(z_i - z')) + R_2^T A_3 \\ 1 \end{bmatrix}. \tag{3.31} $$

In section 3.2.1, the $C_0$ coordinate system is defined to lie at $z = 0$ in the Transgraph coordinate system, while the $C_1$ coordinate system is defined to lie at $z = z_{\mathrm{t}}$. The intersection of ${}^{\mathrm{tg}}l_i(\lambda,\gamma)$ with each coordinate plane can be found by setting the $z$ coordinate to the appropriate value and solving for $\lambda$. That is, for $C_0$

$$ R_2^T A_0 x' + R_2^T A_1 y' + R_2^T A_2 z' + \lambda_0(R_2^T A_0(x_i - x') + R_2^T A_1(y_i - y') + R_2^T A_2(z_i - z')) + R_2^T A_3 = 0 \tag{3.32} $$

$$ \lambda_0 = -\frac{R_2^T A_0 x' + R_2^T A_1 y' + R_2^T A_2 z' + R_2^T A_3}{R_2^T A_0(x_i - x') + R_2^T A_1(y_i - y') + R_2^T A_2(z_i - z')} \tag{3.33} $$

where $\lambda_0$ is the value of the parameter $\lambda$ at which ${}^{\mathrm{tg}}l_i(\lambda,\gamma)$ intersects the $C_0$ coordinate plane. Similarly, $\lambda_1$ can be found according to

$$ R_2^T A_0 x' + R_2^T A_1 y' + R_2^T A_2 z' + \lambda_1(R_2^T A_0(x_i - x') + R_2^T A_1(y_i - y') + R_2^T A_2(z_i - z')) + R_2^T A_3 = z_{\mathrm{t}} \tag{3.34} $$

$$\lambda_1 = -\frac{R_2^T A_0 x' + R_2^T A_1 y' + R_2^T A_2 z' + R_2^T A_3 - z_{\mathrm{t}}}{R_2^T A_0(x_i - x') + R_2^T A_1(y_i - y') + R_2^T A_2(z_i - z')}. \tag{3.35}$$

Substituting $\lambda_0$ and $\lambda_1$ into equation 3.31 gives the following expression for the intersection of the line segment with the two coordinate planes

$${}^{\mathrm{tg}}\boldsymbol{q}_{0,i} = \begin{bmatrix} R_0^T A_0 x' + R_0^T A_1 y' + R_0^T A_2 z' + R_0^T A_3 + \lambda_0(R_0^T A_0(x_i - x') + R_0^T A_1(y_i - y') + R_0^T A_2(z_i - z')) \\ R_1^T A_0 x' + R_1^T A_1 y' + R_1^T A_2 z' + R_1^T A_3 + \lambda_0(R_1^T A_0(x_i - x') + R_1^T A_1(y_i - y') + R_1^T A_2(z_i - z')) \\ 0 \\ 1 \end{bmatrix} \tag{3.36}$$

and

$${}^{\mathrm{tg}}\boldsymbol{q}_{1,i} = \begin{bmatrix} R_0^T A_0 x' + R_0^T A_1 y' + R_0^T A_2 z' + R_0^T A_3 + \lambda_1(R_0^T A_0(x_i - x') + R_0^T A_1(y_i - y') + R_0^T A_2(z_i - z')) \\ R_1^T A_0 x' + R_1^T A_1 y' + R_1^T A_2 z' + R_1^T A_3 + \lambda_1(R_1^T A_0(x_i - x') + R_1^T A_1(y_i - y') + R_1^T A_2(z_i - z')) \\ z_{\mathrm{t}} \\ 1 \end{bmatrix} \tag{3.37}$$

where ${}^{\mathrm{tg}}\boldsymbol{q}_{0,i}$ is the point of intersection with the $C_0$ coordinate system, and ${}^{\mathrm{tg}}\boldsymbol{q}_{1,i}$ is the point of intersection with the $C_1$ coordinate system, both expressed in Transgraph coordinates. The X and Y coordinates of ${}^{\mathrm{tg}}\boldsymbol{q}_{0,i}$ and ${}^{\mathrm{tg}}\boldsymbol{q}_{1,i}$ are substituted into the interpolation equations of section 3.3 to recover the linear attenuation associated with pixel $i$.

### 3.4.3 Computing Derivatives

As noted above, registration can be much faster if the first derivative of the synthesized pixel intensities with respect to the patient pose parameters are known. We can write these derivatives simply

$$\nabla_\gamma({}^{\mathrm{tg}}\boldsymbol{q}_{0,i}) = \begin{bmatrix} \nabla_\gamma(R_0^T A_0 x' + R_0^T A_1 y' + R_0^T A_2 z' + R_0^T A_3 + \lambda_0(R_0^T A_0(x_i - x') + R_0^T A_1(y_i - y') + R_0^T A_2(z_i - z'))) \\ \nabla_\gamma(R_1^T A_0 x' + R_1^T A_1 y' + R_1^T A_2 z' + R_1^T A_3 + \lambda_0(R_1^T A_0(x_i - x') + R_1^T A_1(y_i - y') + R_1^T A_2(z_i - z'))) \\ 0 \\ 0 \end{bmatrix} \tag{3.38}$$

$$\nabla_\gamma({}^{\mathrm{tg}}\boldsymbol{q}_{0,i}) = \begin{bmatrix} \nabla_\gamma(R_0^T A_0 x' + R_0^T A_1 y' + R_0^T A_2 z' + R_0^T A_3 + \lambda_1(R_0^T A_0(x_i - x') + R_0^T A_1(y_i - y') + R_0^T A_2(z_i - z'))) \\ \nabla_\gamma(R_1^T A_0 x' + R_1^T A_1 y' + R_1^T A_2 z' + R_1^T A_3 + \lambda_1(R_1^T A_0(x_i - x') + R_1^T A_1(y_i - y') + R_1^T A_2(z_i - z'))) \\ 0 \\ 0 \end{bmatrix} \tag{3.39}$$

With the exception of $\lambda_0$, $\lambda_1$, and the $A_i$, all of the variables in these two equations are independent of the pose parameter vector. The gradients of $A_i$ are easily computed from equations 3.28 and 3.29

$$\nabla_\gamma A_0 = \begin{bmatrix} 0 & 0 & 0 & \frac{2s(2j^2+2k^2)}{m^2} & \frac{2i(2j^2+2k^2)}{m^2} & \frac{-4jm+2j(2j^2+2k^2)}{m^2} & \frac{-4km+2k(2j^2+2k^2)}{m^2} \\ 0 & 0 & 0 & \frac{2km-4s(ij+sk)}{m^2} & \frac{2jm-4i(ij+sk)}{m^2} & \frac{2im-4j(ij+sk)}{m^2} & \frac{2sm-4k(ij+sk)}{m^2} \\ 0 & 0 & 0 & \frac{-2jm-4s(ik-sj)}{m^2} & \frac{2km-4i(ik-sj)}{m^2} & \frac{-2sm-4j(ik-sj)}{m^2} & \frac{2im-4k(ik-sj)}{m^2} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.40}$$

$$\nabla_\gamma A_1 = \begin{bmatrix} 0 & 0 & 0 & \frac{-2km-4s(ij-sk)}{m^2} & \frac{2jm-4i(ij-sk)}{m^2} & \frac{2im-4j(ij-sk)}{m^2} & \frac{-2sm-4k(ij-sk)}{m^2} \\ 0 & 0 & 0 & \frac{2s(2i^2+2k^2)}{m^2} & \frac{-4im+2i(2i^2+2k^2)}{m^2} & \frac{2j(2i^2+2k^2)}{m^2} & \frac{-4km+2k(2i^2+2k^2)}{m^2} \\ 0 & 0 & 0 & \frac{2im-4s(jk+si)}{2m} & \frac{2sm-4i(jk+si)}{2m} & \frac{2km-4j(jk+si)}{2m} & \frac{2jm-4k(jk+si)}{2m} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.41}$$

$$\nabla_\gamma A_2 = \begin{bmatrix} 0 & 0 & 0 & \frac{2jm-4s(ik+sj)}{m^2} & \frac{2km-4i(ik+sj)}{m^2} & \frac{2sm-4j(ik+sj)}{m^2} & \frac{2im-4k(ik+sj)}{m^2} \\ 0 & 0 & 0 & \frac{-2im-4s(jk-si)}{2m} & \frac{-2sm-4i(jk-si)}{2m} & \frac{2km-4j(jk-si)}{2m} & \frac{2jm-4k(jk-si)}{2m} \\ 0 & 0 & 0 & \frac{2s(2i^2+2j^2)}{m^2} & \frac{-4im+2i(2i^2+2j^2)}{m^2} & \frac{-4jm+2j(2i^2+2j^2)}{m^2} & \frac{2k(2i^2+2j^2)}{m^2} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.42}$$

$$\nabla_\gamma A_3 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{3.43}$$

It is useful (and efficient) to define the scalar intermediate variables $G_{p,q}$:

$$G_{p,q} = R_p^T[\nabla_\gamma A_q]. \tag{3.44}$$

Applying the chain rule to equations 3.33 and 3.35, we find

$$\begin{aligned} \nabla_\gamma(\lambda_0) &= -\frac{G_{2,0}x' + G_{2,1}y' + G_{2,2}z' + G_{2,3}}{R_2^T A_0(x_i - x') + R_2^T A_1(y_i - y') + R_2^T A_2(z_i - z')} \\ &\quad + \frac{\lambda_0(G_{2,0}(x_i - x') + G_{2,1}(y_i - y') + G_{2,2}(z_i - z'))}{R_2^T A_0(x_i - x') + R_2^T A_1(y_i - y') + R_2^T A_2(z_i - z')} \end{aligned} \tag{3.45}$$

$$\begin{aligned} \nabla_\gamma(\lambda_1) &= -\frac{G_{2,0}x' + G_{2,1}y' + G_{2,2}z' + G_{2,3}}{R_2^T A_0(x_i - x') + R_2^T A_1(y_i - y') + R_2^T A_2(z_i - z')} \\ &\quad + \frac{\lambda_1(G_{2,0}(x_i - x') + G_{2,1}(y_i - y') + G_{2,2}(z_i - z'))}{R_2^T A_0(x_i - x') + R_2^T A_1(y_i - y') + R_2^T A_2(z_i - z')}, \end{aligned} \tag{3.46}$$

Which we substitute into equations 3.38 and 3.39

$$\begin{aligned}
\nabla_\gamma({}^{\mathrm{tg}}\boldsymbol{q}_{0,i}) &= \begin{bmatrix} G_{0,0}x' + G_{0,1}y' + G_{0,2}z' + G_{0,3} \\ G_{1,0}x' + G_{1,1}y' + G_{1,2}z' + G_{1,3} \\ 0 \\ 0 \end{bmatrix} \\
&+ \begin{bmatrix} \lambda_0(G_{0,0}(x_i - x') + G_{0,1}(y_i - y') + G_{0,2}(z_i - z')) \\ \lambda_0(G_{1,0}(x_i - x') + G_{1,1}(y_i - y') + G_{1,2}(z_i - z')) \\ 0 \\ 0 \end{bmatrix} \\
&+ \begin{bmatrix} \nabla_\gamma(\lambda_0)(R_0^T A_0(x_i - x') + R_0^T A_1(y_i - y') + R_0^T A_2(z_i - z')) \\ \nabla_\gamma(\lambda_0)(R_1^T A_0(x_i - x') + R_1^T A_1(y_i - y') + R_1^T A_2(z_i - z')) \\ 0 \\ 0 \end{bmatrix}
\end{aligned} \tag{3.47}$$

$$\begin{aligned}
\nabla_\gamma({}^{\mathrm{tg}}\boldsymbol{q}_{1,i}) &= \begin{bmatrix} G_{0,0}x' + G_{0,1}y' + G_{0,2}z' + G_{0,3} \\ G_{1,0}x' + G_{1,1}y' + G_{1,2}z' + G_{1,3} \\ 0 \\ 0 \end{bmatrix} \\
&+ \begin{bmatrix} \lambda_1(G_{0,0}(x_i - x') + G_{0,1}(y_i - y') + G_{0,2}(z_i - z')) \\ \lambda_1(G_{1,0}(x_i - x') + G_{1,1}(y_i - y') + G_{1,2}(z_i - z')) \\ 0 \\ 0 \end{bmatrix} \\
&+ \begin{bmatrix} \nabla_\gamma(\lambda_1)(R_0^T A_0(x_i - x') + R_0^T A_1(y_i - y') + R_0^T A_2(z_i - z')) \\ \nabla_\gamma(\lambda_1)(R_1^T A_0(x_i - x') + R_1^T A_1(y_i - y') + R_1^T A_2(z_i - z')) \\ 0 \\ 0 \end{bmatrix}
\end{aligned} \tag{3.48}$$

Composing these derivatives with equations 3.16, 3.17, 3.18, and 3.19 from section 3.3.3 gives a symbolic expression for the first derivative of pixel intensity with respect to pose parameters.

## 3.5 Discussion

This chapter has presented a software-only method of accelerated volume rendering for transmission imaging. DRR generation is reduced to a sequence of 4D interpolations, with the consequence that

the time required to generate a DRR is independent of the size of the original CT dataset. The key component of this rendering method is a data structure which we call a Transgraph. The Transgraph is itself based on a data structure called a Lumigraph [20] or Light Field [36], which is part of the computer graphics field called *view-based rendering*.

The Transgraph permits efficient differentiation of DRR pixel intensity with respect to patient pose parameters. As deiscussed in chapter 2, these derivatives can be used to greatly improve convergence of our registration algorithm.

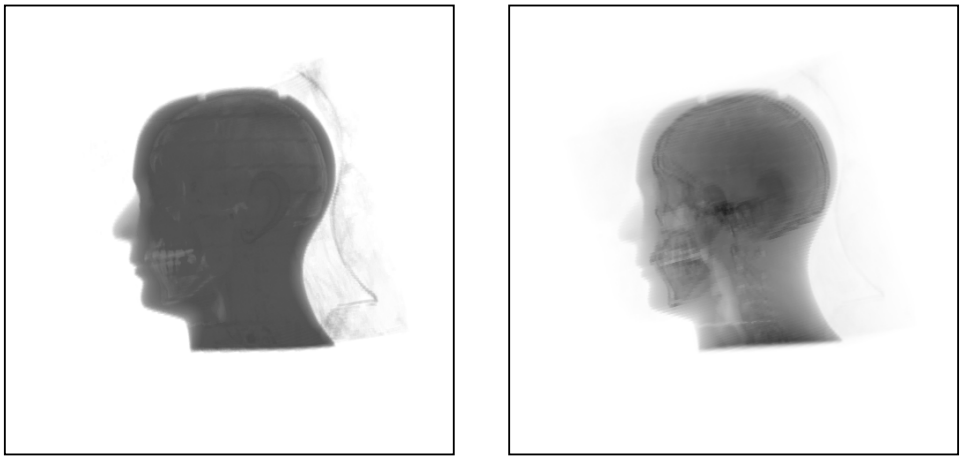
# Chapter 4

# Volume Rendering Using 2D Textures

Chapter 3 introduces a software-based algorithm for rendering transmission images. While this approach gives a significant speedup over ray-casting, image generation is still a bottleneck in the registration process. In this chapter, we introduce techniques which permit the rapid generation of DRRs using consumer-grade computer graphics hardware. In particular, we use a GeForce based card from NVIDIA Corporation to generate 512x512 images from CT volumes of up to 256x256x256 voxels at rates of roughly 14 Hz. Smaller 200x200 images, such as those required in our image-guided radiosurgery application can be computed at rates of over 40Hz. Both of these benchmarks are significantly affected by shortcomings in the current vendor supplied driver release, and we expect significant further speedups in the coming month.

To permit this computation, we have developed a new method of carrying bits between 8 bit color channels in order to perform higher precision (e.g. 16 bit) operations. This method permits emulation of hardware accelerated accumulation buffer operations on cards which do not implement a hardware accelerated accumulation buffer, and is presented in detail in chapter 5.

Our current implementation of hardware accelerated DRR generation suffers from one major drawback: the derivatives of pixel intensity with respect to patient pose parameters cannot be reliably computed. This limitation is discussed in section 4.4.1. Section 5.2.1 presents a technique which addresses this problem, but requires features which are only available in the next generation of the graphics chipset. Pending release of this new hardware (NVIDIA GeForce3, projected availability in late May, 2001) the hardware-based DRR generation algorithm is suitable for non-gradient-based optimization algorithms only.



(a) (b)

Figure 4.1: Back-to-front alpha blending results in images which look like semi-transparent volumes, as shown in (a). These images differ from transmission images (b) in that they exhibit occlusion effects. Features at the back of the object, far from the viewer, are obscured by nearby anatomy. Note how the esophagus is visible in image (b), but not in image (a). Both of these renderings are of an anthropomorphic Rando phantom. The slicing visible at the base of the neck in image (b) is an actual gap in the phantom, not a rendering artifact.

## 4.1 Background

There is already considerable work in accelerated volume rendering. Notably, Levoy [37] presents a factorization of the viewing transform which lends itself well to implementation using 2D texture mapping hardware, and extends this work with Lacroute [32]. Rezk-Salama [46] presents a hardware implementation of 3D volume rendering based on this factorization. Dachille [8] describes a volume rendering approach which combines texture hardware with host-based processing to render high quality volume images, and Eckel [12] describes a programming library which implements volume rendering using 3D texture mapping operations.

Most of the existing work, however, addresses rendering of reflectance images with opacity. Techniques such as back-to-front alpha blending are used to compose the individual texture contributions, with the result that voxels in the foreground occlude those in the background as shown in figure 4.1(a). The resulting images do not reflect the physics of transmission imaging. A simulated transmission image is shown in figure 4.1(b).

Cabral [7] describes an implementation of volume rendering using texture mapping and accumulation buffer hardware which results in realistic transmission images and is very similar to the algorithm presented here. We cannot implement Cabral's technique directly, however, since hard-

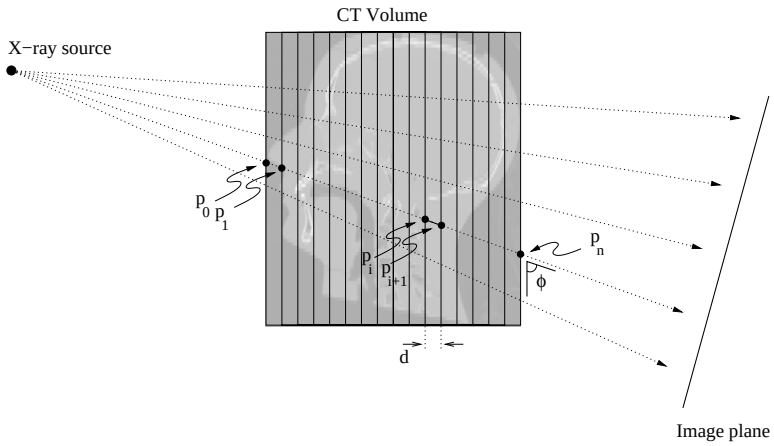



Figure 4.2: Here is a cross section of the CT, with object-aligned slices.

ware accelerated accumulation buffering is not implemented on most PC graphics cards.

This chapter introduces the concepts of hardware accelerated volume rendering using 2D texture mapping in section 4.2, and explains the role of accumulation operations in section 4.3. Section 4.4 describes relates the techniques of section 4.2 to the problem of DRR generation based on parameterized patient pose. Detailed discussion of our accumulation algorithm is deferred until chapter 5.

## 4.2 2D Texture Mapping

In order to use 2D texture mapping in volume rendering, we think of the CT volume as being made up of a collection of parallel slices. As X-rays pass from the radiation source to the imaging surface, they pass through each of these slices. We imagine the boundaries of these slices to lie exactly on the sample planes of the CT volume. If the CT sample density is high compared to the spatial frequency of the patients anatomy, we can assume linear interpolation between slices, and write the approximate log total attenuation as a ray passes through one slice

$$U_{\mathrm{ct}}\left(\boldsymbol{p}_i, \boldsymbol{p}_{i+1}\right) = \int_0^{\left\|\boldsymbol{p}_{i+1}-\boldsymbol{p}_i\right\|} \left[ \frac{s}{\left\|\boldsymbol{p}_{i+1}-\boldsymbol{p}_i\right\|} \mu_{\mathrm{ct}}\left(\boldsymbol{p}_i\right) + \left(1 - \frac{s}{\left\|\boldsymbol{p}_{i+1}-\boldsymbol{p}_i\right\|}\right) \mu_{\mathrm{ct}}\left(\boldsymbol{p}_{i+1}\right) \right] ds, \tag{4.1}$$

where $\boldsymbol{p}_i$ and $\boldsymbol{p}_{i+1}$ are 3D points on either side of the slice, as shown in figure 4.2, $\mu_{\mathrm{ct}}(x)$ is the linear attenuation coefficient corresponding to the CT value at point $x$, and $U_{\mathrm{ct}}\left(\boldsymbol{p}_i, \boldsymbol{p}_{i+1}\right)$ is the log total attenuation through the slice from point $\boldsymbol{p}_i$ to point $\boldsymbol{p}_{i+1}$ as described in section 3.1. Note that the expression $s/\left\|\boldsymbol{p}_{i+1}-\boldsymbol{p}_i\right\|$ varies from 0 to 1 over the course of the integral, so the entire expression in square brackets is simply a linear interpolation between $\mu_{\mathrm{ct}}\left(\boldsymbol{p}_i\right)$ and $\mu_{\mathrm{ct}}\left(\boldsymbol{p}_{i+1}\right)$. We

separate the two terms in the integral

$$U_{\mathrm{ct}}\left(\boldsymbol{p}_i, \boldsymbol{p}_{i+1}\right) = \int_0^{\left\|\boldsymbol{p}_{i+1}-\boldsymbol{p}_i\right\|} \frac{s}{\left\|\boldsymbol{p}_{i+1}-\boldsymbol{p}_i\right\|}\left(\mu_{\mathrm{ct}}\left(\boldsymbol{p}_i\right)-\mu_{\mathrm{ct}}\left(\boldsymbol{p}_{i+1}\right)\right) ds + \int_0^{\left\|\boldsymbol{p}_{i+1}-\boldsymbol{p}_i\right\|} \mu_{\mathrm{ct}}\left(\boldsymbol{p}_{i+1}\right) ds, \tag{4.2}$$

and integrate to solve for $U_{\mathrm{ct}}\left(\boldsymbol{p}_i, \boldsymbol{p}_{i+1}\right)$

$$\begin{aligned} U_{\mathrm{ct}}\left(\boldsymbol{p}_i, \boldsymbol{p}_{i+1}\right) &= \frac{1}{2\left\|\boldsymbol{p}_{i+1}-\boldsymbol{p}_i\right\|}\left(\mu_{\mathrm{ct}}\left(\boldsymbol{p}_i\right)-\mu_{\mathrm{ct}}\left(\boldsymbol{p}_{i+1}\right)\right) s^2 \Big|_{s=0}^{\left\|\boldsymbol{p}_{i+1}-\boldsymbol{p}_i\right\|} \\ &\quad +\mu_{\mathrm{ct}}\left(\boldsymbol{p}_{i+1}\right) s \Big|_{s=0}^{\left\|\boldsymbol{p}_{i+1}-\boldsymbol{p}_i\right\|} \qquad (4.3) \\ &= \left\|\boldsymbol{p}_{i+1}-\boldsymbol{p}_i\right\|\left(\frac{1}{2}\mu_{\mathrm{ct}}\left(\boldsymbol{p}_i\right)+\frac{1}{2}\mu_{\mathrm{ct}}\left(\boldsymbol{p}_{i+1}\right)\right). \qquad (4.4) \end{aligned}$$

The distance $\left\|\boldsymbol{p}_{i+1}-\boldsymbol{p}_i\right\|$ depends on the spacing, $d$, between adjacent slices, and the angle, $\phi$ between the ray and the surface of the slice:

$$\left\|\boldsymbol{p}_{i+1}-\boldsymbol{p}_i\right\| = \frac{d}{\cos(\phi)}. \tag{4.5}$$

Assuming uniform slice spacing, the entire line integral through the CT volume can be written as a summation,

$$\begin{aligned} U_{\mathrm{ct}}\left(\boldsymbol{p}_0, \boldsymbol{p}_n\right) &= \frac{d}{\cos(\phi)} \sum_{i=0}^{n-1}\left(\frac{1}{2}\mu_{\mathrm{ct}}\left(\boldsymbol{p}_i\right)+\frac{1}{2}\mu_{\mathrm{ct}}\left(\boldsymbol{p}_{i+1}\right)\right) \qquad (4.6) \\ &= \frac{d}{\cos(\phi)}\left(\frac{1}{2}\mu_{\mathrm{ct}}\left(\boldsymbol{p}_0\right)+\frac{1}{2}\mu_{\mathrm{ct}}\left(\boldsymbol{p}_{n-1}\right)\right)+\frac{d}{\cos(\phi)} \sum_{i=1}^{n-2}\mu_{\mathrm{ct}}(\boldsymbol{p}_i), \qquad (4.7) \end{aligned}$$

where $n$ is the total number of slices.

In order to evaluate equation 4.7, it is necessary to compute $\mu_{\mathrm{ct}}$ at the points $\boldsymbol{p}_i$, where the ray intersects each plane of the CT. Assuming that the CT volume has be preprocessed so that the voxel values reflect $\mu_{\mathrm{ct}}$, this computation is equivalent to projecting each CT plane onto the imaging surface, as shown in figure 4.3, and then interpolating in the neighborhood of each pixel location. This projection is conveniently implemented as a 2D texture mapping operation. Most 2D texture hardware implements accelerated bilinear interpolation, making computation of the values $\mu_{\mathrm{ct}}\left(\boldsymbol{p}_i\right)$ very fast.

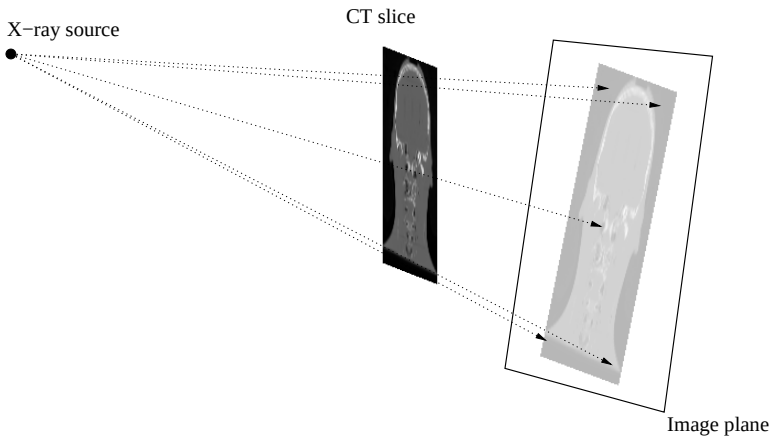



Figure 4.3: The correspondence between CT values and image pixels is easy found by texture mapping.

### 4.2.1 Projection Matrices

Our implementation of texture based volume rendering uses a graphics API called OpenGL 1.2.1 [49]. OpenGL maintains a pair of 4x4 matrices which are used to transform object coordinates before they are rendered to the screen. The matrices are called the *modelview matrix* and the *projection matrix*. When a 3D point $[{}^{\mathrm{m}}x,\ {}^{\mathrm{m}}y,\ {}^{\mathrm{m}}z]^T$ is rendered, its coordinates are projected first using the modelview matrix, and then using the projection matrix.

$$\begin{bmatrix} {}^{\mathrm{c}}x \\ {}^{\mathrm{c}}y \\ {}^{\mathrm{c}}z \\ {}^{\mathrm{c}}w \end{bmatrix} = P * M * \begin{bmatrix} {}^{\mathrm{m}}x \\ {}^{\mathrm{m}}y \\ {}^{\mathrm{m}}z \\ 1 \end{bmatrix}, \tag{4.8}$$

where $P$ is the projection matrix, and $M$ is the modelview matrix. We say that $[{}^{\mathrm{m}}x,\ {}^{\mathrm{m}}y,\ {}^{\mathrm{m}}z]^T$ is expressed in *object coordinates*, and the 3D homogeneous point $[{}^{\mathrm{c}}x,\ {}^{\mathrm{c}}y,\ {}^{\mathrm{c}}z,\ {}^{\mathrm{c}}w]^T$ is the corresponding point in *clip coordinate*s. OpenGL rendering is performed in such a way as to discard any vertices which do not satisfy the inequalities

$$-{}^{\mathrm{c}}w \leq {}^{\mathrm{c}}x \leq {}^{\mathrm{c}}w \tag{4.9}$$

$$-{}^{\mathrm{c}}w \leq {}^{\mathrm{c}}y \leq {}^{\mathrm{c}}w \tag{4.10}$$

$$-{}^{\mathrm{c}}w \leq {}^{\mathrm{c}}z \leq {}^{\mathrm{c}}w. \tag{4.11}$$

Finally, the OpenGL specification states that perspective division is performed to obtain normalized device coordinates $[^{\mathrm{d}}\mathrm{x},\ ^{\mathrm{d}}\mathrm{y}]^T$

$$^{\mathrm{d}}x = \frac{^{\mathrm{c}}x}{^{\mathrm{c}}w} \tag{4.12}$$

$$^{\mathrm{d}}y = \frac{^{\mathrm{c}}y}{^{\mathrm{c}}w}. \tag{4.13}$$

Normalized device coordinates are rendered to the screen so that the point $[-1,\ -1]^T$ maps to the lower left corner of the viewport, and the point $[1,\ 1]^T$ maps to the upper right corner of the viewport.

In order to project a CT slice onto the image plane as shown in figure 4.3, we first define a scale matrix which maps image coordinates into clip coordinates so that pixels within the boundary of the image will map to clip coordinates which satisfy equations 4.9, 4.10, and 4.11

$$S = \begin{bmatrix} \frac{2}{x_{\max}-x_{\min}} & 0 & 0 & \frac{-(x_{\max}+x_{\min})}{(x_{\max}-x_{\min})} \\ 0 & \frac{2}{y_{\max}-y_{\min}} & 0 & \frac{-(y_{\max}+y_{\min})}{y_{\max}-y_{\min}} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{4.14}$$

where $x_{\min}$ and $x_{\max}$ are the minimum and maximum image $x$ coordinates. Also, $y_{\min}$ and $y_{\max}$ are the minimum and maximum image $y$ coordinates.

Next, we write a 3x4 matrix describing the projection from 3D coordinates to 2D coordinates in the image plane. For the work in this thesis, this matrix generally has the form of pinhole camera projection

$$P' = \begin{bmatrix} fk_{\mathrm{x}} & 0 & x_0 & 0 \\ 0 & fk_{\mathrm{y}} & y_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \tag{4.15}$$

where $P'$ is the camera projection matrix (different from the OpenGL projection matrix), and the constants $f$, $k_x$, $k_y$, $x_0$, and $y_0$ are the camera *intrinsic parameters* [14]. We define a minimum and maximum for the 3D $z$ coordinates which will be projected into the image, and add a third row to the camera projection matrix so that this range of $z$ values will project to normalized device coordinates between $-1$ and 1,

$$P'' = \begin{bmatrix} fk_x & 0 & x_0 & 0 \\ 0 & fk_y & y_0 & 0 \\ 0 & 0 & \frac{z_{\min}+z_{\max}}{z_{\min}-z_{\max}} & \frac{-2z_{\min}z_{\max}}{z_{\min}-z_{\max}} \\ 0 & 0 & 1 & 0 \end{bmatrix}, \tag{4.16}$$

where $z_{\min}$ and $z_{\max}$ are the minimum and maximum expected $z$ coordinates.

Finally, we set the OpenGL projection matrix to the product of $S$ and $P''$,

$$P = SP''. \tag{4.17}$$

This is easily done using the OpenGL commands *glLoadIdentity()* and *glMultMatrixd()*.

Generally, the CT slices are specified in a coordinate system which is different from that of the pinhole camera. We set the OpenGL modelview matrix to a 4x4 transformation matrix which takes coordinates from the CT coordinate system to the camera coordinate system. The slices are then rendered by specifying rectangular polygons in CT coordinates and texture mapping them with the appropriate texture.

```
glBegin(GL_QUADS);
glTexCoord2d(0.0, 0.0);
glVertex3d(coord0.x(), coord0.y(), coord0.z());
glTexCoord2d(0.0, 1.0);
glVertex3d(coord1.x(), coord1.y(), coord1.z());
glTexCoord2d(1.0, 1.0);
glVertex3d(coord2.x(), coord2.y(), coord2.z());
glTexCoord2d(1.0, 0.0);
glVertex3d(coord3.x(), coord3.y(), coord3.z());
glEnd();
```

where the variables *coord0*, *coord1*, *coord2*, and *coord3* are expressed in the CT coordinate system, and the appropriate texture has been previously bound using *glBindTexture()*.

Texture mapping hardware gives us an efficient way to compute the terms of the summation in equation 4.7, however the actual summation remains a problem. PC graphics hardware typically represents image pixels with a maximum of only 8 bits per channel. These 8-bit values quickly overflow with the addition of subsequent slices. The process of successively rendering textures to the framebuffer and adding their values is known as *accumulation*, and is discussed in the next section.

## 4.3 Accumulation

In order to overcome the precision limits of the framebuffer, some high-end graphics hardware provides an *accumulation buffer*. The accumulation buffer is a separate region of memory in which pixels are represented with a higher resolution than they are in the frame buffer. After an image

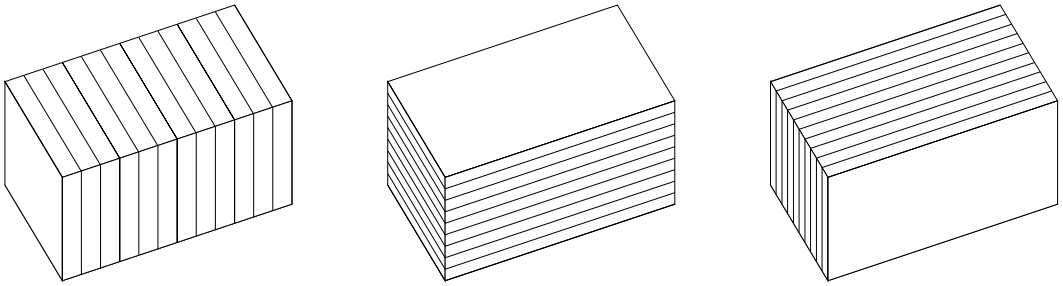


Figure 4.4: Three stacks of textures are generated by slicing the CT along each of the three major axes. The texture stacks used in this research have between 100 and 256 slices.

is rendered to the frame buffer, it can be copied to the accumulation buffer, where it may add to, subtract from, or replace the existing accumulation buffer contents.

If a hardware accelerated accumulation buffer is available, the sum in equation 4.7 can be computed by rendering each texture in turn, and then adding it to the accumulation buffer. When each CT slice has been rendered and added, the accumulation buffer contents are scaled appropriately and copied back to the frame buffer.

Unfortunately, nearly all PC graphics hardware does not support hardware accelerated accumulation buffering. For these cards, all accumulation operations are done using the host processor. This makes the accumulation buffer too slow for use in interactive rendering. Even with hardware accelerated accumulation buffering, the cost of copying data from the frame buffer to the accumulation buffer is significant, and can noticeably increase the time required to synthesize an image.

Chapter 5 describes a technique for emulating a 16 bit monochrome accumulation buffer by using the three 8 bit color channels in concert. This technique relies on the NV_register_combiners OpenGL extension and currently enables accumulation operations to run in less than $1/40^{\text{th}}$ of the time required for equivalent operations using the vendor supplied library calls. We expect to see further performance gains as the available hardware drivers mature. For monochrome applications, we anticipate that our emulated accumulation buffer will soon be faster than the equivalent operations on a card which natively supports hardware accelerated accumulation buffering.

## 4.4 Generating DRRs Using Texture Hardware

In order to generate DRRs using texture hardware, data from the CT volume is first used to create three sets of 2D textures. Each set represents a slicing of the CT volume along one axis, as illustrated in figure 4.4. In this work, we set the slice spacing equal to the voxel spacing in the CT volume, and use each plane of voxels to define one 2D texture. Along with each slice, we record the location of its four corners in CT coordinates.

To render an image, the pose parameter vector $\gamma$ is used to compute a 4x4 transformation matrix which takes coordinates in the CT coordinate system to coordinates in the 3D coordinate system of the imager. This transformation, and the imager calibration parameters from chapter 6, are used to set the OpenGL projection and modelview matrices as described in section 4.2.1. Each texture is rendered in turn, and the accumulated image is copied from the frame buffer using *glReadPixels()*.

### 4.4.1 Computing Derivatives

Modern graphics hardware is very fast. The NVIDIA GeForce2 Ultra based hardware used in this research is capable of sustained fill rates of over 250 million pixels per second. Consequently it is reasonable to compute pixel intensity gradients by the method of finite differences. In order to compute the first derivative of pixel intensity with respect to the $i^{\text{th}}$ element of $\gamma$, we proceed as follows:

1. First, a baseline image is rendered, corresponding to the 2D pose parameter vector $\gamma$. This image is read from the frame buffer into an array in host memory.

2. A new parameter vector, $\gamma'$ is defined. The elements of $\gamma'$ are identical to those of $\gamma$, except that the $i^{\text{th}}$ element of $\gamma'$ is incremented by a small amount, $\epsilon$.

3. Another image is generated, corresponding to the parameter vector $\gamma'$. This image is also read from the frame buffer.

4. The baseline image from step 1 is subtracted from the newly read image, and the difference divided by $\epsilon$. The resulting array contains an approximation of the first derivative of pixel intensity with respect to the $i^{\text{th}}$ element of $\gamma$.

Steps 2–4 are repeated for each element $\gamma$.

At the time of publication our accumulation technique suffers from one major drawback: it interferes with texture interpolation during rendering. The rendered images are sufficiently realistic to permit accurate registration, but the lack of interpolation interferes with gradient computation. Consequently, we currently use hardware accelerated volume rendering in conjunction with non-gradient based registration methods. We anticipate that the method described in section 5.2.1 will resolve this problem as soon as GeForce 3 based hardware becomes available. The details of this drawback are described in section 5.2.1.

# Chapter 5

# Hardware Accelerated Accumulation

Chapter 4 presents an algorithm for rendering DRRs using 2D texture hardware. This algorithm depends heavily upon a set of graphics features known as *accumulation buffering*. Unfortunately, hardware accelerated accumulation buffer operations are extremely rare among PC graphics cards. In nearly all currently available hardware, accumulation buffer operations are performed using the host CPU over the system bus, resulting in very slow performance.

This chapter describes how to emulate hardware accelerated accumulation operations using recently released graphics hardware from NVIDIA. This implementation avoids much of the data transfer associated with traditional accumulation buffering, resulting in 16-bit accumulation at very nearly the maximum texture fill-rate of the card.[1]

This technique relies on the NV_register_combiners OpenGL extension, and currently provides more than a 40 times speedup over software accumulation buffering. We expect to see further performance gains as the vendor supplied driver matures. For monochrome applications, we anticipate that our emulated accumulation buffer will soon be faster than the equivalent operations on a card which natively supports hardware accelerated accumulation buffering.

In this chapter, section 5.1 introduces accumulation buffer operations and describes our emulation strategy, while section 5.2 provides details of the implementation.

Note that NV_register_combiners is a *vendor specific* extension to OpenGL. In other words, the techniques described here are not portable to graphics hardware from other vendors.

[1]The current implementation runs considerably slower than this, due to slow glCopyTexSubImage2d() performance in version 0.96 of the vendor supplied driver. We expect this performance to improve dramatically in subsequent driver releases.

## 5.1 Accumulation Buffer Concept

The OpenGL 1.2.1 specification defines the *glAccum()* function, which copies pixels between the framebuffer and a separate *accumulation buffer*. [49] Pixels in the accumulation buffer are typically represented with higher precision than pixels in the framebuffer. For example, accumulation buffer pixels may have 16-bit red, green, and blue values, while framebuffer pixels may have only 8 bits per channel. A 16-bit accumulation buffer can be used to sum or average as many as 257 8-bit images with no loss of precision.

Unfortunately, hardware accelerated accumulation buffer operations are frequently not implemented by graphics vendors and driver writers. Most PC graphics card/driver combinations implement a host-based accumulation buffer which is too slow to be useful in our application.

The GeForce family of graphics processors introduce the NV_register_combiners OpenGL extension [3]. The following sections describe how this extension allows single-channel data to be accumulated directly to the framebuffer. Single channel accumulation can be generalized to RGB data by accumulating each channel independently, and finally combining the result from each channel.

The NV_register_combiners extension is important in this scheme since it provides a convenient way of transferring bits from one channel to another. This allows the three color channels to be used in concert, representing numbers with more bits than the native framebuffer precision. The rest of this section describes how the color channels can be used together.

### 5.1.1 Channel-Distributed Representation

The GeForce frame buffer uses an 8-bit fixed point format, which represents numbers uniformly distributed in the range [0, 1.0). For example, a byte containing the binary number $11110000$ (decimal $240$) corresponds to the fixed point number $\frac{240}{256} = 0.9735$. The maximum representable number is $\frac{255}{256} = 0.99609375$, and the minimum representable number is $\frac{0}{255} = 0.0$. In order to emphasize this representation, we will frequently write 8-bit fixed point numbers as fractions with the numerator and denominator expressed in hexadecimal notation. Continuing the example above, we write the fixed point number $0.9735$ using the fraction $\frac{\mathrm{0xf0}}{\mathrm{0xff}}$.

Suppose that a single-channel, 8-bit image is split into two 4-bit images and rendered to the Blue and Green channels with the four low-order bits going into the low-order nibble of the Blue channel, and the four high-order bits going into the low-order nibble of the Green channel as shown in figure 5.1. Subsequent images can be similarly split and simply added to the frame buffer. Since the largest possible four-bit number is $\frac{\mathrm{0x0f}}{\mathrm{0xff}} = 0.05859375$, a total of 17 images can be accumulated in this way without risk of overflowing the Blue and Green channels. After the $17^{\mathrm{th}}$ image has been rendered, the four high-order bits of the Green channel can be carried into the low-order bits of the

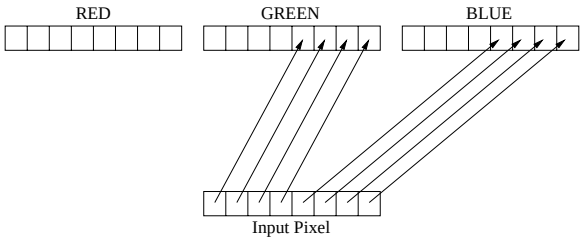



Figure 5.1: In one accumulation scheme, the four high-order bits of each pixel are rendered to the Green channel while the four low-order bits are rendered to the Blue channel.

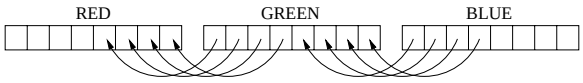



Figure 5.2: In the accumulation scheme of figure 5.1, a carry operation clears the four high-order bits of the Green channel, adding them to the low-order bits of the Red channel, and then clears the four high-order bits of the Blue channel, adding them to the low-order bits of the Green channel.

Red channel, and the four high-order bits of the Blue channel can be carried into (and added to) the four low-order bits of the Green channel. This carry operation is illustrated in figure 5.2. Once the carry operation has been completed, another 15 images can be accumulated before the Green channel is in danger of overflow and a second carry is required.

Under this scheme, a total of 257 8-bit images can be accumulated before the Red channel is in danger of overflowing. The value of the accumulated sum is represented by the contents of all three color channels. We call this *channel-distributed representation*.

The key idea is to distribute the bits of an image between framebuffer channels. Exactly how the bits should be distributed depends on the application. Figure 5.3 shows two other accumulation schemes. One of these schemes distributes 2 bits to the Red channel, 3 bits to the Green channel, and 3 bits to the Blue channel. Splitting the bits in this way decreases the effective size of the accumulation buffer to 14 bits, but allows 36 images to be accumulated before the first carry. Still another scheme distributes 3 bits to the Green channel, and the remaining 5 bits to the Blue channel. This increases the effective size of the accumulation buffer to 20 bits, but requires carries after the 8th image, and after every subsequent 7th image. It is good to minimize the number of carries required, since although the carry operation is hardware accelerated, it still has non-zero computational cost. In particular, the carry operation involves a call to *glCopyTexSubImage2d()*. At the time of publication, glCopyTexSubImage2d() requires nearly 10 ms for a 512x512 image on our test machine, possibly due to a data transfer over the AGP bus. It is our understanding that pending releases of the NVIDIA drivers will significantly speed up this call.

For the rest of this report, we discuss the 0-4-4 distribution scheme of figure 5.1 only. Most of

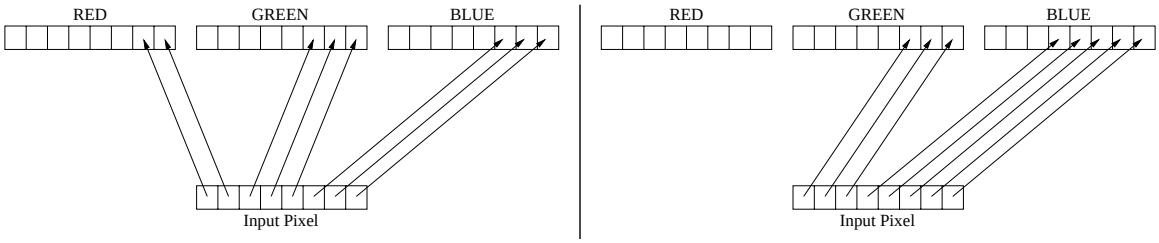



Figure 5.3: Other accumulator bit assignments are useful as well, providing either greater precision, or less frequent carry operations.

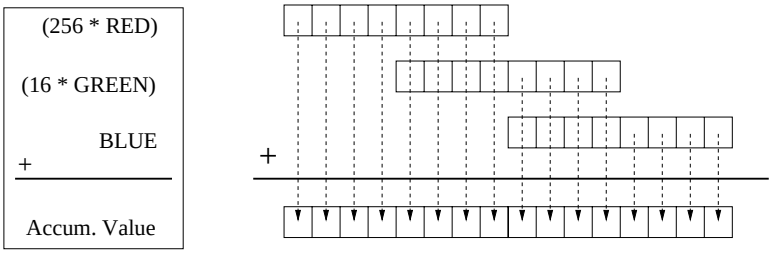



Figure 5.4: The accumulated value from figures 5.1 and 5.2 depends on all three Channels. The 8-bit Red, Green, and Blue channels are used in concert to represent a 16-bit accumulator.

the discussion will generalize to other distribution schemes.

### 5.1.2 Interpreting Channel-Distributed Numbers

After a number of images have been accumulated in channel-distributed representation, and before the result is displayed on the screen, the information in the three color channels must be combined to recover a single channel result. This can be done by appropriately scaling the Red, Green, and Blue channels of the frame buffer, and then adding the scaled values. The accumulated sum can be recovered by computing the sum $x = 256 * \mathrm{Red} + 16 * \mathrm{Green} + \mathrm{Blue}$, as shown in figure 5.4. Often it is useful to scale this recombined sum so as not to overflow the 8-bit frame buffer. For example, if $N$ images have been accumulated, the average image is simply $x/N$. Our current implementation permits scaling only by powers of two, however we anticipate that more general scaling will be straightforward to implement.

## 5.2 Accumulation Buffer Implementation Using Register Combiners

The NV_register_combiners extension bypasses the normal OpenGL texture pipeline, replacing it with a series of configurable texture processing units called *general register combiners*, followed by

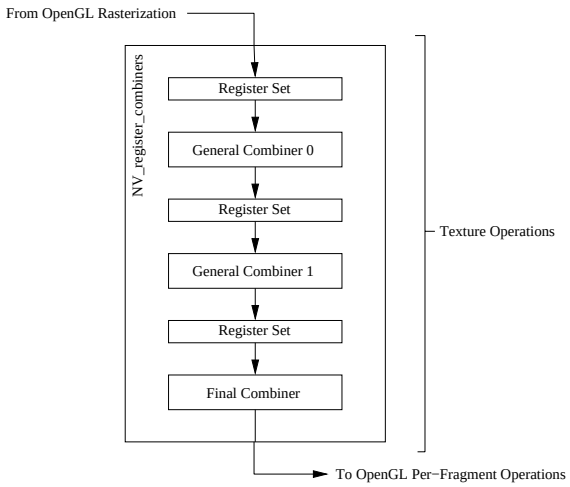



Figure 5.5: The NV_register_combiners extension replaces the standard OpenGL texture pipeline. Implementations provide at least two general combiners.

a single *final combiner*. The general structure of the register combiners pipeline is shown in figure 5.5.

Input textures, fragment primary and secondary color, fog color, and several other values are made available to each combiner through a set of registers. The general combiner takes four inputs, which can be drawn from any of the available registers, and computes up to three output values, which are written back to the register set. The actual computations performed by the general register combiner are controlled through the NV_register_combiners API, and can include summations, multiplications, and dot-product operations. Each general combiner is applied in turn, and modifies the set of register values available to the next combiner. A schematic representation of a general combiner is shown in figure 5.6.

The final combiner takes up to seven inputs, which are also drawn from the register set. The output of the final combiner is an RGBA texture, which is sent for standard OpenGL per-fragment processing. The final combiner is illustrated in figure 5.7.

For more information on the structure and programming of the NV_register_combiners interface, please refer to the NVIDIA website, http://www.nvidia.com.

### 5.2.1 Rendering

When a series of images is to be rendered to the emulated accumulation buffer, these images must be modified so that their Red, Green, and Blue colors match the bit patterns described in section 5.1.1. That is, the rendered images must reach the framebuffer in channel-distributed representation. This chapter describes how to configure the hardware so that the rendered images are converted to

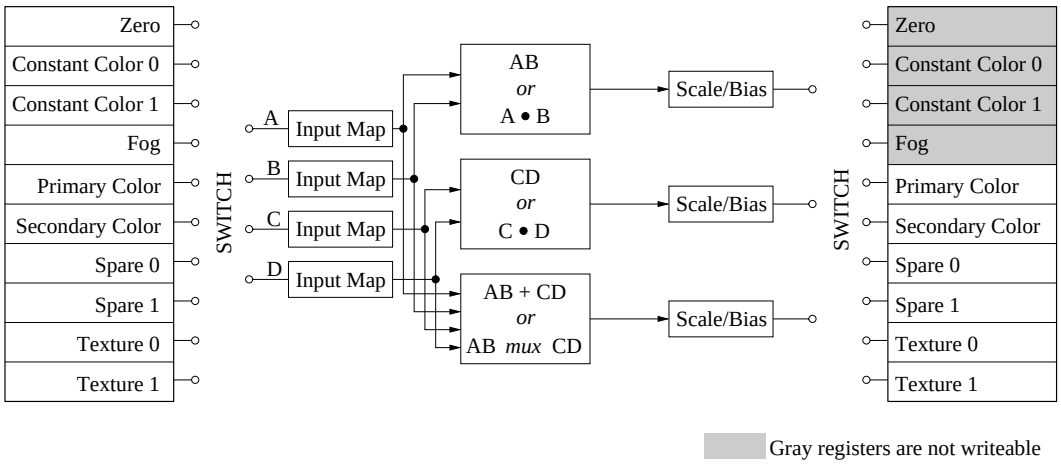



Figure 5.6: General combiner stages can perform flexible operations on both RGB and Alpha values. RGB and Alpha processing are controlled independently.

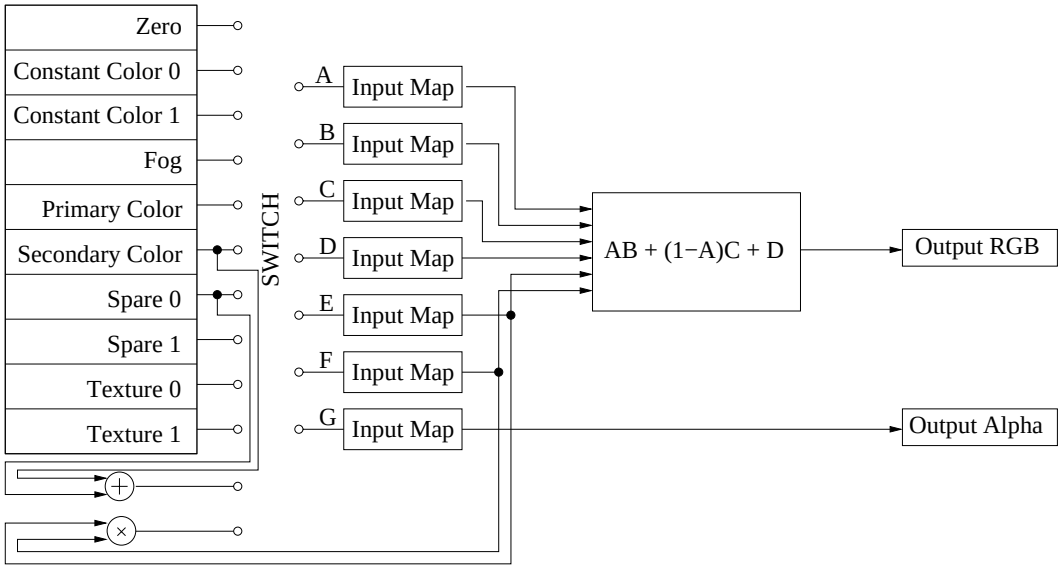



Figure 5.7: The final combiner stage performs a fixed computation, and sends the output value to the standard OpenGL per-fragment operations.

channel-distributed representation automatically during rendering. For the rest of this chapter, we assume that the rendered images consist of 8-bit intensity data. When full RGB data must be rendered, the Red, Green, and Blue channels must be accumulated independently, and recombined after all accumulation operations are completed.

Graphics cards based on the GeForce 3 chipset implement 8 general combiner stages. This provides enough flexibility to perform normal texturing operations while reserving 3 general combiner stages for the job of converting the rendered image to channel-distributed representation. This conversion breaks down into two distinct tasks: selecting the four high-order bits of the pixel intensity, right-shifting them, and rendering to the Green channel; and selecting the four low-order bits of the pixel intensity for rendering to the Blue channel.

In general, the NV_register_combiners extension does not support bit selection, however the current hardware implementation uses a 9-bit signed fixed point representation which can be exploited to perform these operations.[2] Simply multiplying the intensity value by $1/16$ very nearly accomplishes the high-order bit selection and right-shifting simultaneously. Unfortunately, the fixed point register values are rounded to the nearest representable value, which introduces a rounding error. We defeat the rounding by subtracting $1/32$ from the pixel intensity before multiplying by $1/16$. Selecting the low-order bits can be done by left shifting this result 4 bits (multiplying by 16), and subtracting the left shifted bits from the original intensity value. These operations require three general combiner stages, which are configured as follows:

1. In the first of the three general combiner stages, we need to subtract a bias value of $1/32$ ($0\mathrm{x}08/0\mathrm{xff}$, corresponding to an unsigned integer value of 8) from the pixel intensity value, and multiply the sum by $1/16$. We observe that this is equivalent to computing $\frac{1}{16}I - \frac{1}{16}\left(\frac{1}{32}\right) = \frac{0\mathrm{x}10}{0\mathrm{xff}}I - \frac{0\mathrm{x}10}{0\mathrm{xff}}\left(\frac{0\mathrm{x}08}{0\mathrm{xff}}\right)$, and compute this quantity using the combiner's sum output.

2. The result of step 1 is passed to the second of the three general combiner stages through a register. This truncates the four low-order bits, which have been shifted below the resolution of the 9-bit signed fixed point representation.

3. The second combiner stages is configured to multiply the truncated intensity value by 8. This rescaled value is passed to the third combiner through a second register.

4. The intensity value from step 3 must be multiplied by a further factor of two before being subtracted from the original intensity value. Rather than use another general combiner stage for

[2]The NV_register_combiners documentation explicitly states that the 9-bit signed fixed point representation is not part of the extension specification. In other words, future implementations of NV_register_combiners may not use this representation, and this approach may not be portable to future versions of the card.

this operation, the original intensity value is multiplied by $1/2$, the subtraction is performed, and the result is scaled by two on output from the general combiner stage. There is no loss of precision through this operation, since the arithmetic units of the combiner maintain several bits of precision below the 9-bit threshold. This is an important distinction: strict 9-bit truncation occurs when passing values through the registers, while arithmetic operations maintain several additional bits of precision.

5. The final combiner stage is configured to multiply the truncated, shifted value from step 1 by (0, 1, 0), to multiply the masked value from step 4 by (0, 0, 1), and to pass the sum of these two products to the frame buffer.

This configuration is illustrated in figure 5.8.

At the time of publication, GeForce 3 cards are not yet available for testing. Consequently the algorithm described in this section has not been fully tested. The following section presents methods for directly specifying channel-distributed colors. These methods work for chipsets prior to the GeForce 3, however their use requires that bilinear color interpolation be disabled during rendering of distributed colors. Section 5.2.2 describes this drawback more fully.

**Direct specification of channel-distributed colors.**

When GeForce 3 hardware is not available, channel distributed colors can be directly specified using the OpenGL API.. Finding the correct values is especially easy because the routines which set color silently convert unsigned integers to the corresponding fixed point values. For example, if the primary color is 0.203125, which corresponds to an 8-bit unsigned value of 52, the appropriate channel-distributed color can be set using *glColor3i()*.

```
glColor3i(0, (52 & 0x00f0) >> 4, 52 & 0x000f);
```

If desired, lookup tables can be used to simplify color assignment.

```
unsigned int i;
GLuint redMap044[255], greenMap044[255], blueMap044[255];
for(i = 0; i < 256; ++i) {
  redMap044[i] = 0;
  greenMap044[i] = (i & 0x00f0) >> 4;
  blueMap044[i] = i & 0x000f;
}
[...]
```

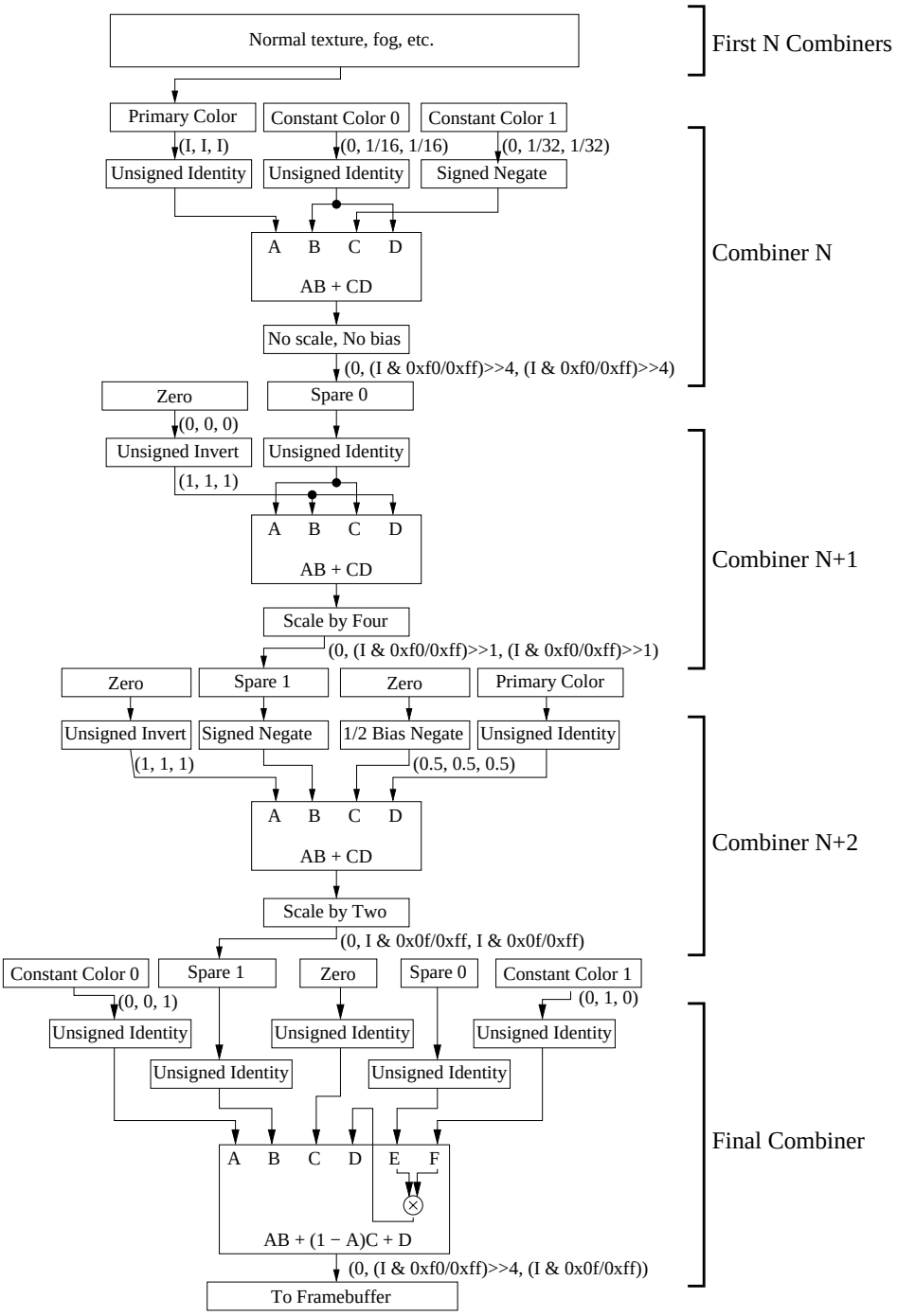



Figure 5.8: The NV_register_combiners extension can be used to render channel-distributed images. Note that the use of more than two general combiner stages means this configuration is not appropriate for GeForce 2 and lower. The register variables *Constant Color 0* and *Constant Color 1* take on different values at different stages of the pipeline. This is supported in the NV_register_combiners2 extension, which is available on GeForce 3 cards.

```
glShadeModel(GL_FLAT);
glColor3i(redMap044[52], greenMap044[52],
          blueMap044[52]);
```

The call to *glShadeModel()* is explained in section 5.2.2.

Similarly, RGB textures can be specified by explicitly mapping the input intensities to distributed RGB colors before calling *glTexImage*()*, but it is often more convenient to use the EXT_paletted_textures extension. [3] Under this extension, single-channel textures are used to index into colormap. Values from the colormap are then passed to the texture pipeline. This extension is accessed through the *glColorTableEXT()* API call.

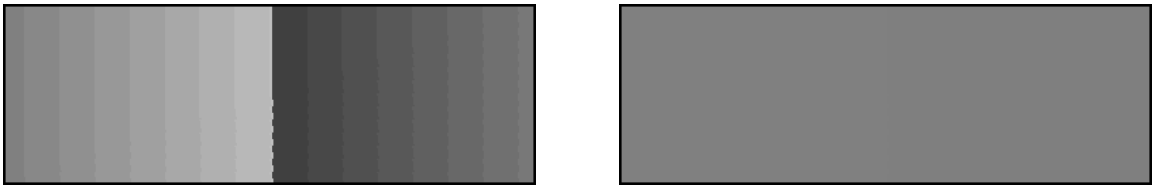
```
unsigned int i;
GLubyte paletteRGBA[255 * 4];
for(i = 0; i < 4 * 256; i += 4) {
  paletteRGBA[i] = 0;
  paletteRGBA[i + 1] = (i & 0x00f0) >> 4;
  paletteRGBA[i + 2] = i & 0x000f;
  paletteRGBA[i + 3] = 1;
}
glColorTableEXT(GL_TEXTURE_2D, GL_RGBA, 256, ...,
                (GLvoid*)paletteRGBA);
[...]
glBindTexture(GL_TEXTURE_2D, ...);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
glTexImage2D(..., GL_COLOR_INDEX_8_EXT, ...);
```

If more than one texture must be specified, the EXT_shared_texture_palette extension gives even more convenience and savings in texture memory by allowing multiple textures to use the same color table.

The calls to *glTexParameteri()* in this example are important. Setting GL_TEXTURE_MIN_FILTER or GL_TEXTURE_MIN_FILTER to GL_LINEAR can lead to interpolation artifacts as described in section 5.2.2.

### 5.2.2 A Note About Interpolation

Explicitly setting the drawing color to channel-distributed representation using the direct specification methods of section 5.2.1 has one major drawback. Unless the shade model is explicitly set



(a) (b)

Figure 5.9: (a) Explicitly setting a channel-distributed rendering color can lead to color interpolation artifacts as described in section 5.2.2, and as shown in this image of a single quadrilateral. The color of the quadrilateral should vary smoothly and almost imperceptibly from left to right as described in the text. (b) The same image, this time rendered without explicitly setting a channel-distributed rendering color, and without interpolation artifacts.

to GL_FLAT using the *glShadeModel()* command and bilinear texture interpolation is disabled, OpenGL linearly interpolates colors between vertices and texture pixels. If a color is converted to channel-distributed representation before this linear interpolation takes place, the high- and low-order nibbles will be interpolated independently, introducing artifacts to the image. These artifacts are illustrated in figure 5.9(a), which shows a rendering of a single quadrilateral. The two vertices at the left side of the image have a primary color one grey-level brighter than the two vertices at the right side of the image. Figure 5.9(b) shows the same quadrilateral without interpolation artifacts.

These interpolation artifacts mean that bilinear texture interpolation cannot be enabled on pre-GeForce 3 cards during DRR generation. Lack of bilinear interpolation during 2D texture mapping results in quantization artifacts, as shown in figure 5.10. More importantly, though, this lack interferes with the forward differences computation of section 4.4.1, and currently precludes the use of (GeForce2) hardware generated DRRs in gradient-based optimization.

### 5.2.3 Carrying

#### Accessing the texture hardware

The carry operation differ from the rendering operation described above in that the rendered colors depend on what is already in the framebuffer. That is, in order to carry the high-order bits of the Green channel into the Red channel, it is necessary to first inspect the Green channel and determine what needs carrying. Since the inspection and carry operations are implemented using the NV_register_combiners extension, this means that the framebuffer contents must be made accessible at the beginning of the texture pipeline.

The framebuffer contents are made available to the texture pipeline. using the *glCopyTexSubImage2D()* function. This function copies the specified region from the framebuffer into texture mem-

(a) (b)

Figure 5.10: Lack of bilinear interpolation using current GeForce hardware leads to quantization artifacts, which are particularly visible in the forehead of the skull in (a). A rendering with bilinear interpolation (b) does not show these artifacts.

ory.[3] This texture can then be manipulated using the texture hardware, and rendered back to the framebuffer. In order to make the rendered texture overlay the original exactly, it is important to choose the texture coordinates with care. For the remainder of this report we make the simplifying assumption that the framebuffer is 512x512 pixels in size. Other sizes can be accommodated by carefully choosing texture coordinates and texture sizes.

### Masking high-order bits using *glLogicOp()*

Once the framebuffer contents have be copied into a texture object, the high-order bits of the Green and Blue channels must be cleared. The most straightforward way to do this is using *glLogicOp()*.

```
glMatrixMode(GL_MODELVIEW);
glPushMatrix();
glLoadIdentity();
glMatrixMode(GL_PROJECTION);
glPushMatrix();
glLoadIdentity();
glColor3i(0x00ff, 0x000f, 0x000f);
glEnable(GL_COLOR_LOGIC_OP);
glLogicOp(GL_AND);
```

[3] At the time of publication, the implementation of *glCopyTexSubImage2D()* is quite slow, requiring approximagely 13ms per megabyte of image data on our 550MHz test machine (2x AGP bus, NVIDIA drivers version 0.96). It is our understanding that pending driver releases will significantly speed up texture copying performance.

```
glBegin(GL_QUADS);
glVertex3f(-1.0, -1.0, 0.0);
glVertex3f(-1.0, 1.0, 0.0);
glVertex3f(1.0, 1.0, 0.0);
glVertex3f(1.0, -1.0, 0.0);
glEnd();
glDisable(GL_COLOR_LOGIC_OP);
glMatrixMode(GL_MODELVIEW);
glPopMatrix();
glMatrixMode(GL_PROJECTION);
glPopMatrix();
```

Unfortunately, at the time of publication, hardware accelerated logical operations do not appear to be supported on GeForce hardware.

#### Masking high-order bits without using *glLogicOp()*

When *glLogicOp()* is too slow to be useful, high-order bits can be be zeroed by carefully exploiting the fixed point representation used in the GeForce texture hardware, and by using the EXT_blend_subtract OpenGL extension. The approach is to render the texture from section 5.2.3 into the framebuffer so that it exactly overlays the original data. During the rendering pass, the texture is processed so that the low-order bits are set to zero. The remaining high-order bits are subtracted from the framebuffer. The texture processing requires two general register combiner stages, and proceeds as follows:

1. The first general combiner is configured to multiply Red, Green, and Blue by 0, $1/16$, and $1/16$ respectively. This effectively zeros the Red channel, while right shifting the Green and Blue channels by 4 bits. As in section 5.2.1 a bias term of $1/32$ is needed to eliminate rounding errors.

2. The result of step 1 is passed to the second general combiner through a register. This truncates the four low-order bits, which have been shifted below the resolution of the 9-bit signed fixed point representation.[4]

3. The second general combiner and final combiner are configured to scale Green and Blue by a factor of 16. This shifts the four high-order bits back to their original places, so that they can be subtracted from the framebuffer.

[4] Please see footnote 2 on page 65.

This process is illustrated in figure 5.11.

**Transferring data between channels**

Once the high-order bits of the framebuffer have been zeroed, the carry is completed by adding the carry bits to the appropriate channels. This is done by again rendering the texture from section 5.2.3 into the framebuffer so that it exactly overlays the original data. As before, the texture values are right shifted four bits, but this time the NV_register_combiners dot-product operation is used to transfer the data between the color channels.

1. The the first general combiner is configured to multiply Red, Green, and Blue by 0, $1/16$, and $1/16$ respectively. As described in section 5.2.1, the green and blue channels are biased by $-1/32$ prior to the multiplication in order to avoid rounding errors. This effectively zeros the Red channel, while right shifting the Green and Blue channels by 4 bits.

2. The result of step 1 is passed to the second general combiner through a register. This truncates the four low-order bits, which have been shifted below the resolution of the 9-bit signed fixed point representation.[5]

3. The second general combiner is configured to compute the dot product of the scaled texture with the {Red, Green, Blue} triple {0.0, 1.0, 0.0}. This combiner is further configured to simultaneously compute the dot product of the scaled texture with {0.0, 0.0, 1.0}. The first dot product effectively distributes the scaled Green component across all three channels. The second dot product distributes the Blue component across all three channels.

4. The final combiner stage is configured to multiply the two dot-products from step 3 by the RGB triples {1.0, 0.0, 0.0} and {0.0, 1.0, 0.0} respectively. This zeros the unwanted channels in each dot-product, leaving only the carry bits. The final combiner stage is further configured to add these two results into one RGB triple.

5. The resulting value is added to the frame buffer *glBlend()*.

This process is illustrated in figure 5.12.

### 5.2.4 Recovering Accumulated Data

Once the accumulation has been carried out in channel-distributed representation, the result must be converted back into RGB format. Since the different channels of the distributed representation

[5] Please see footnote 2 on page 65.

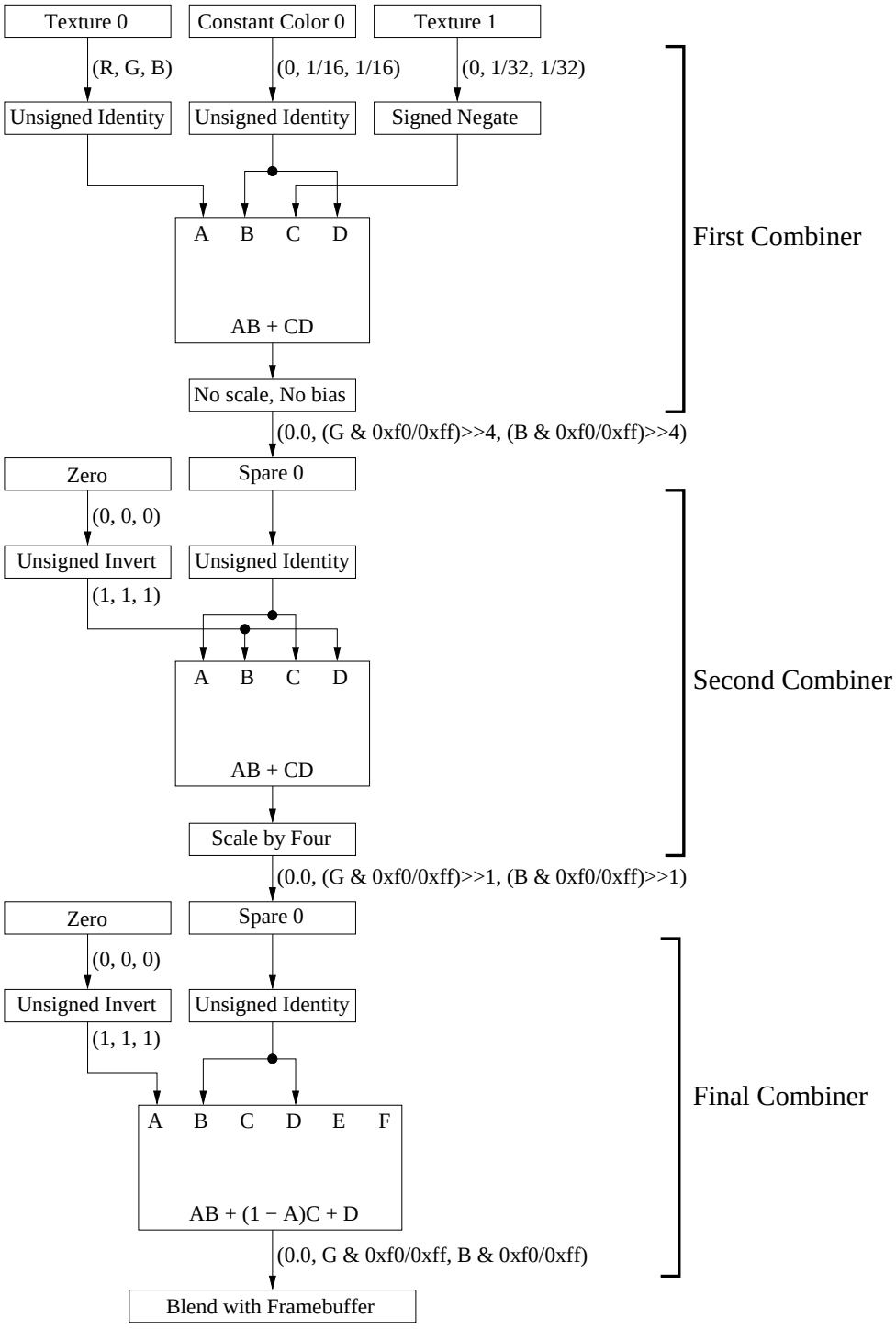



Figure 5.11: The high-order bits of the framebuffer can be selected by exploiting NVIDIA's fixed point texture representation.

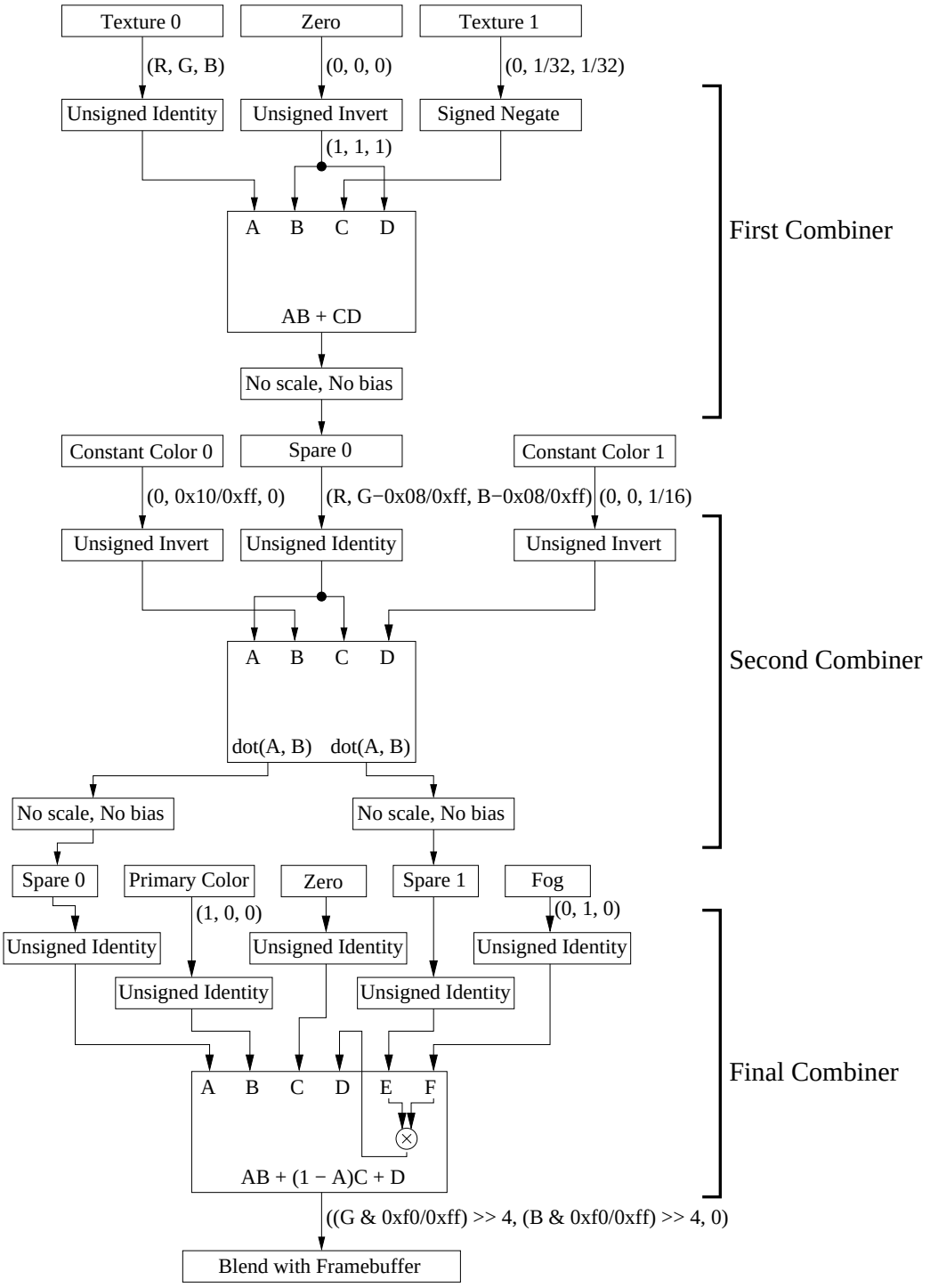



Figure 5.12: Bits are carried from one channel to another using the dot-product operation. The two dot-product outputs of of the second combiner are as follows: Spare0 = ((Green & 0xf0/0xff) >> 4, (Green & 0xf0/0xff) >> 4, (Green & 0xf0/0xff) >> 4); and Spare1 = ((Blue & 0xf0/0xff) >> 4, (Blue & 0xf0/0xff) >> 4, (Blue & 0xf0/0xff) >> 4).

correspond to specific bit ranges in the accumulated value, the recovery amounts to scaling each channel by a different amount and summing the results. The 16-bit accumulated value will be returned to the 8-bit frame buffer, and it is therefor important to scale the result during the recover operation. This scaling essentially selects which part of the 16-bit range will map to representable RGB values. We describe a combiner configuration here which recovers the 16 bit accumulated value and copies the high order byte to the frame buffer.

1. The first step in the process is to make sure a carry operation has been performed. This clears the high-order bits of the Green and Blue channels, and prevents overflow in the subsequent steps.

2. The first general combiner stage is configured subtract a $1/32$ bias term from the Blue channel as described in section 5.2.1.

3. The second general combiner is configured to compute the dot product of the result from step 2 with the scale vector $(1,\ 1/16,\ 1/256)$. The resulting value is passed to the final combiner for rendering to the framebuffer.

This is illustrated in figure 5.13.

## 5.3 Other Accumulation Buffer Operations

The OpenGL 1.2.1 specification defines five accumulation buffer operations. The emulated accumulation buffer described here allows a subset of these operations.

GL_ACCUM This operation adds data to the accumulation buffer after first scaling the data by a user specified floating point factor. In the scheme described here, data is added to the framebuffer using *glBlend()* operations. No direct way of scaling the data is implemented. Some scaling effects can be emulated by manipulating the color table of palletted textures, but this has no effect on primary color, fog, and lighting effects.

GL_LOAD This operation is similar to GL_ACCUM in that it scales rendered data and transfers it to the accumulation buffer. The GL_LOAD operation is different from GL_ACCUM in that the scaled data replaces the contents of the accumulation buffer, instead of being added to them. This action can be emulated by specifying *glBlendFunc(*GL_ONE, GL_ZERO*)*, but as before arbitrary scaling is not implemented.

GL_RETURN This operation transfers data from the accumulation buffer back to the framebuffer, and corresponds to the step described in section 5.2.4.

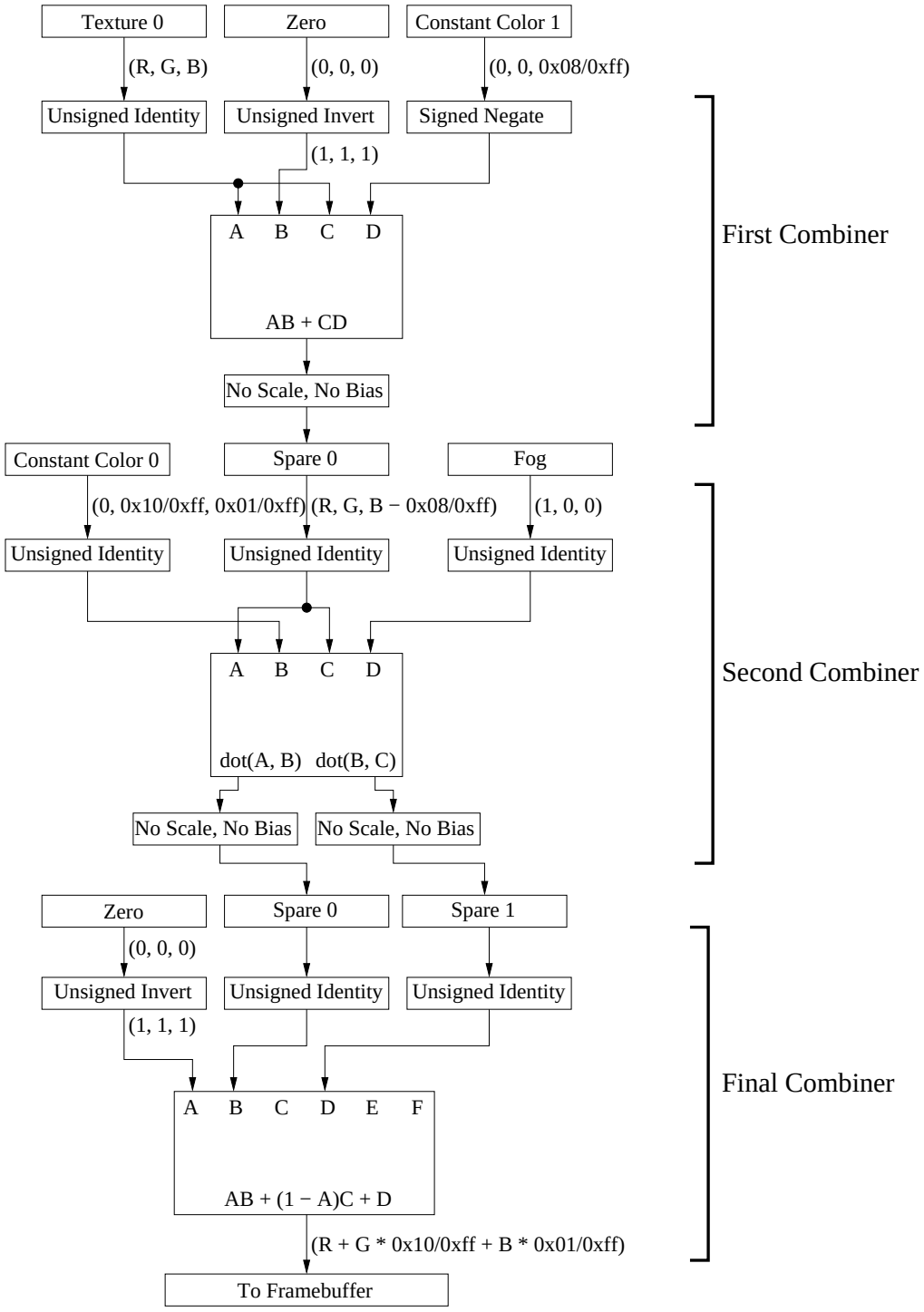



Figure 5.13: The distributed representation is consolidated using a dot-product operation. Scaling by factors of 2, 4, 8, 16, and 32 can be implemented using the register combiners input/output mappings.

GL_MULT This operation simply scales each value in the accumulation buffer by a user specified floating point factor. Scale factors which are integer powers of two can be implemented by shifting the frame buffer contents left or right, carrying bits as appropriate, but I'm not going to write this up until after I defend my thesis.

GL_ADD This is like GL_MULT, except that instead of scaling data in the accumulation buffer it simply adds a user specified constant. To emulate this, the user must render a constant value image to the framebuffer after setting *glBlendFunc()* to produce the desired result.

# Chapter 6

# Imager Calibration

Chapter 3 describe models for the attenuation of X-rays as they pass from a radiation source to the surface of an imager. It does not, however, describe how X-rays at the surface of the imager are converted to digital images. This image generation process is characterized by geometric distortions and intensity mappings specific to the imaging hardware and software. Some kinds of distortion can confuse the image comparison measures which are part of our registration process, so it is important to identify and correct these types of distortion. This section presents models for representing and correcting geometric distortions and intensity mappings.

The registration algorithms described in this thesis have been applied to two distinct types of X-ray imagers: the first type of imager is a fixed X-ray imager used in image guided radiosurgery; and the second kind of imager is a conventional film system. These two types of imagers are discussed in sections 6.1 and 6.2 respectively.

## 6.1 Fixed X-ray Imager

Chapter 7 describes the application of the 2D/3D registration algorithm to an image guided radiosurgery system. This system includes a pair of nearly orthogonal X-ray imagers. Each imager has its own diagnostic level X-ray source, which transmits radiation through the patient to a fluorescent screen. Energy from the X-rays causes the screen to fluoresce, and the resulting pattern of fluorescence is captured using an image intensified CCD camera and digitizer. The physical layout of the system is illustrated in figure 6.1.

We model this type of imager using three distinct sets of parameters: the first set of parameters describes the geometric mapping from 2D coordinates at the imaging surface to 2D coordinates in the output image; the second set of parameters describes the projection from 3D space onto the imaging surface; and the third set of parameters describes the intensity response of the system.

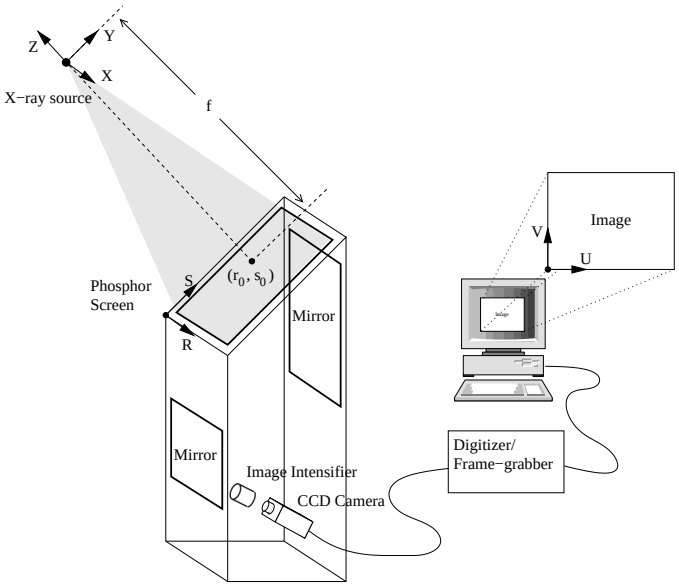



Figure 6.1: Components of the fixed X-ray imagers.

Calibration methods for these three sets of parameters are described in sections 6.1.1, 6.1.2, and 6.1.3 respectively.

Prior to all three of these calibrations, we define coordinate systems associated with the image and with the imager. We define a 3D coordinate system (X, Y, Z) associated with each imager so that the X-Y plane is parallel to the surface of the phosphor screen. The origin of this coordinate system is at the location of the corresponding X-ray source, and the Z axis points away from the imager. We define a 2D coordinate system (R, S) which describes positions in the image plane of the imager. This R axis is parallel to the X axis of the 3D coordinate system, and the S axis is parallel to the Y axis of the 3D coordinate system. Finally, we define a 2D coordinate system (U, V) associated with the image. Points in this coordinate system correspond to pixel locations in the digitized image. All three of these coordinate systems are shown in figure 6.1.

### 6.1.1 2D ↔ 2D Parameters

The first step in calibrating the fixed X-ray imager is to characterize the mapping between (R, S) coordinates at the surface of the imager and (U, V) coordinates in the output image. This mapping reflects any projections and distortions in the imaging chain between the phosphor screen and the output image. We model this mapping using two parts:

- Radial distortion, reflecting the characteristics of the focusing lens in the CCD camera.

- Projective transformation, reflecting the relative positions of the phosphor screen, mirrors, image intensifier and CCD camera.

Following Tsai [55], we model radial distortion as a function which takes points $[u, v]^T$ in the image coordinate system to point $[u', v']^T$ in an intermediate coordinate system. Tsai expresses the function in terms of distance from a central point $[u_0, v_0']^T$

$$d' = d\left(1 + \kappa \|d\|^2\right) \tag{6.1}$$

$$d = \begin{bmatrix} u \\ v \end{bmatrix} - \begin{bmatrix} u_0 \\ v_0 \end{bmatrix} \tag{6.2}$$

$$d' = \begin{bmatrix} u' \\ v' \end{bmatrix} - \begin{bmatrix} u_0 \\ v_0 \end{bmatrix}, \tag{6.3}$$

where $\kappa$, $u_0$, and $v_0$ are are the three parameters describing the radial distortion. The parameter $\kappa$ is called the *first-order radial distortion coefficient.* We rewrite equation 6.1 to make the relationship between $\begin{bmatrix} u \\ v \end{bmatrix}$ and $\begin{bmatrix} u' \\ v' \end{bmatrix}$ clearer:

$$\begin{bmatrix} u' \\ v' \end{bmatrix} = \begin{bmatrix} u_0 \\ v_0 \end{bmatrix} + \left(1 + \kappa \left\| \begin{bmatrix} u \\ v \end{bmatrix} - \begin{bmatrix} u_0 \\ v_0 \end{bmatrix} \right\|^2\right) \left( \begin{bmatrix} u \\ v \end{bmatrix} - \begin{bmatrix} u_0 \\ v_0 \end{bmatrix} \right). \tag{6.4}$$

We express the projective transformation as a homographic mapping from points $\begin{bmatrix} u' \\ v' \end{bmatrix}$ in the intermediate coordinate system to points $\begin{bmatrix} r \\ s \end{bmatrix}$ on the surface of the imager:

$$\begin{bmatrix} r \\ s \end{bmatrix} = \begin{bmatrix} (h_{(0,0)}u' + h_{(0,1)}v' + h_{(0,2)})/(h_{(2,0)}u' + h_{(2,1)}v' + 1) \\ (h_{(1,0)}u' + h_{(1,1)}v' + h_{(1,2)})/(h_{(2,0)}u' + h_{(2,1)}v' + 1) \end{bmatrix}, \tag{6.5}$$

where the eight parameters $h_{(i,j)}$ describe the the projection.

Image-intensified radiographic imagers are subject to another type of distortion, known as S-distortion, which varies depending on the characteristics of the local magnetic field. S-distortion is problematic for mobile fluoroscopic imagers because of its dependence on the the position and orientation of the imager. In our application, the imagers are rigidly attached to the operating room floor, and distortions remain consistent from image to image. In fact, the two components described above model the observed projection very well: the image pixel size is roughly 1mm, and RMS Residuals for the calibration routine described below are under 0.2mm, which is consistent with the

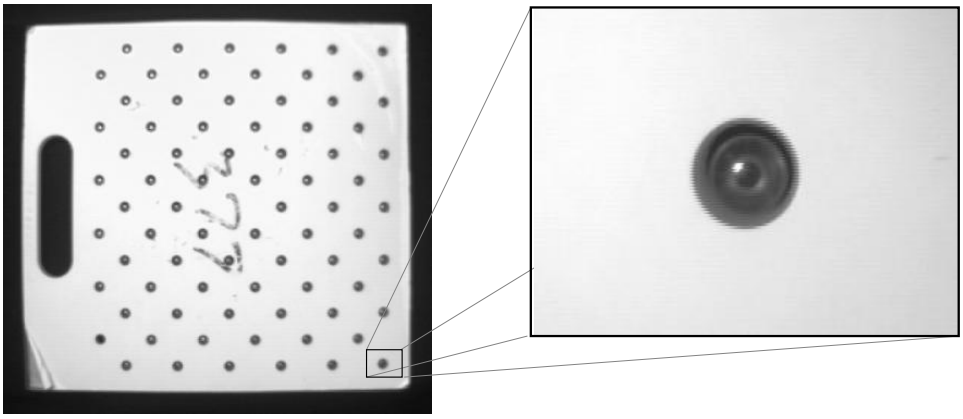


Figure 6.2: The geometric distortion calibration target holds 0.25in steel ball bearings in relative position. The force of gravity causes each ball bearing to rest against the downward edge of its hole.

expected observation noise. Consequently we do not model the effects of S-distortion for this type of imager.

### Calibration Target

The mapping between (R, S) coordinates and (U, V) coordinates was calibrated using a plastic panel having dimensions of 11.25” x 10.5” x 0.375”. The panel had 77 holes of approximately 0.3” diameter drilled through it at regularly spaced intervals. During calibration, the panel was placed flat against the face of imager, and 0.25” ball bearings were placed in the holes so that each rested flush against the surface of the phosphor screen. One hole was left empty so that the orientation of the target could be easily determined in the X-ray image. Although the holes in the panel were of larger diameter than the ball-bearings, each ball-bearing was pulled by gravity to the lowest position within its respective hole. This, together with the slant of the imager face, ensured that the relative positions of the ball-bearings reflected the actual drilled pattern, as shown in figure 6.2. An image was acquired, in which the ball bearings stood out clearly against the radiolucent plastic, and each ball bearing was located in the resulting image by fitting a series of concentric circles. These locations were used to calibrate the system as described in the next section.

### Calibration Algorithm

Referring to equations 6.4 and 6.5, we see that our model of the mapping between (R, S) coordinates and (U, V) coordinates has eleven geometric parameters. The first three parameters, $\kappa$, $x_0$, and $y_0$ describe the radial distortion, and the remaining eight parameters describe the homography in equation 6.5.

We cast the recovery of these eleven parameters as a nonlinear optimization problem. It is not necessary, however, to optimize in an eleven-dimensional space. The calibration is instead performed as an optimization over the three radial distortion parameters. The objective function for the optimization is computed as follows:

1. At each step of the optimization, the radial distortion parameters are applied to the observed positions of the calibration target fiducials equation 6.4.

2. Once equation 6.4 has been applied, the best fit homography can be approximated by solving a system of linear equations. We are looking for the eight parameter values which most nearly satisfy equation 6.5. This equation can be rearranged to give

$$\begin{bmatrix} r_n(h_{(2,0)}u'_n + h_{(2,1)}v'_n + 1) \\ s_n(h_{(2,0)}u'_n + h_{(2,1)}v'_n + 1) \end{bmatrix} = \begin{bmatrix} (h_{(0,0)}u'_n + h_{(0,1)}v'_n + h_{(0,2)}) \\ (h_{(1,0)}u'_n + h_{(1,1)}v'_n + h_{(1,2)}) \end{bmatrix}, \tag{6.6}$$

   where the points $[u'_n, v'_n]^T$ are drawn from the result of step 1, above, and the points $[r_n, s_n]^T$ are the corresponding (known) 2D positions of the ball bearings in the calibration target. We use the subscript $n$ to indicate that this equation holds for each of the 76 ball bearings in the image. A further rearrangement gives

$$\begin{bmatrix} u'_n & v'_n & 1 & 0 & 0 & 0 & -r_n u'_n & -s_n v'_n \\ 0 & 0 & 0 & u'_n & v'_n & 1 & -r_n u'_n & -s_n v'_n \end{bmatrix} \begin{bmatrix} h_{(0,0)} \\ h_{(0,1)} \\ h_{(0,2)} \\ h_{(1,0)} \\ h_{(1,1)} \\ h_{(1,2)} \\ h_{(2,0)} \\ h_{(2,1)} \end{bmatrix} = \begin{bmatrix} r_n \\ s_n \end{bmatrix}. \tag{6.7}$$

   Combining equation 6.7 over all the observed fiducials gives an overconstrained system of linear equations, which is easily solved using the Moore-Penrose pseudoinverse [43] [44].

3. Using the parameter values, $h_{m,n}$, recovered in step 2, the radially undistorted points from step 1 are projected into the (R, S) coordinate system. The objective function value is the RMS residual between these projected points and the known positions of the ball bearings in the calibration target.

We define an objective function which returns the RMS residual from step 3, and then estimate the optimal calibration parameters using the downhill simplex method of Nelder and Mead to minimize

this residual. [45]

Note that using the pseudoinverse in step 2, above, *doesn't* give values for $h_{(i,j)}$ which are least-squares optimal. This is because the algebraic manipulation in equation 6.6 weights each of the original equations by the quantity $h_{(2,0)}u'_n + h_{(2,1)}v'_n + 1$. If necessary, the estimate of the parameters $h_{(i,j)}$ can be refined iteratively by solving the system of equations 6.7, then weighting each equation by $1/(h_{(2,0)}u'_n + h_{(2,1)}v'_n + 1)$, and re-solving.

### The Geometry Corrected 2D Image

Once we have recovered the mapping between (R, S) coordinates at the surface of the imager and (U, V) coordinates in the output image, we define a function $p : \Re^2 \rightarrow \Re^2$ which implements this mapping. In other words, the function $p()$ takes points in (U, V) coordinates as arguments, and returns the corresponding points in (R, S) coordinates. The implementation of this function follows from equations 6.4 and 6.5.

It is also useful to compute the inverse mapping, $p^{-1}$, from points in (R, S) back to points in (U, V). The inverse of equation 6.5 is found by expressing the equality in homogeneous coordinates

$$\begin{bmatrix} \alpha r \\ \alpha s \\ \alpha \end{bmatrix} = \begin{bmatrix} h_{0,0} & h_{0,1} & h_{0,2} \\ h_{1,0} & h_{1,1} & h_{1,2} \\ h_{2,0} & h_{2,1} & 1 \end{bmatrix} \begin{bmatrix} u' \\ v' \\ 1 \end{bmatrix}. \tag{6.8}$$

Inverting this homogeneous equation gives

$$\begin{bmatrix} \beta u' \\ \beta v' \\ \beta \end{bmatrix} = \begin{bmatrix} h_{0,0} & h_{0,1} & h_{0,2} \\ h_{1,0} & h_{1,1} & h_{1,2} \\ h_{2,0} & h_{2,1} & 1 \end{bmatrix}^{-1} \begin{bmatrix} r \\ s \\ 1 \end{bmatrix}. \tag{6.9}$$

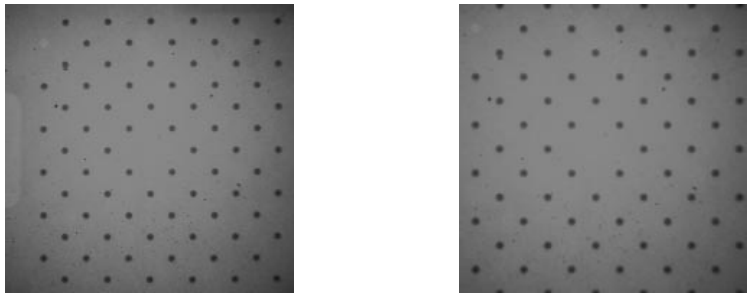
The corresponding values for $r$ and $s$ are easily found: $r = \beta r/\beta$, and $s = \beta s/\beta$.

We invert equation 6.4 through a straightforward application of Newton-Raphson iteration, however a closed form solution can be found using the cubic formula [45].

The significance of $p^{-1}$ is that it allows us, for any point in (R, S), to determine the corresponding point in (U, V). We can determine the image intensity at the corresponding (U, V) point by bilinear interpolation among the image pixels. In other words, $p^{-1}$ lets us remap the output image intensity values into (R, S) coordinates. We call the result of this mapping the *geometry-corrected* image, and write

$$I'(r, s) = I\left(p^{-1}(r, s)\right), \tag{6.10}$$

where $I(u, v)$ is the actual output image, and $I'(r, s)$ is the geometry-corrected image. Figure



(a) (b)

Figure 6.3: An image of the geometric distortion calibration target is shown in (a) and contains some small geometric distortions. The geometry-corrected image was sampled on a regular pixel grid, and is shown in (b).

### 6.1.2 3D $\leftrightarrow$ 2D Parameters

The image guided radiosurgery system includes two fixed X-ray imagers. We must characterize the relative positions and orientations of these imagers, as well as the 3D projection geometry of imager.

As before, we use a fiducial-based calibration technique. Figure 6.5 shows one fiducial, at position $(x_{\mathrm{f}}, y_{\mathrm{f}}, z_{\mathrm{f}})$ being projected onto the imaging surface. We define the scalar parameter $f$ to be the Z coordinate at which the Z axis intersects the surface of the imager, and we define $[r_0, s_0]$ to be the location of this intersection in (R, S) coordinates. Using similar triangles, we see that the following two equations hold

$$\frac{r_{\mathrm{f}} - r_0}{f} = \frac{x_{\mathrm{f}}}{z_{\mathrm{f}}} \tag{6.11}$$

$$\frac{s_{\mathrm{f}} - s_0}{f} = \frac{y_{\mathrm{f}}}{z_{\mathrm{f}}}, \tag{6.12}$$

where $[r_{\mathrm{f}}, s_{\mathrm{f}}]$ is the 2D point in (R, S) coordinates to which the fiducial projects. Rearranging these two equations, we can write a general homogeneous equation which relates the projection from 3D imager coordinates to 2D imager coordinates

$$\begin{bmatrix} \alpha r \\ \alpha s \\ \alpha \end{bmatrix} = \begin{bmatrix} f & 0 & r_0 & 0 \\ 0 & f & s_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}, \tag{6.13}$$

where the scale factor $\alpha$ is easily factored out. This is a simplified form of the standard pinhole

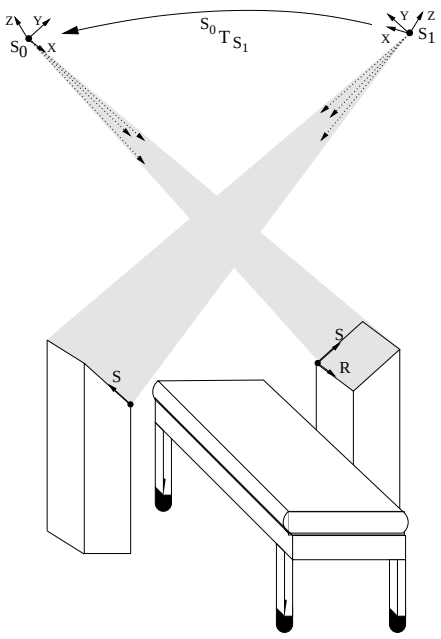



Figure 6.4: The treatment room contains two fixed X-ray imagers. The positions and orientations of these two imagers are related by the coordinate transformation ${}^{S_0}T_{S_1}$. For each imager, the projection from 3D coordinates to 2D coordinates depends on the position of the X-ray source with respect to the imager.

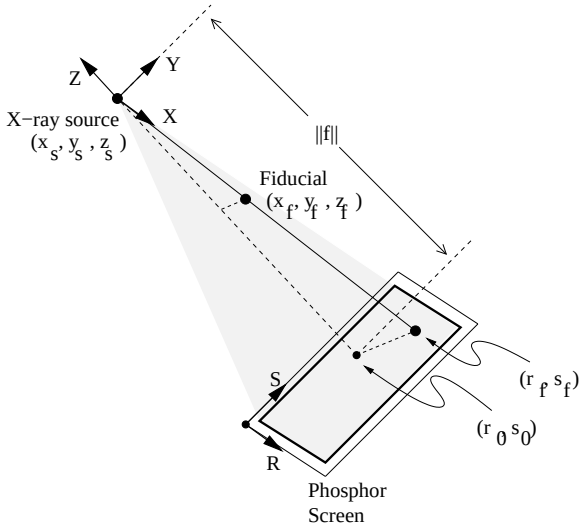



Figure 6.5: Projection geometry for 3D fiducials. A fiducial at $(x_{\mathrm{f}}, y_{\mathrm{f}}, z_{\mathrm{f}})$ projects to a coordinate $(r_{\mathrm{f}}, s_{\mathrm{f}})$ at the imager surface.

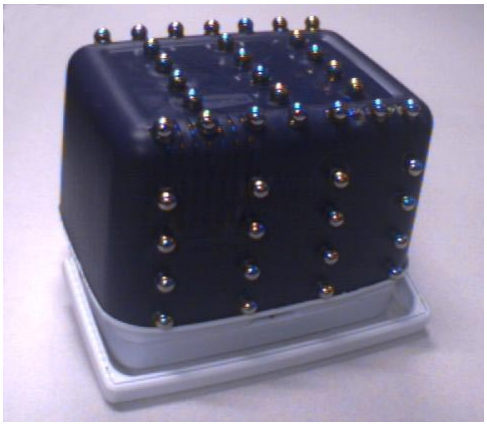


Figure 6.6: The calibration target for imager 3D geometry was constructed by attaching 58 steel ball bearings to the surface of a plastic six-pack cooler.

camera model discussed in [14]. Accordingly, we describe the projection using three intrinsic parameters: focal length, $f$, and center of projection $[r_0, s_0]$.

The final component of the 3D calibration is a coordinate transformation describing the relative position of the two imagers. This coordinate transformation is represented using a 4 by 4 transformation matrix ${}^{S_0}T_{S_1}$, which takes coordinates from the 3D coordinate system of the one imager to the 3D coordinate system of the other, as shown in figure 6.4.

### Calibration Target

A geometric calibration target was constructed from a plastic six-pack cooler, manufactured by Rubbermaid, Incorporated, and measuring approximately 23cm x 20cm x 15cm. 58 steel ball bearings, each having 0.5 in diameter were seated into the surface of the cooler by heating the plastic. After suitable surface preparation, each ball bearing was secured to its seat using epoxy.

An arbitrary coordinate system was defined by attaching an optical tracking beacon to the surface of the cooler. We call this the *target coordinate system*, Q. The position of each ball bearing was measured relative to the target coordinate system using an optically tracked pointer as described in appendix B.3, and the tracking beacon was removed from the cooler. A photograph of the completed target is shown in figure 6.6.

The calibration target was placed so that it lay within the field of view of both imagers, and a pair of images was acquired. This is illustrated in figure 6.7. This figure also illustrates two unknown coordinate transformations ${}^{\mathrm{S_0}}T_{\mathrm{Q}}$ and ${}^{\mathrm{S_1}}T_{\mathrm{Q}}$. These two coordinate transformations take coordinates in the target coordinate system to the corresponding points in the 3D coordinate systems of the two imagers. The positions of the calibration target fiducials were detected in each image and

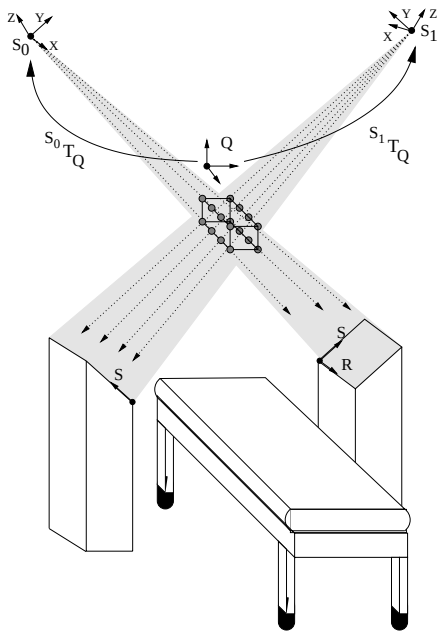



Figure 6.7: The 3D calibration target is viewed simultaneously with both imagers.

transformed using the nonlinear transformation $p()$ (page 84) in order to recover the corresponding positions in the 2D imager coordinate system (R, S) of each imager.

### Calibration Algorithm

We parameterized the two coordinate transformations, ${}^{\mathrm{S}_0}T_{\mathrm{Q}}$ and ${}^{\mathrm{S}_1}T_{\mathrm{Q}}$, using the seven element parameterization described in section 2.1.2. In addition, each imager has the three projection parameters, $f$, $r_0$, and $s_0$, for a total of ten parameters per imager. This part of the calibration is done for each imager independently. The calibration is cast as a nonlinear optimization over the seven elements in the parameterization of ${}^{\mathrm{S}_j}T_{\mathrm{Q}}$, where $j$ is 0 or 1, depending on which imager is being calibrated. The objective function for the optimization is computed as follows:

1. The transformation matrix ${}^{\mathrm{S}_j}T_{\mathrm{Q}}$ is recovered from the seven element parameter vector following equation 2.18.

2. The known 3D positions of the calibration fiducials in the target coordinate system are transformed into the 3D coordinate system of imager $j$ using ${}^{\mathrm{S}_j}T_{\mathrm{Q}}$. We write these transformed points $(x_i, y_i, z_i)$, where the subscript $i$ indicates that this coordinate corresponds to the $i^{\mathrm{th}}$ fiducial.

3. We are looking for the projection parameters $f$, $r_0$, and $s_0$ which most nearly satisfy equation 6.13 for each fiducial. By rearranging this equation and factoring out $\alpha$, we have

$$\begin{bmatrix} r_i z_i \\ s_i z_i \end{bmatrix} = \begin{bmatrix} f x_i + r_0 z_i \\ f y_i + s_0 z_i \end{bmatrix}, \tag{6.14}$$

where $(r_i, s_i)$ is the observed position of the $i^{\text{th}}$ fiducial in the (R, S) coordinate system of imager $j$. A further rearrangement gives

$$\begin{bmatrix} x_i & z_i & 0 \\ y_i & 0 & z_i \end{bmatrix} \begin{bmatrix} f \\ r_0 \\ s_0 \end{bmatrix} = \begin{bmatrix} r_i z_i \\ s_i z_i \end{bmatrix}. \tag{6.15}$$

Combining equation 6.15 over all of the 3D fiducials gives an overconstrained system of linear equations, which is easily solved using the Moore-Penrose pseudoinverse [43] [44].

4. Using the values for $f$, $r_0$, and $s_0$ computed in step 3, each 3D fiducial is projected into the (R, S) plane, and the RMS residual is computed between the projected 3D fiducials and the geometry-corrected observed fiducial positions. This residual is returned as the value of the objective function.

We define an objective function which returns the RMS residual from step 4, and then minimize this function using the downhill simplex method of Nelder and Mead [45]. After completing this minimization for each imager we have estimates for the two coordinate transformations ${}^{\mathrm{S}_0}T_{\mathrm{Q}}$ and ${}^{\mathrm{S}_1}T_{\mathrm{Q}}$, and for the two sets of projection parameters $f$, $r_0$, and $s_0$.

Note that using the pseudoinverse in step 3, above, *doesn't* give values of $f$, $r_0$, and $s_0$ which are least-squares optimal. This is because the algebraic manipulation in equation 6.14 essentially weights each of the original equations by the quantity $z_i$. If necessary, the estimate can be refined by iteratively solving the system of equations 6.15, then weighting each equation by $1/z_i$, and re-solving. In practice, the $z_i$ values are all relatively similar, and we do not do this reweighting.

### 6.1.3 Intensity Parameters

Figure 6.10 shows a pair of "blank" images acquired using the fixed X-ray imaging hardware. These images show significant spatial variation in intensity. Although some of this variation may be due to non-uniformity in the X-ray source, much of it is a result of non-uniform response in the imaging chain. This is clear because the spatial variation persists even when the phosphor screen is replaced with uniformly illuminated translucent plastic plate.

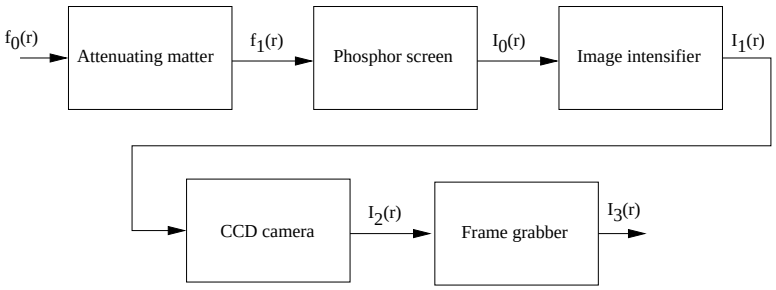



Figure 6.8: The gain characteristic of the fixed imager can be viewed as the composition of the characteristics of its components.

In order to represent this spatial variation, we model the imager as an array of independent gain elements. Each gain element corresponds to one pixel in the geometry-corrected image, and the corresponding intensity response reflects the combined characteristics of the phosphor screen, image intensifier, CCD camera, and frame grabber. The imaging chain is represented schematically in figure 6.8. We model the CCD camera and frame grabber as spatially uniform linear gains.

$$I_2(\boldsymbol{r}) = a_{\mathrm{cam}} I_1(\boldsymbol{r}) + b_{\mathrm{cam}}, \tag{6.16}$$

$$I_3(\boldsymbol{r}) = a_{\mathrm{fg}} I_2(\boldsymbol{r}) + b_{\mathrm{fg}}, \tag{6.17}$$

where $a_{\mathrm{cam}}$ and $b_{\mathrm{cam}}$ are parameters describing the gain of the CCD camera, $a_{\mathrm{fg}}$ and $b_{\mathrm{fg}}$ are parameters describing the gain of the frame grabber, and $\boldsymbol{r} = [r, s]^T$ is a position in 2D imager coordinates. The image intensifier is modeled as a linear gain as well, however it is assumed to have spatially varying characteristics

$$I_1(\boldsymbol{r}) = a_{\mathrm{ii}}(\boldsymbol{r}) I_0(\boldsymbol{r}) + b_{\mathrm{ii}}(\boldsymbol{r}), \tag{6.18}$$

where the functions $a_{\mathrm{ii}}(\boldsymbol{r})$ and $b_{\mathrm{ii}}(\boldsymbol{r})$ describe the spatially varying gain of the image intensifier.

We model the response of the phosphor as a uniform linear function of the incident photon fluence

$$I_0(\boldsymbol{r}) = a_{\mathrm{p}} f_1(\boldsymbol{r}), \tag{6.19}$$

where $a_{\mathrm{p}}$ is the single parameter describing the linear gain, and $f_1(\boldsymbol{r})$ is the photon fluence at the surface of the imager. When attenuating matter is present in the image, photon fluence at the surface of the phosphor screen decreases following an exponential attenuation law as described in section 3.1

$$f_1(\boldsymbol{r}) = f_0(\boldsymbol{r}) e^{-k_0 U(\boldsymbol{r})}, \tag{6.20}$$

where $U(\boldsymbol{r})$ is the log total attenuation between the radiation source and the point $\boldsymbol{r}$ on the surface of

the phosphor screen, and $f_0(\boldsymbol{r})$ is the unattenuated photon fluence. In other words, $f_0(\boldsymbol{r})$ describes the photon fluence at the surface when no attenuating matter is present. The constant $k_0$ is included to account for any scaling errors in our estimate of $U(\boldsymbol{r})$.

Composing these gains gives

$$I_3(\boldsymbol{r}) = a_{\mathrm{fg}}\left(a_{\mathrm{cam}}\left(a_{\mathrm{ii}}(\boldsymbol{r})a_{\mathrm{p}}f_0(\boldsymbol{r})e^{-k_0U(\boldsymbol{r})} + b_{\mathrm{ii}}(\boldsymbol{r})\right) + b_{\mathrm{cam}}\right) + b_{\mathrm{fg}} = a_{\mathrm{tot}}(\boldsymbol{r})f_0(\boldsymbol{r})e^{-k_0U(\boldsymbol{r})} + b_{\mathrm{tot}}(\boldsymbol{r}), \tag{6.21}$$

where $a_{\mathrm{tot}}(\boldsymbol{r}) = a_{\mathrm{fg}}a_{\mathrm{cam}}a_{ii}(\boldsymbol{r})a_{\mathrm{p}}$, and $b_{\mathrm{tot}} = a_{\mathrm{fg}}\left(a_{\mathrm{cam}}b_{\mathrm{ii}}(\boldsymbol{r}) + b_{\mathrm{cam}}\right) + b_{\mathrm{fg}}$. For convenience of notation, we define $I_{\mathrm{b}}(\boldsymbol{r}) = a_{\mathrm{tot}}(\boldsymbol{r})f_0(\boldsymbol{r})$, and express the gain characteristic of the entire system as

$$I_3(\boldsymbol{r}) = I_{\mathrm{b}}(\boldsymbol{r})e^{-k_0U(\boldsymbol{r})} + b_{\mathrm{tot}}(\boldsymbol{r}). \tag{6.22}$$

The unknown parameters in equation 6.22 are $I_{\mathrm{b}}(\boldsymbol{r})$, $k_0$, and $b_{\mathrm{tot}}(\boldsymbol{r})$. Given these, and the total linear attenuation $U(\boldsymbol{r})$ we can predict the pixel intensity $I_4(\boldsymbol{r})$ at any pixel in the geometry-corrected image. Note that $k_0$ is *not* spatially varying. The same value of $k_0$ is used for each pixel in the image.

One additional gain is not accounted for in this model. The image intensifier/CCD system incorporates an automatic gain adjustment to handle large changes in image input image brightness. It is possible to disable this variable gain, however the established treatment protocols do not do so. In practice, the range of input brightnesses for our images is fairly small, and our model fit very well even without modeling this gain.

### Calibration Target

In order to measure the intensity response of each imager, a series of phantoms were constructed using uniformly thick sheets of Solid Water. Solid Water is a commercial plastic, available from Gammex RMI, and has a known linear attenuation coefficient at diagnostic energies. These sheets were placed so that they occluded the entire phosphor screen as shown in figure 6.9.

Images of each phantom were acquired, and the image intensity $I_3(\boldsymbol{r})$ was measured. We denote the thicknesses of each phantoms by $d_i$, $0 \leq i < N$, where $N$ is the number of phantoms used. Each observation of $I_3(\boldsymbol{r})$ was recorded as a vector, $T_i$, where each element corresponds to one pixel in the corresponding $I_3$ image.

### Calibration Algorithm

The point of the intensity calibration targets is to provide known values of $U(\boldsymbol{r})$ in equation 6.22. It is clear from figure 6.9 that the distance a ray travels through the attenuating phantom, $q$, depends

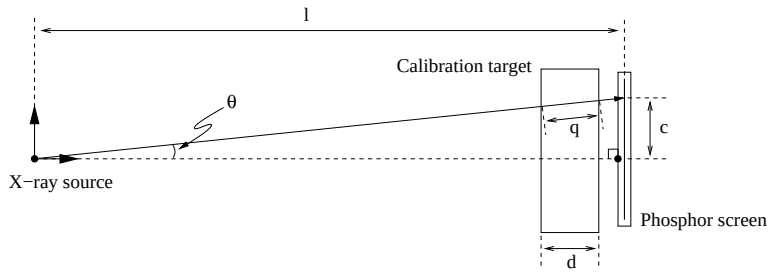



Figure 6.9: The constant density phantoms provide known values for $U(p(x))$.

on the angle $\theta$. That is,

$$q = \frac{d}{\cos(\theta)} = \frac{d\sqrt{l^2 + c^2}}{l}, \tag{6.23}$$

where $d$, $l$, $c$, and $\theta$ are dimensions as indicated in figure 6.9. For the imager in question, $l$ is about 3m̀, while $c$ reaches a maximum of about 25ċm, or 0.25m̀. Substituting these numbers into equation 6.23, we see that $q$ ranges from $d$ to about $1.003d$. In other words, the attenuation of photon fluence is very nearly uniform across the surface of the imager. Accordingly, we assume that each intensity calibration target introduces a constant attenuation over the surface of the imager having value

$$U(\boldsymbol{r}) = \mu_w d_i, \tag{6.24}$$

where $\mu_w$ is the linear attenuation coefficient of the target material, and $d_i$ is the thickness of the $i^{\text{th}}$ calibration target. Under this assumption, equation 6.22 becomes

$$I_3(\boldsymbol{r}) = I_{\text{b}}(\boldsymbol{r}) e^{-k_0 \mu_w d_i} + b_{\text{tot}}(\boldsymbol{r}). \tag{6.25}$$

Note that $e^{-k_0 \mu_t d_i}$ is a scalar. Remember that the the observations $T_i$ are simply measurements of $I_3$

$$T_i = \text{vec}\left( I_{\text{b}}(\boldsymbol{r}) e^{-k_0 \mu_w d_i} + b_{\text{tot}}(\boldsymbol{r}) \right). \tag{6.26}$$

This implies that the observations $T_i$ lie on a line in high-dimensional space with direction vector equal to $\frac{\text{vec}(I_{\text{b}})}{\|\text{vec}(I_{\text{b}})\|}$.

The best fit line for the observations $T_i$ is computed simply. We define the sample covariance matrix $K$ which describes the distribution of $T_i$

$$K = \left[ T_0 - \overline{T} \mid T_1 - \overline{T} \mid ... \mid T_{N-1} - \overline{T} \right] \left[ T_0 - \overline{T} \mid T_1 - \overline{T} \mid ... \mid T_{N-1} - \overline{T} \right]^T, \tag{6.27}$$

$$\overline{T} = \frac{1}{N} \sum_{i=0}^{N-1} T_i. \tag{6.28}$$

and compute the eigenvector $V$ corresponding to largest eigenvalue of $K$. The best fit line for the observations of $T_i$ lies parallel to this vector, and can be written parametrically:

$$S(\lambda) = \overline{T} + \lambda V, \tag{6.29}$$

We define a 1D coordinate system with its origin at $\overline{T}$ and its axis in the direction of $V$, and project each of the observations into this coordinate system.

$$t_i = (T_i - \overline{T}) \cdot V, \tag{6.30}$$

where $t_i$ is the 1D coordinate corresponding to the projected observation $T_i$. Referring to equation 6.22, we denote the projection of $\mathrm{vec}(I_\mathrm{b})$ into this coordinate system as $c$, and the projection of $\mathrm{vec}(b)$ as $\beta$, and we can write

$$t_i = ce^{-k_0\mu_w d_i} + \beta. \tag{6.31}$$

In this system of nonlinear equations there are three unknowns: $c$, $k_0$, and $\beta$. Currently we find the solution using an iterative nonlinear solver to minimize the quantity

$$E = \sum \left( ce^{-k_0\mu_w d_i} + \beta - t_i \right)^2. \tag{6.32}$$

After solution, the newly discovered constants $c$ and $b$ are projected back into high dimensional space to recover $I_\mathrm{b}(x)$ and $b_\mathrm{tot}(x)$ respectively

$$\mathrm{vec}(I_\mathrm{b}) = \overline{T} + cV \tag{6.33}$$

$$\mathrm{vec}(b_\mathrm{tot}) = \overline{T} + \beta V \tag{6.34}$$

Once the the gain parameters have been recovered, equation 6.22 can be inverted to give

$$U_{tot}(\boldsymbol{r}) = \frac{-log\left((I_3(\boldsymbol{r}) - b_\mathrm{tot}(\boldsymbol{r}))/I_\mathrm{b}(\boldsymbol{r})\right)}{k_0}. \tag{6.35}$$

The quantity $U_{tot}(\boldsymbol{r})$ will be used in the registration algorithm.

Figure 6.11 shows a pair of geometry-corrected intensity images taken directly from the image processing system. These images were processed using equation 6.35, and the resulting $U_{tot}$ is graphically represented in figure 6.12. For comparison, corresponding images were constructed using a transgraph, and are shown in figure 6.13.

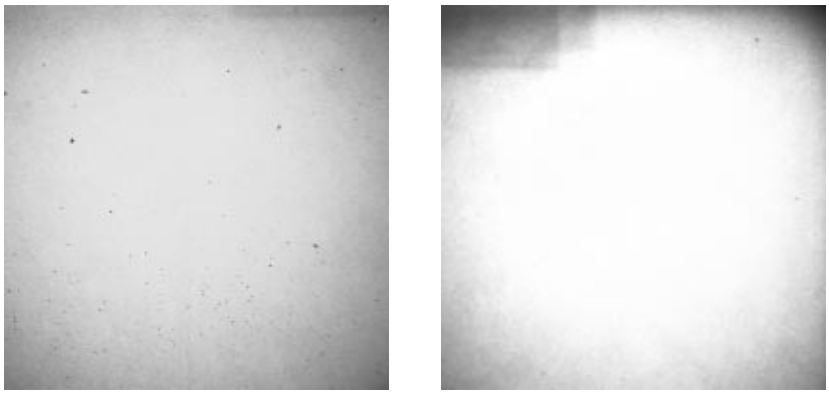


Figure 6.10: These images were collected with only air in the field of view of the imagers.

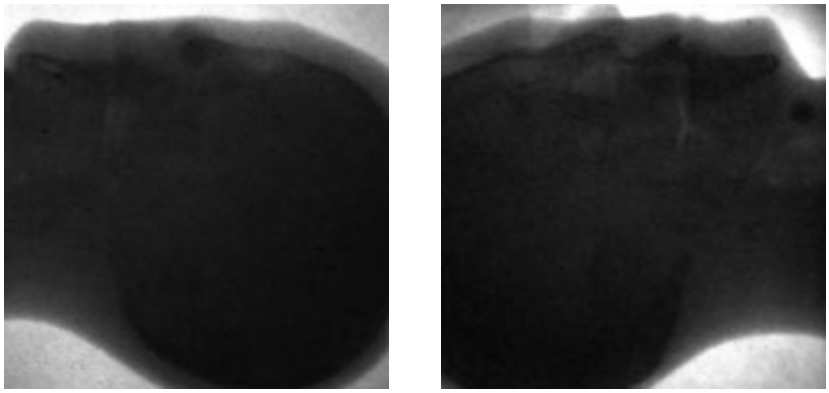


Figure 6.11: Sample geometry-corrected images from the X-ray imagers..

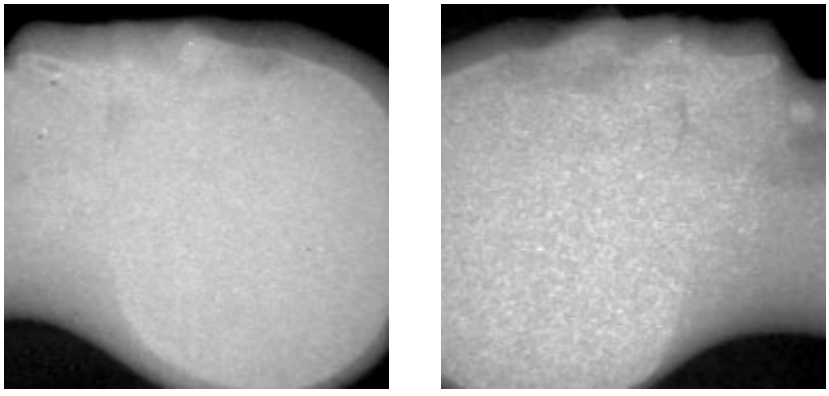


Figure 6.12: Recovered attenuation images after correction of geometric and intensity distortions.

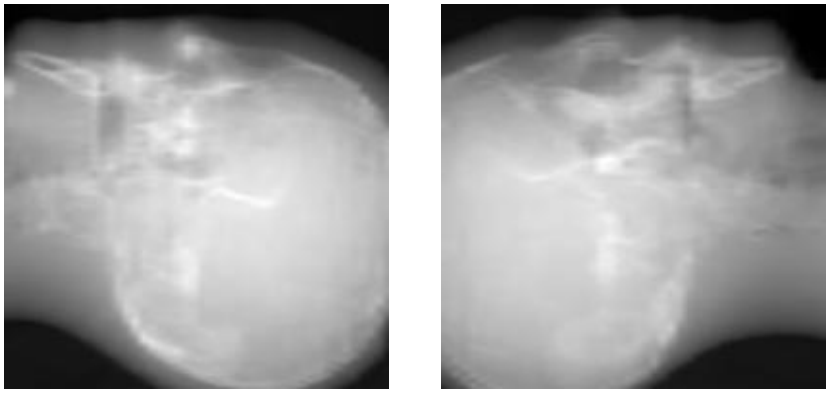


Figure 6.13: Synthetic images corresponding to the attenuation images of figure 6.12.

## 6.2 Film/Digitizer System

Our registration algorithm was tested with a second type of imager. This is a conventional radiographic imager which obtains images using X-ray film. Each image is acquired and developed using traditional radiological techniques. After development the film is digitized using an optical scanner, and the resulting digital images are used as input to the registration algorithm. The layout of this system is illustrated in figure 6.14.

### 6.2.1 Geometric Calibration

As with the fixed X-ray imager, we define three coordinate systems to help with calibration. We define a 3D coordinate system (X, Y, Z) which is attached to the to the X-ray source, and has its X-Y plane parallel to the film surface. The 2D coordinate system (R, S) lies in the plane of the film, and has its R axis parallel to the X axis of the 3D coordinate system. The S axis is parallel to the Y axis of the 3D coordinate system. Finally, we define a 2D coordinate system (U, V) associated with the output image. All three of these coordinate systems are shown in figure 6.14. The arrangement of these coordinate systems differs from that of the fixed X-ray imager only in that Z axis of the (X, Y, Z) coordinate system points towards the imaging surface, rather than away.

This projection geometry exactly matches the discussion in section 6.1.2. Accordingly, we write

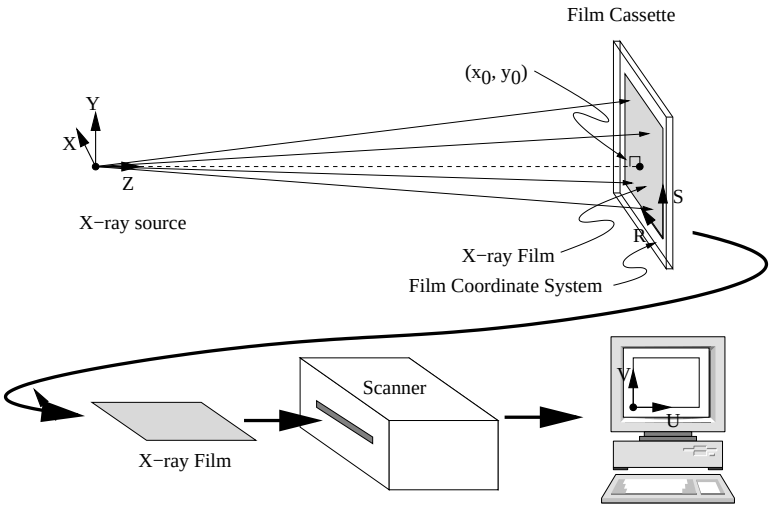



Figure 6.14: Schematic of the film/digitizer imaging system

the projection from 3D to 2D using homogeneous coordinates.

$$\begin{bmatrix} \alpha r \\ \alpha s \\ \alpha \end{bmatrix} = \begin{bmatrix} f & 0 & r_0 & 0 \\ 0 & f & s_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}, \tag{6.36}$$

where the scale factor $\alpha$ is easily factored out.

We assume that the film is dimensionally stable, and introduces no geometric distortions. The scanning and digitizing process introduces a 2D rigid transformation (rotation + translation) to the image, as well as rescaling along both axes. Accordingly, we can write the transformation from 2D film coordinates to 2D screen coordinates as

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} k_{\mathrm{u}} & 0 & 0 \\ 0 & k_{\mathrm{v}v} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\theta) & -\sin(\theta) & u_0 \\ \sin(\theta) & \cos(\theta) & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r \\ s \\ 1 \end{bmatrix}. \tag{6.37}$$

where $\theta$, $u_0$, and $v_0$ describe the transformation introduced by the scanner. The parameters $k_{\mathrm{u}}$, and $k_{\mathrm{v}}$ describe the spatial resolution of the final image along the U and V axes, respectively. Composing

equations 6.36 and 6.37 gives

$$\begin{bmatrix} \alpha u \\ \alpha v \\ \alpha \end{bmatrix} = \begin{bmatrix} k_{\mathrm{u}} & 0 & 0 \\ 0 & k_{\mathrm{v}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\theta) & -\sin(\theta) & u_0 \\ \sin(\theta) & \cos(\theta) & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & r_0 & 0 \\ 0 & f & s_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}. \quad (6.38)$$

Note that the choice of X-Y orientation for the imager coordinate system is completely arbitrary. The X and Y axes of this coordinate system need not correspond to any physical direction. Accordingly, we introduce a 3D rotation about the Z axis, having the same magnitude as the rotation introduced by the scanner,

$$\begin{bmatrix} \alpha u \\ \alpha v \\ \alpha \end{bmatrix} = \begin{bmatrix} k_{\mathrm{u}} & 0 & 0 \\ 0 & k_{\mathrm{v}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\theta) & -\sin(\theta) & u_0 \\ \sin(\theta) & \cos(\theta) & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & r_0 & 0 \\ 0 & f & s_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 & 0 \\ -\sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}, \quad (6.39)$$

which simplifies to:

$$\begin{bmatrix} \alpha u \\ \alpha v \\ \alpha \end{bmatrix} = \begin{bmatrix} k_{\mathrm{u}} & 0 & 0 \\ 0 & k_{\mathrm{v}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & d_{\mathrm{u}} & 0 \\ 0 & f & d_{\mathrm{v}} & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}, \quad (6.40)$$

where $d_{\mathrm{u}} = \cos(\theta) r_0 - \sin(\theta) s_0 + u_0$ and $d_{\mathrm{v}} = \cos(\theta) r_0 + \sin(\theta) s_0 + v_0$ define the *effective center of projection* of the imaging system. Under this model, the imaging system has five geometric parameters, $f$, $d_{\mathrm{u}}$, $d_{\mathrm{v}}$, $k_{\mathrm{u}}$, and $k_{\mathrm{v}}$.

### Calibration target

A calibration target was constructed of 0.375" Lexan sheet. The Lexan was used to build a cube measuring 9" on each side. A total of 18 steel ball bearings were implanted in the top and bottom faces of the cube, 9 bearings per face on 3.625" centers. The ball bearings have a diameter of 0.125", and are clearly visible in X-ray images of the cube. The completed assembly is shown in figure 6.15.

### Calibration algorithm

The five geometric calibration parameters, $f$, $d_{\mathrm{u}}$, $d_{\mathrm{v}}$, $k_{\mathrm{u}}$, and $k_{\mathrm{v}}$, are recovered in straightforward fashion. The pixel resolution of the output image is a function of the digitization process, and is controlled by the scanning hardware to be one of several standard values. The medical digitizer

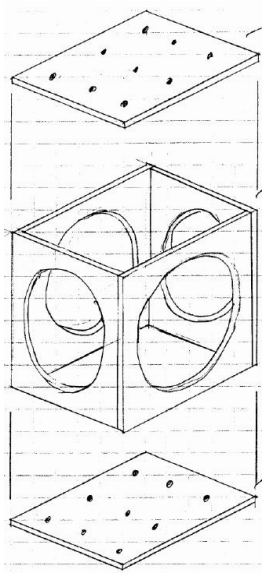


Figure 6.15: Calibration cube for film based imaging system.

used in these studies was configured to produce images with isotropic pixel spacing of 72 pixels per inch. This corresponds to $k_{\mathrm{u}} = k_{\mathrm{v}} = 3.528 * 10^{-5}$ meters per pixel. Once $k_{\mathrm{u}}$ and $k_{\mathrm{v}}$ are known, equation 6.40 can be rearranged to give

$$\begin{bmatrix} \frac{\alpha u}{k_{\mathrm{u}}} \\ \frac{\alpha v}{k_{\mathrm{v}}} \\ \alpha \end{bmatrix} = \begin{bmatrix} f & 0 & d_{\mathrm{u}} & 0 \\ 0 & f & d_{\mathrm{v}} & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}.$$

To calibrate the remaining parameters, the Lexan target was held in the field of view of the imager during image acquisition, and the projected positions of the ball bearings were measured in the output image. The calibration procedure is identical to that of section 6.1.2, except that equation 6.15 is replaced with

$$\begin{bmatrix} x_i & z_i & 0 \\ y_i & 0 & z_i \end{bmatrix} \begin{bmatrix} f \\ d_{\mathrm{u}} \\ d_{\mathrm{v}} \end{bmatrix} = \begin{bmatrix} u_i z_i \\ v_i z_i \end{bmatrix}. \tag{6.41}$$

An equivalent objective function was defined, and the downhill simplex method of Nelder and Mead was to recover the three remaining calibration parameters, $f$, $d_{\mathrm{u}}$, and $d_{\mathrm{v}}$.

### 6.2.2 Intensity Parameters

The intensity response of X-ray film has been studied in detail. Unfortunately, the intensity response varies depending on the specific type of X-ray film used. Even worse, substantial variations have been observed even between samples of the same film type. To address this problem, we do not explicitly model the intensity response of the film/digitizer imaging system, and instead choose image comparison metrics which are robust to nonlinearities in image intensity.

# Chapter 7

# Image-guided Radiosurgery

In intracranial radiosurgery, cancer cells inside a patient's head are killed using a beam of ionizing radiation. Registration is traditionally performed using a stereotactic frame. Traditional stereotactic frames provide accurate registration, however their use is uncomfortable for the patient, relatively invasive, and requires that preoperative CT scan, treatment planning, and surgical procedure all be performed within a short period of time.

The system described in this thesis permits registration without the use of a stereotactic frame. This allows much greater flexibility in scheduling treatments. Therapeutic radiation can be delivered in a series of small doses over the course of weeks. This *temporal fractionation* of treatment is often correlated with better surgical outcomes [1]. In addition, we anticipate that the ability to register the patient without the use of a stereotactic frame will open the door to accurate extracranial registration for treatment of tumors in the neck and abdomen.

We have evaluated the X-ray/CT registration system using data from an existing image-guided radiosurgery system. This system uses high-energy therapeutic radiation from a source external to the patient to kill cancer cells located inside the patient. In order to minimize damage to healthy tissue, therapeutic radiation is applied in several beams which overlap only in the vicinity of the tumor. The positions and orientations of these beams, as well as their intensities, durations, and other parameters, are specified in a presurgical plan, which is generated by a medical physicist. The goal of the presurgical plan is to maximize the radiation dose delivered to the tumor and minimize the dose delivered to normal tissues. Ideally, the diseased tissue is exposed to the combined radiation from all of the beams, while nearby healthy tissue receives a much smaller dose. This is illustrated in figure 7.1.

Precise intra-operative delivery of the specified beams requires that the patient and the treatment device be registered to a common coordinate system. For radiosurgical applications involving intracranial tumors, this registration is typically performed using a stereotactic frame. For thoracic

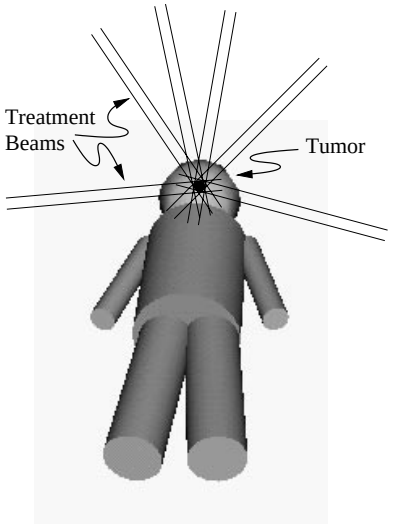



Figure 7.1: Treatment beams overlap at the tumor

and abdominal tumors, such as prostate and lung tumors, registration is traditionally performed by manually aligning the patient so that tattooed skin markings coincide with three laser beams which project from the sides and ceiling of the treatment room. Laser alignment is non-invasive and permits scheduling flexibility, but typically results in significant registration errors. Standard deviations in registration accuracy as high as 5mm are not uncommon [47] [25]. Large registration errors necessitate large treatment margins, which in turn increase the total radiation dose to the patient.

This chapter describes experiments in which we estimate registration errors using an anthropomorphic head phantom. We demonstrate relative registration errors of less than 1mm RMS, and worst case 3D registration errors of approximately 3 mm. Section 7.1 describes the radiosurgery system on which the registration algorithm was tested, section 7.2 describes an experiment using an anthropomorphic phantom, and section 7.3 presents the results of this experiment.

## 7.1 Hardware

We tested our system using images from a Neurotron 1000 Cyberknife (N1000), manufactured by Accuray, Inc. of Sunnyvale, California. The N1000 is currently undergoing clinical trials at the University of Pittsburgh Medical Center in Pittsburgh, PA, as well as several other sites around the United States.

The N1000 consists of a six-axis robotic manipulator which carries a 6 MV X-ray producing linear accelerator. The patient lies still on a treatment couch, while the robotic manipulator moves

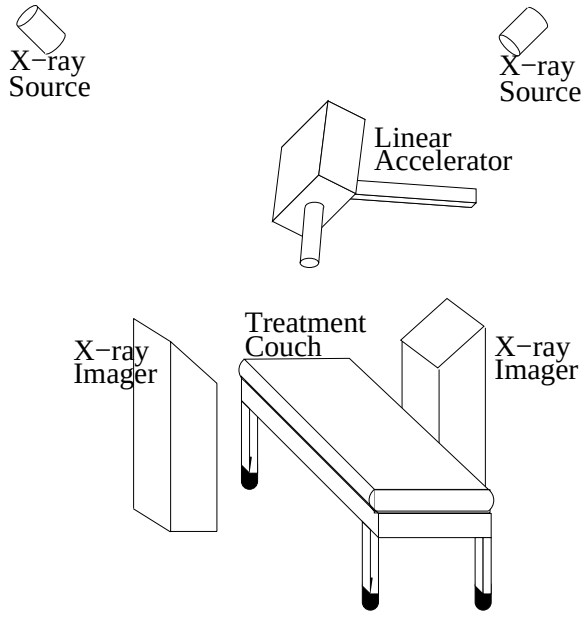



Figure 7.2: The Accuray Cyberknife

the radiation source around him. The manipulator pauses at a sequence of *treatment nodes* to deliver therapeutic radiation in accordance with the presurgical plan. The physical layout of the system is illustrated in figure 7.2.

The current Accuray registration system includes two nearly orthogonal diagnostic level X-ray imagers. Each imager has its own X-ray source, which transmits radiation through the patient to a fluorescent screen. Energy from the X-rays causes the screen to fluoresce, and the resulting pattern of fluorescence is captured using an image-intensified CCD camera and digitizer. This imager is described more fully in chapter 6, and is illustrated in figure 6.1.

Before the high energy therapeutic beam is applied from each treatment node, a pair of low energy X-ray images is acquired from each camera and used to measure the position of the patient. The registration system supplied with the N1000 is described in [4]. It measures patient translation by comparing the X-ray images with a catalog of synthetic X-ray images (DRRs), which have been computed off-line using a presurgically acquired CT scan. but does not measure patient rotation, and consequently requires the use of an external fixation device. This limitation of the existing system motivates the application of our registration algorithm to this hardware.

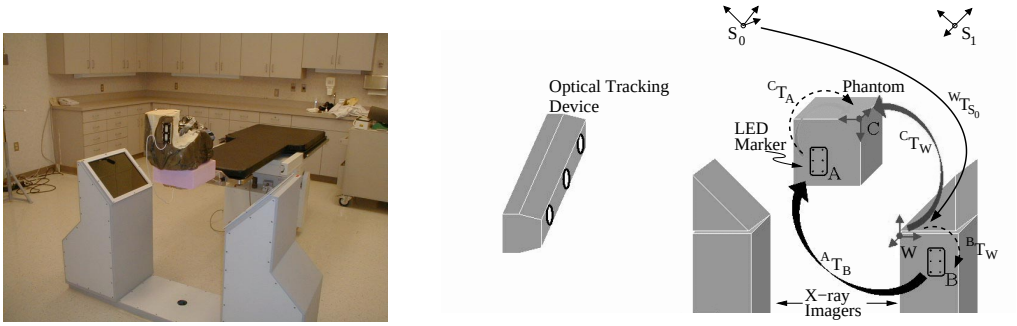



Figure 7.3: Experimental Setup.

## 7.2 Experiment

An anthropomorphic Rando® head phantom, manufactured by The Phantom Laboratory, Incorporated, was tracked through a sequence of poses using both our registration algorithm and an independent optical tracking device. A photograph of the experimental setup is shown in figure 7.3, along with a labeled schematic. The head was packed in low-density alpha-cradle foam, and placed on a movable table to approximate the position of a patient's head during treatment. An LED marker was attached to the foam in such a way that it was outside of the field of view of the X-ray imagers during the entire experiment. This marker is labeled "A" in the schematic. A second LED marker was attached to the housing of one of the X-ray imagers so as to provide a reference frame during the experiment. This marker is labeled "B" in the schematic. Because marker A was attached to the phantom, it had a fixed position with respect to the CT coordinate system, which is labeled "C" in the schematic. Similarly, marker B had a fixed position with respect to the imager to which it was attached, and consequently with respect to the world coordinate system, W, which was associated with the treatment room, and chosen to be roughly aligned with coordinate system B. The position and orientation of this coordinate system were specified with respect to the 3D coordinate system of the rightmost imager, which is labeled $S_0$ in figure 7.3(b). We represent the position and orientation of coordinate system $W$ with respect to the $S_0$ coordinate system using the matrix transformation ${}^{\mathrm{W}}T_{\mathrm{S}_0}$.

The two imagers were calibrated as described in section 6.1. This calibration measures the geometric and intensity characteristics of each imager, and also estimates the relative positions and orientations of the two 3D imager coordinate systems, $S_0$ and $S_1$.

Several aluminum spheres of 1cm diameter were attached to the outside of the alpha-cradle. These fiducials were used to establish ground truth measurements of the phantom position and orientation as described in section 7.2.1. The phantom was moved through a series of 358 poses

spanning roughly 3cm of translation along each axis and approximately $10^o$ of rotation around each axis. At each pose, a pair of X-ray images was acquired, and the positions of the LED markers were recorded using a Northern Digital Optotrak 3D sensing device. A CT scan of the phantom was acquired, having an in-slice pixel size of approximately 1 mm and an inter-slice spacing of 3 mm. Using this CT volume, the pose of the phantom was estimated from each pair of images using the gradient based registration algorithm of section 2.3, and the recovered pose estimates were compared with the Optotrak measurements.

### 7.2.1 Ground Truth

The registration algorithm provides an estimate of the position and orientation of the CT coordinate system with respect to the world coordinate system. This estimate is illustrated with an arrow between coordinate systems W and C in figure 7.3. We represent this coordinate transformation using the 4x4 matrix ${}^{\mathrm{C}}T_{\mathrm{W}}$. The Optotrak provides a measurement of the coordinate transformation between the two LED markers B and A. This transformation is illustrated with an arrow connecting A and B in figure 7.3, and represented using the 4x4 matrix ${}^{\mathrm{A}}T_{\mathrm{B}}$. Transformations ${}^{\mathrm{C}}T_{\mathrm{W}}$ and ${}^{\mathrm{A}}T_{\mathrm{B}}$ are related in that the both reflect the same motions: coordinate system A is rigidly attached to coordinate system C, while coordinate system B is rigidly attached to coordinate system W. They differ, however, in that the coordinate system associated with the LED marker A is not coincident with the coordinate system of the CT volume, while the coordinate system of marker B is not coincident with the world coordinate system, W. Before the ground truth measurements can be directly compared with the pose estimates, it is necessary to find the transformation between coordinate systems A and C, and also the transformation between coordinate systems W and B. These coordinate transformations correspond to the two dotted arrows in figure 7.3. We represent them using the matrices ${}^{\mathrm{C}}T_{\mathrm{A}}$ and ${}^{\mathrm{B}}T_{\mathrm{W}}$. Once ${}^{\mathrm{C}}T_{\mathrm{A}}$ and ${}^{\mathrm{B}}T_{\mathrm{W}}$ have been recovered, they can be composed with the Optotrak measurement, ${}^{\mathrm{A}}T_{\mathrm{B}}$, to provide a ground truth measurement of the transformation from coordinate system W to coordinate system C. This ground truth is an independent estimate of ${}^{\mathrm{C}}T_{\mathrm{W}}$ which can be directly compared to the 2D/3D registration result. We call this *comparison of absolute motion*.

#### Why absolute motion is important

When ${}^{\mathrm{C}}T_{\mathrm{A}}$ and ${}^{\mathrm{B}}T_{\mathrm{W}}$ are not known, direct comparison of ${}^{\mathrm{C}}T_{\mathrm{W}}$ and ${}^{\mathrm{A}}T_{\mathrm{B}}$ is not possible, and the study can only evaluate the *relative motion* reported by the registration system and by the Optotrak. For example, if the phantom is moved in a straight line, then the corresponding pose estimates can be fit to a line, and the relative spacing of the two sequences of translations can be compared. Murphy [42] uses this approach to evaluate a 2D/3D registration system, plotting relative errors for each of six pose parameters. In general, the computation of relative motion error involves implicitly

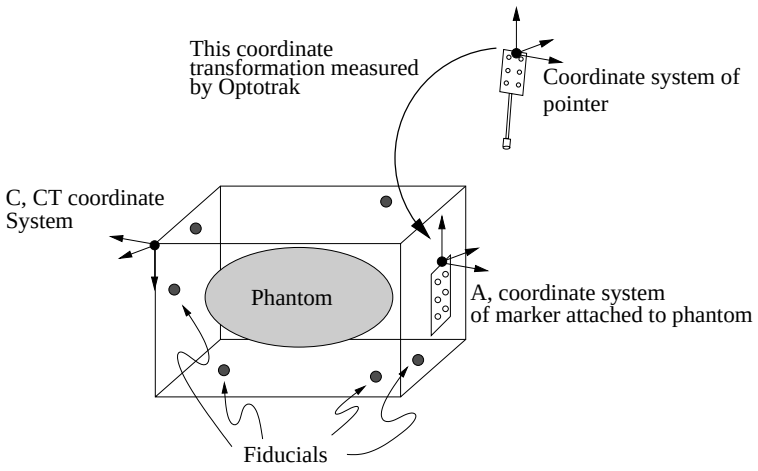



Figure 7.4: Aluminum fiducials have roughly the density of bone, and can be located in both the CT coordinate system and the coordinate system of the optical marker.

or explicitly fixing some of the unknown parameters at their best fit values for the observed data. If the actual system geometry differs from the hypothesized values, actual registration errors will be higher than that reported by relative motion comparisons. Unless it is clear that the best fit parameters reflect the actual system geometry, relative motion results should be used as a measure of "best case" registration accuracy only.

### Recovering ${}^{\mathrm{C}}T_{\mathrm{A}}$

The coordinate transformation matrix ${}^{\mathrm{C}}T_{\mathrm{A}}$ was measured using several spherical aluminum fiducials, which were attached to the phantom prior to the experiment, as illustrated in figure 7.4. The 3D locations with respect to the CT coordinate system were found by local center-of-mass calculations in the CT volume. The accuracy of this center of mass measurement was improved by acquiring a supplemental CT scan in the neighborhood of each fiducial. This supplemental scan had an in-slice voxel size of approximately 1 mm and an inter-slice spacing of 1 mm, and was not used in any other part of the experiment.

The local center of mass computation was initialized based on a priori knowledge of the CT orientation and the initial fiducial placement. A volume of interest was defined around each approximate fiducial position, and the enclosed voxels were compared against an intensity threshold. The fiducial location in CT coordinates was computed by a weighed average of those voxels passing the threshold

$$\boldsymbol{p}_i = \frac{\sum_{\boldsymbol{x}\in R_i} k(\boldsymbol{x}) v(\boldsymbol{x}) \boldsymbol{x}}{\sum_{\boldsymbol{x}\in R_i} k(\boldsymbol{x})}, \tag{7.1}$$

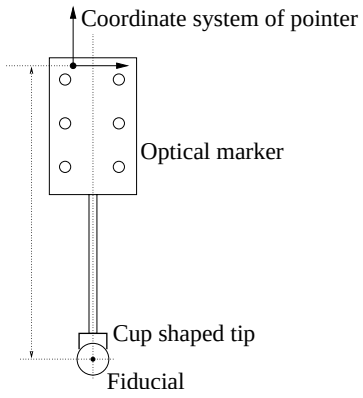



Figure 7.5: The cup shaped pointer tip mates with the spherical fiducials in a repeatable way.

where $\boldsymbol{p}_i$ is the location of the $i^{\text{th}}$ fiducial, $R_i$ is the set of voxel coordinates which make up the volume of interest surrounding the $i^{\text{th}}$ fiducial, $\boldsymbol{x}$ is a 3D vector describing the CT coordinates of a particular voxel, $v(\boldsymbol{x})$ is the CT value at voxel $\boldsymbol{x}$, and $k(\boldsymbol{x})$ is defined below:

$$k(\boldsymbol{x}) = \begin{cases} 1, & c_0 \leq v(\boldsymbol{x}) < c_1 \\ 0, & \text{otherwise} \end{cases} , \tag{7.2}$$

where $c_0$ and $c_1$ are thresholds which depend on the density of the fiducials and the specifics of the CT scan. We select these thresholds empirically.

The positions of the fiducials in the coordinate frame of LED marker A were found using an optically tracked pointer. The tip of the pointer was cup shaped, so that when the cup is placed against the surface of a spherical fiducial, the rim of the cup engaged the surface of the sphere in a repeatable way, as shown in figure 7.5. The pointer was was calibrated as described in appendix B.3.

Once the positions of the aluminum spheres were known in both coordinate system $A$ and coordinate system C, the coordinate transformation matrix ${}^{\text{C}}T_{\text{A}}$ was found by a rigid registration using Horn's method [23].

### Recovering ${}^{\text{B}}T_{\text{W}}$

One of the verification procedures for the Cyberknife radiosurgery system involves a vertical aluminum post which attaches to the floor of the treatment room. The top end of this post is a truncated cone, and its image is visible in both of the X-ray imagers. This post is shown schematically in figure 7.6(a). The truncated cone projects into the X-ray images as a triangle with one of its corners cut off, as shown in figure 7.6(b).

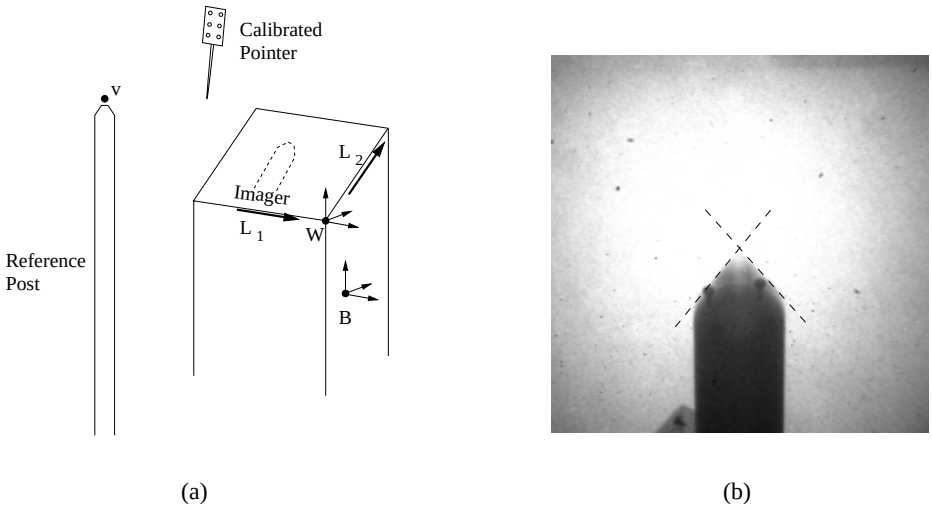



Figure 7.6: (a) The stationary coordinate system W can be registered with the coordinate system of the Optotrak marker, B, based on measurements with a calibrated pointer. The pointer is used to locate point $v$ and to trace lines $L_1$ and $L_2$. (b) The tip of the cone can be found in both X-ray images by fitting lines to the sides of its projection and computing the intersection of those lines.

An image was acquired from each imager in which the truncated cone was clearly visible. Each image was remapped as described in section 6.1.1 to correct geometric distortions, and line segments were fit to the two sides of the triangle adjacent to the missing vertex. These lines are illustrated in figure 7.6(b). The missing vertex was located by finding the intersection of the two lines in each image.

The location of each missing vertex was back-projected into 3D space as illustrated in figure 7.7, and the 3D position of the vertex of the truncated cone was found as follows: Two 3x4 matrices $P_0'$ and $P_1'$ were defined

$$P_0' = P_0 \left( {}^{W}T_{\mathrm{S}_0}^{-1} \right) \tag{7.3}$$

$$P_1' = P_1 \left( {}^{S_0}T_{\mathrm{S}_1}^{-1} \right) \left( {}^{\mathrm{W}}T_{\mathrm{S}_0}^{-1} \right), \tag{7.4}$$

where ${}^{\mathrm{W}}T_{\mathrm{S}_0}$ is the 4x4 matrix transformation which describes the position of coordinate system $W$ with respect to the 3D imager coordinate system $S_0$, and is defined on page 104. ${}^{S_0}T_{\mathrm{S}_1}$ is a 4x4 matrix transformation relating the 3D coordinate systems of the two imagers, and is defined in section 6.1.2. The two 3x4 projection matrices $P_0$ and $P_1$ describe the projection geometry of the two imagers, and are taken from equation 6.13 on page 85. We know that the 3D position of the

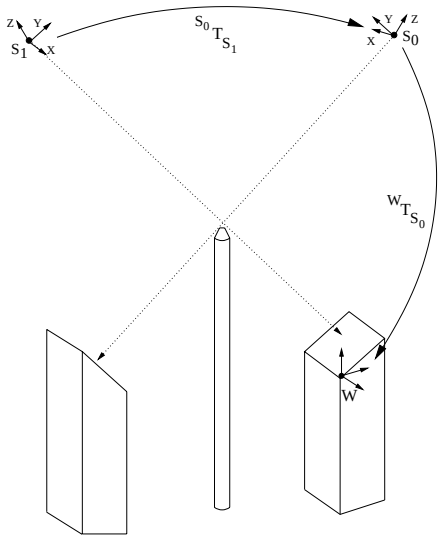



Figure 7.7: The position of the cone vertex is found with respect to coordinate system W by back-projecting from the two images.

cone vertex satisfies the homogeneous equations

$$\alpha \boldsymbol{r}_0 = P_0' \boldsymbol{v} \tag{7.5}$$

$$\beta \boldsymbol{r}_1 = P_1' \boldsymbol{v}, \tag{7.6}$$

where $\boldsymbol{r}_0$ and $\boldsymbol{r}_1$ are the 2D homogeneous positions of the projected vertex in the images from imager 0 and imager 1 respectively, and $\boldsymbol{v}$ is the 3D homogeneous position of the cone vertex in world coordinates. Expanding equations 7.5 and 7.6, and multiplying through by $\alpha$ and $\beta$ respectively, gives a system of linear equations which is easily solved for $\boldsymbol{v}$. This gives the position of the cone vertex with respect to coordinate system W.

To find the position of the cone vertex with respect to coordinate system B, an optically tracked pointer having a sharp tip was constructed and calibrated as described in appendix B.3. This pointer was used to record the coordinates of a set of points on the surface of the cone, and a least squares fit of a conical surface was used to determine the position of $v$ in the coordinate system of marker B.

The position of the cone vertex provides an absolute point of reference in both coordinate systems, and fixes the relationship between the two coordinate systems up to a rotation.

During calibration of the imaging system, the 3D coordinate system $S_0$ was defined with respect to the imaging phosphor of imager A. Consequently, its X and Y axes are parallel to edges of the phosphor screen. These edges are labeled $L_1$ and $L_2$ in figure 7.4(b). The orientation of coordinate

system W was established by tracing these two edges with the calibrated pointer and fitting line segments to the two trajectories.

## 7.3 Results

At each pose in the study, the registration algorithm was used to estimate the position and orientation of the CT volume with respect to coordinate system W. Image comparison was done using the SLNC metric of section 2.2.2, and DRRs were generated using the software based technique of chapter 3. Optimization was done using the Quasi-Newton method of Broyden, Fletcher, Goldfarb, and Shanno [45]. Incorrect convergence of the optimization was detected and remedied by discarding any optimizations which terminated with a final value more than three standard deviations above the group mean. These cases were marked as poorly convergent and discarded from the sample set. From the original group of 358 poses, 6 cases showed incorrect convergence.

For the remaining 352 poses, corresponding ground truth measurements were made by composing the Optotrak measurements with the transformations recovered in section 7.2.1. The registration results and ground truth measurements were compared as described in the following sections.

### 7.3.1 Pose Parameter Error

The ground truth measurements and and pose estimates were both represented as six element vectors of translations and rotations as described in section in section 2.1.1, with the exception that rotations were expressed around a point inside the head, rather than around the origin of the CT coordinate system. We write this pose parameter vector $[x,\ y,\ z,\ \theta_{\mathrm{x}},\ \theta_{\mathrm{y}},\ \theta_{\mathrm{z}}]$. The physical interpretation of this vector is that $x$, $y$, and $z$ represent the position of the rotation center in the world coordinate system, while $\theta_{\mathrm{x}}$, $\theta_{\mathrm{y}}$, and $\theta_{\mathrm{z}}$, are rotations around the X, Y, and Z axes respectively. Figure 7.8 shows the ground truth measurements for each pose, and figure 7.9 shows the corresponding pose estimates generated by our algorithm. For each pose, the difference between the pose estimate and ground truth measurement was computed, as shown in figure 7.10.

RMS translation errors were 0.39 mm, 0.18 mm, and 0.48 mm along the X, Y, and Z axes respectively, while RMS rotation errors were $0.22^o$, $0.99^o$, and $0.34^o$ around the X, Y, and Z axes respectively. There is some bias in the rotation estimates: mean rotation errors are $-0.15^o$, $0.97^o$, and $-0.31^o$ around the X, Y, and Z axes respectively.

This bias suggests a systematic error. Some of this is almost certainly due to errors in the estimation of the two transforms in section 7.2.1 during recovery of the ground truth. Accordingly, we also present relative motion results, which are not dependent on these measurements.

A set of 25 poses was randomly selected, and new estimates of for the coordinate transfor-

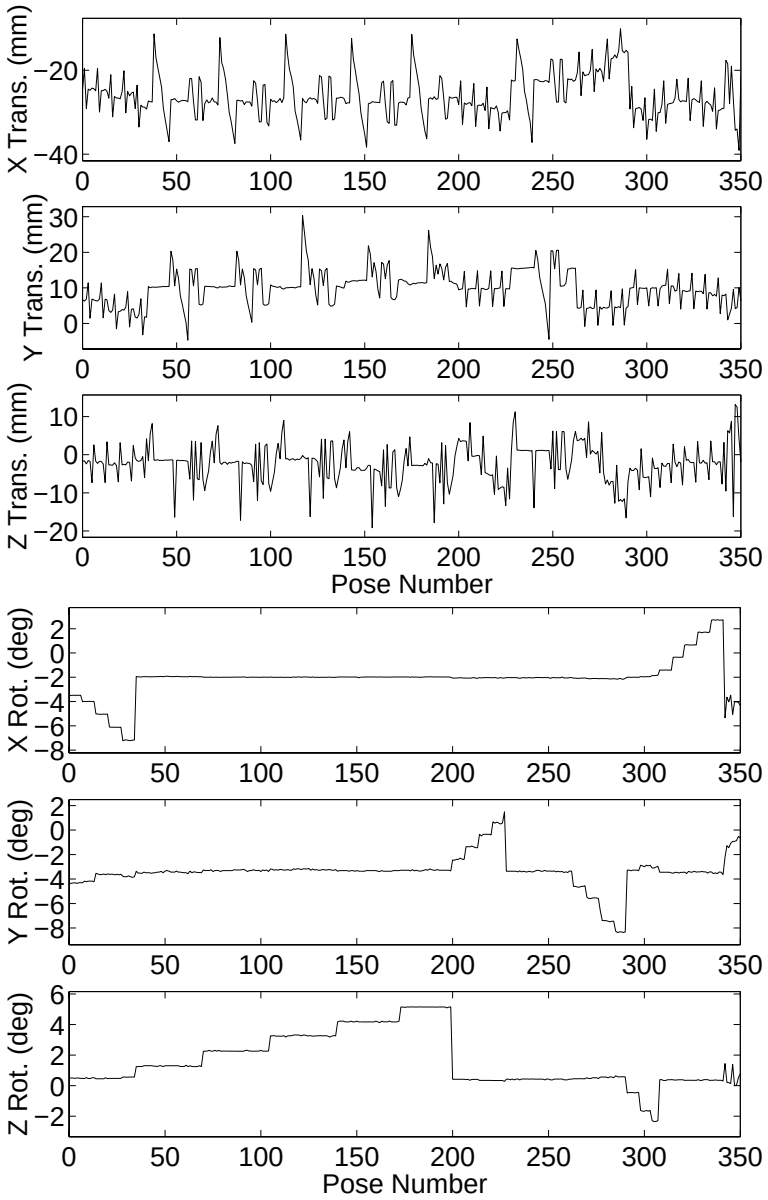



Figure 7.8: Pose parameters returned by the independent ground truth measurement for each pose in the test sequence. The center of rotation is inside the head at a plausible tumor location.

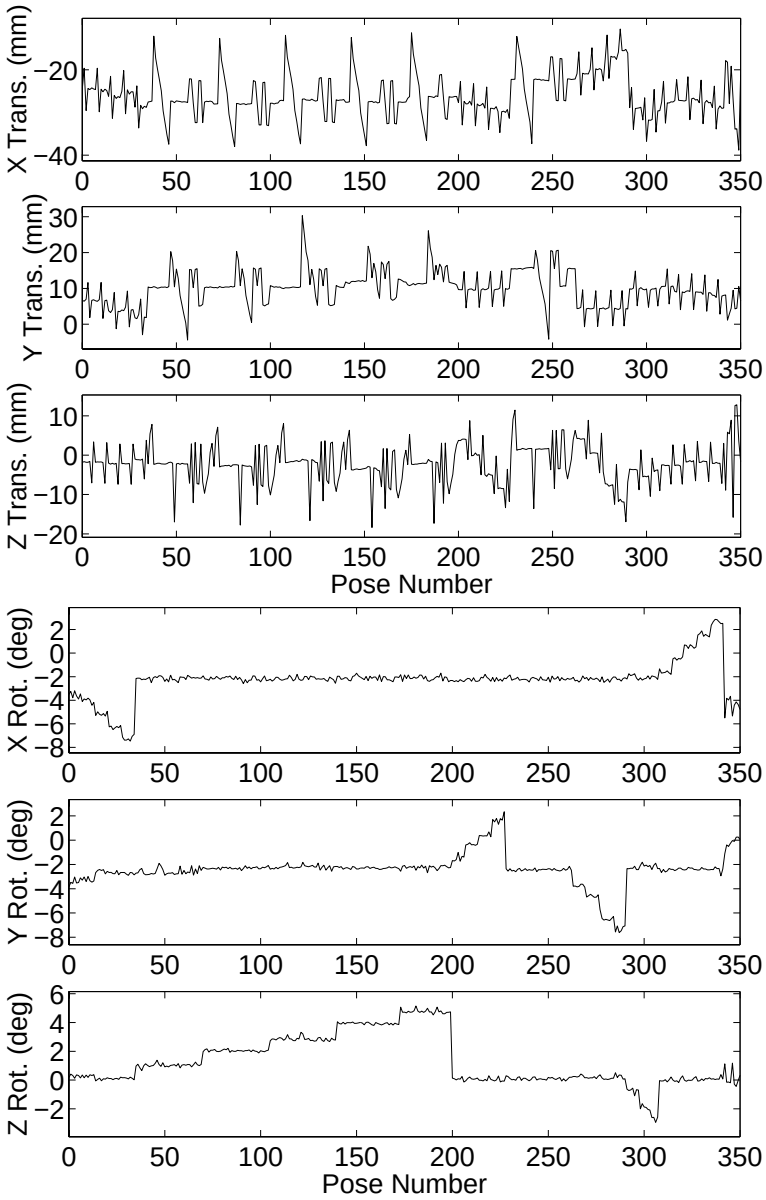



Figure 7.9: Pose parameters returned by the registration algorithm for each pose in the test sequence. The center of rotation is inside the head at a plausible tumor location.

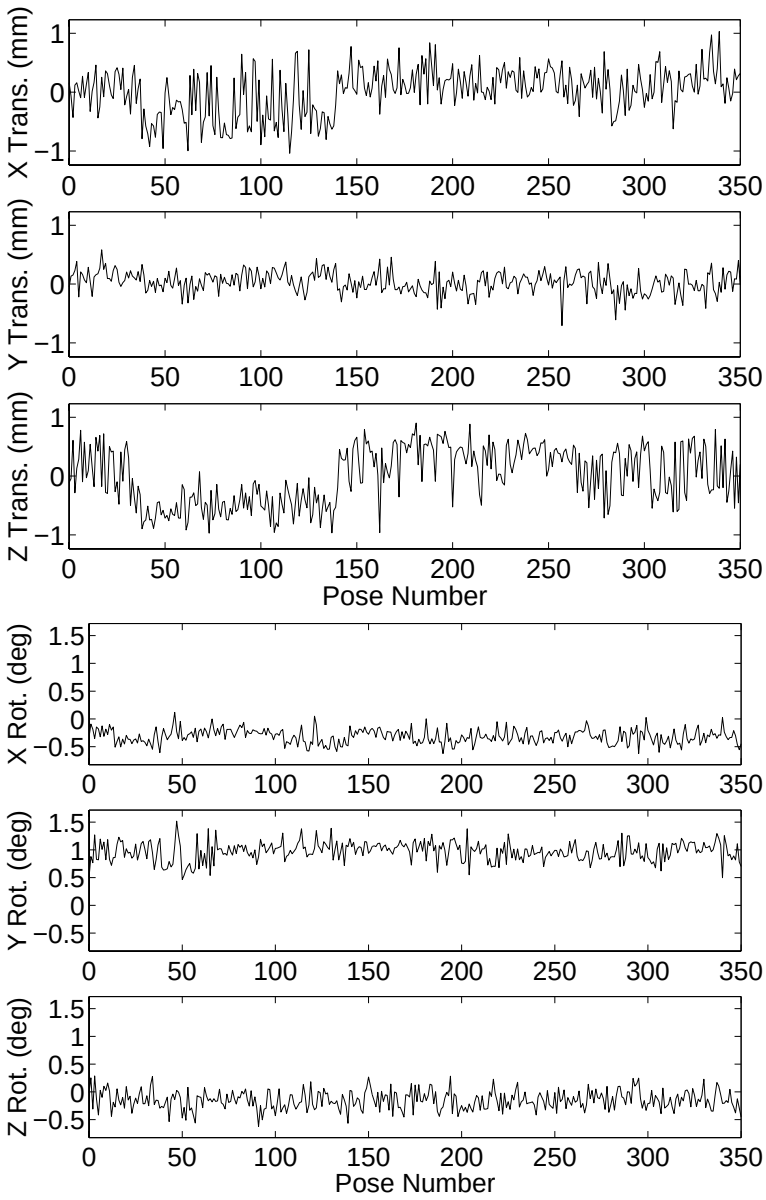



Figure 7.10: Absolute pose parameter error for $[x,\ y,\ z,\ \theta_{\mathrm{x}},\ \theta_{\mathrm{y}},\ \theta_{\mathrm{z}}]$. The center of rotation is inside the head at a plausible tumor location.

mations ${}^{\mathrm{C}}T_{\mathrm{A}}$ and ${}^{\mathrm{B}}T_{\mathrm{W}}$ were generated. These new transforms were chosen using a non-linear optimization routine so as to minimize the difference between the ground truth and the pose estimates for the selected poses. Pose parameter error was recomputed for the entire dataset using the new estimates of ${}^{\mathrm{C}}T_{\mathrm{A}}$ and ${}^{\mathrm{B}}T_{\mathrm{W}}$, as shown in figure 7.11. We call these results a comparison of *relative motion*, since they indicate the extent to which the registration results are consistent with the ground truth. RMS relative translation errors were 0.42 mm, 0.22 mm, and 0.50mm along the X, Y, and Z axes respectively, and RMS rotation errors were $0.17^o$, $0.17^o$, and $0.13^o$ around the X, Y, and Z axes respectively. It is important to stress that accurate relative motion results do not imply accurate registration. It is very possible that the mean errors of 7.3.1 reflect biases in the estimation procedure, rather than biases in the ground truth computation. They do, however indicate the extent to which the measured poses are consistent with the ground truth, and provide an idea of best case accuracy, assuming all biases to be due to errors in estimating ${}^{\mathrm{C}}T_{\mathrm{A}}$ and ${}^{\mathrm{B}}T_{\mathrm{D}}$.

### 7.3.2 Limitations of Pose Parameter Error

It is important to use caution when interpreting registration errors such as those presented in section 7.3.1. Since the actual registration errors vary spatially, the reported x, y, and z translation errors are very much depend on the choice of center of rotation[52]. To see this, consider figure 7.12. Both plots in this figure represents a 6cm x 6cm region of interest drawn from one slice of the CT volume. Each arrow in the plots represents the 3D registration error at one point within the region of interest. The plot on the left has a clear rotational component, while the plot on the right appears to show almost purely translational error. In fact, the errors in the plot on the right can be accurately described as a pure rotation (zero translational error) about a point which lies some distance below and to the left of the graph.

To see why this is important, consider the absolute pose parameter errors shown in figure 7.13. These graphs describe exactly the same registration errors as the graphs in figure 7.10, but now the rotation is expressed around a different point in CT coordinates. With this change, RMS translation errors increase to 1.9 mm, 0.69 mm, and 1.11 mm along the X, Y, and Z axes respectively, while RMS rotation errors remain at $0.22^o$, $0.99^o$, and $0.34^o$ around the X, Y, and Z axes respectively. Clearly, choice of rotation center has a marked effect on the apparent accuracy of the registration when errors are reported in this space.

Although pose parameter error does fully describe 3D registration error, the dependence on rotation center makes it difficult to interpret the results. Pose parameterizations such as $[x,\ y,\ z,\ \theta_{\mathrm{x}},\ \theta_{\mathrm{y}},\ \theta_{\mathrm{z}}]^T$ can be especially confusing, since the pose parameters are expressed in units which have familiar geometric interpretations but have no direct mapping to the actual 3D registration errors. Results based solely on comparison of pose parameters should be used only when their physical meaning

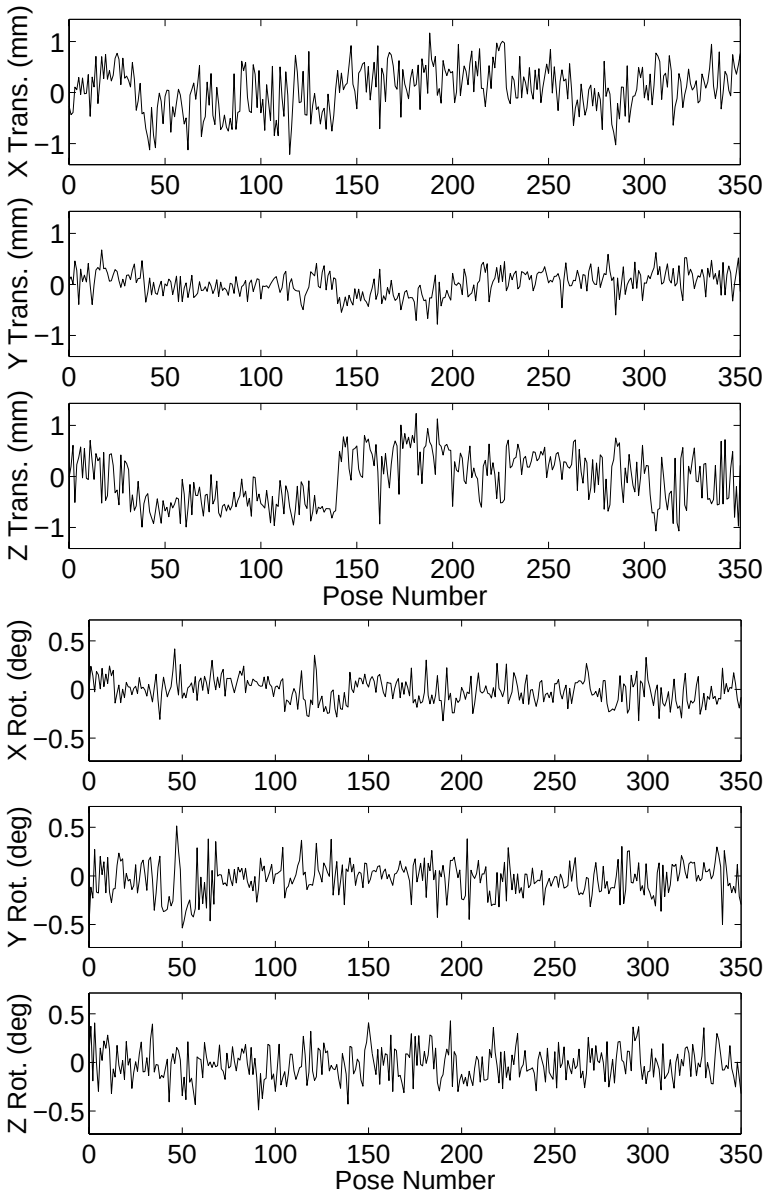



Figure 7.11: Relative pose parameter error for $[x,\ y,\ z,\ \theta_{\mathrm{x}},\ \theta_{\mathrm{y}},\ \theta_{\mathrm{z}}]$. The center of rotation is inside the head at a plausible tumor location.

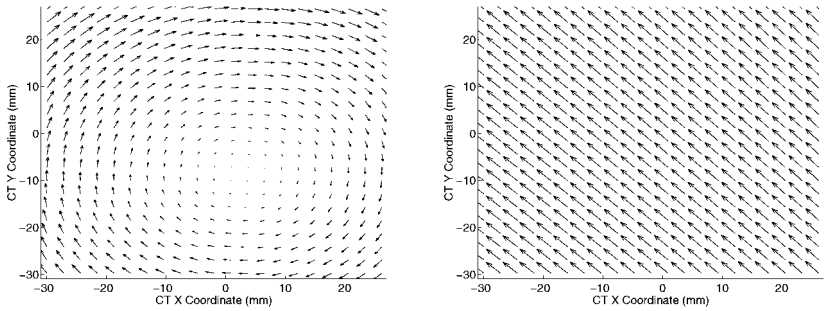



Figure 7.12: Actual registration errors vary spatially within the volume of interest.

is clear, or when followed by a clear geometric interpretation. Accordingly, we present an alternate measurement of registration accuracy.

### 7.3.3 Physically Meaningful Registration Errors

A 6 cm x 6 cm x 6 cm volume of interest was defined, roughly centered in the cranium. Within the volume of interest, 8000 regularly spaced sample points were defined. For each pose, the position of each sample point was calculated with respect to coordinate system W based on the registration result, and again using the ground truth measurement. For each point, the difference between the two calculated positions is a vector in 3D space, which we call the *3D registration error* at that point. We call the magnitude of this vector the *magnitude of registration error* at that point. This measure of registration error is related to the *target registration error* described by Fitzpatrick[15]. It is independent of the choice of rotation center, and provides an intuitive, physically meaningful representation of the registration accuracy. For each pose, both the RMS registration error magnitude and maximum registration error magnitude were computed over the set of sample points. These values are shown in figure 7.14. In addition, the RMS magnitude of absolute registration error was computed over all sample points in all poses, and found to be 1.3 mm. The maximum magnitude of absolute registration error over all sample points in all poses was 3.1 mm in one corner of the (6 cm)$^3$ volume of interest at pose number 336. This pose corresponds to pose parameters $[x, y, z, \theta_x, \theta_y, \theta_z]$ = $[-29.79\,\mathrm{mm},\ 7.75\,\mathrm{mm},\ -1.87\,\mathrm{mm},\ 2.7^o,\ -3.42^o,\ 0.35^o]$, as illustrated in figure 7.8. For comparison of relative motion, the RMS magnitude of registration error over all sample points in all poses was 0.72 mm, and the maximum was 1.66 mm.

To facilitate interpretation of these results, a histogram of registration error magnitude was computed over all of the sample points and all of the measured poses. That is, for each of the 8000 points within the volume of interest, the magnitude of registration error was computed at each of the 352 poses, and all of these values were used to construct a histogram. These histogram values were

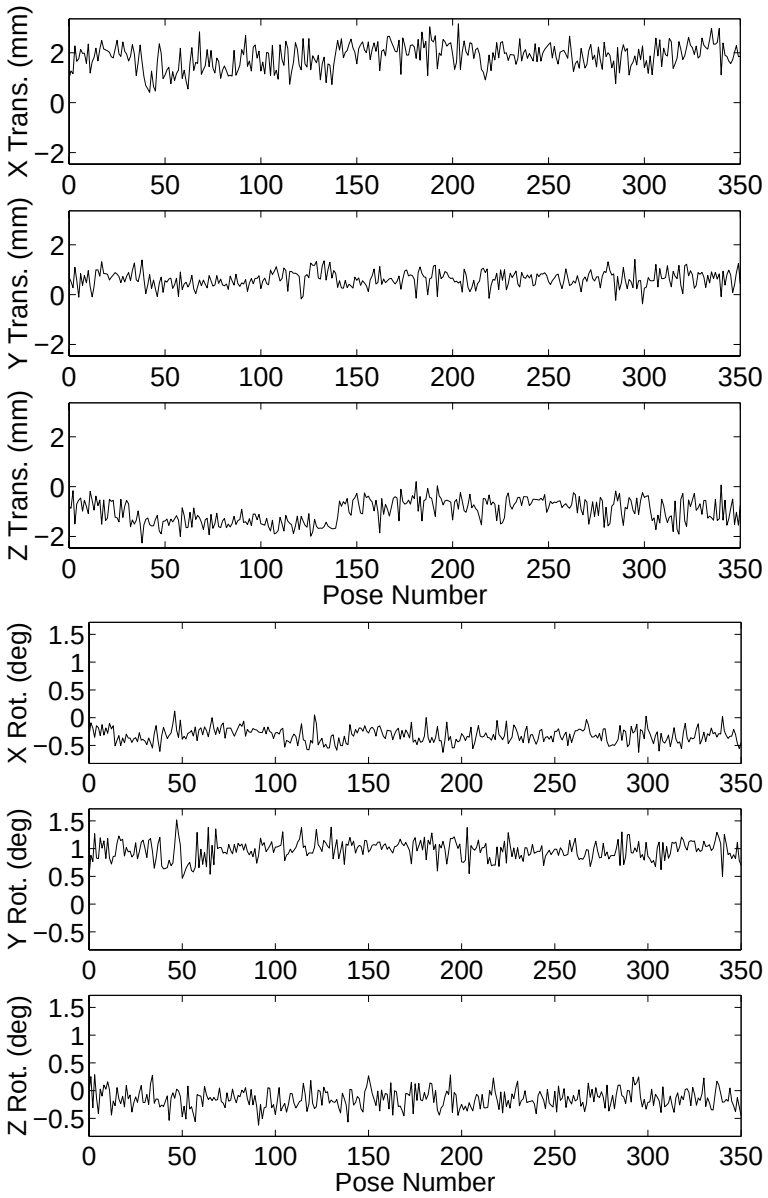



Figure 7.13: These plots show exactly the same errors as those of figure 7.10, with the exception that rotations are now expressed around a different point in the CT volume. Note that the apparent translation error is dramatically increased.

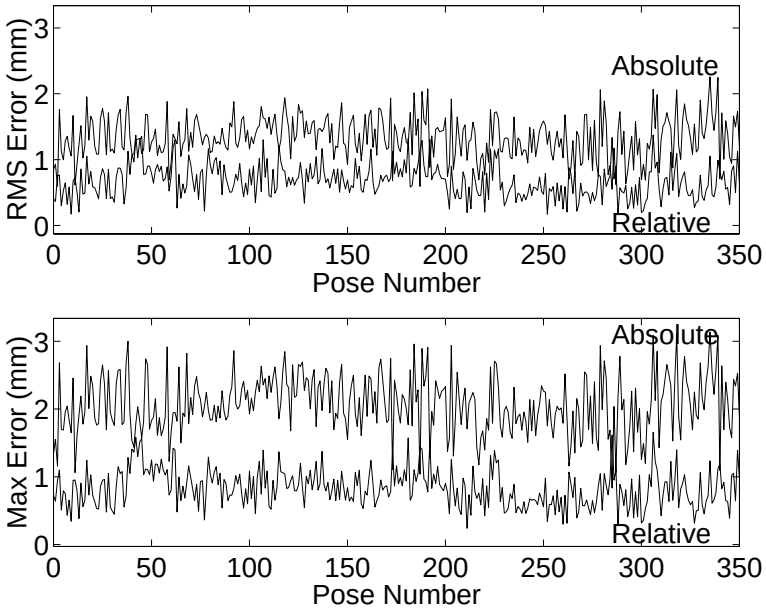
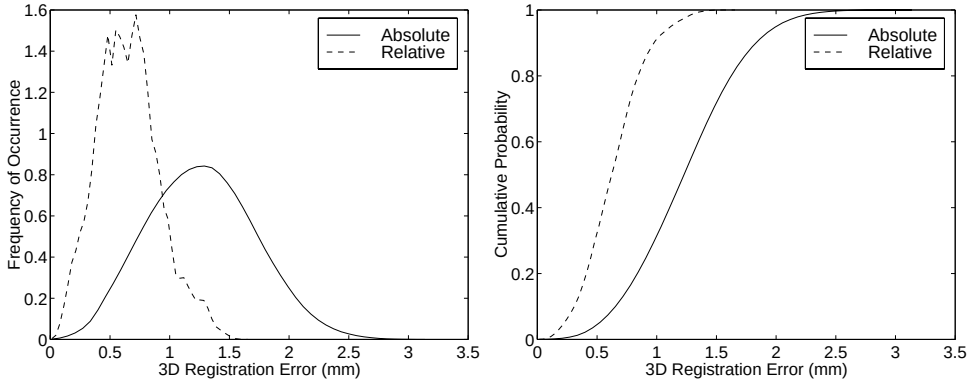



Figure 7.14: These graphs show the RMS and Maximum registration errors over a 6cm$^3$ volume centered in the cranium. Each plot has two lines: the absolute error measurement, which includes errors in estimating coordinate transforms ${}^{\mathrm{C}}T_{\mathrm{A}}$ and ${}^{\mathrm{B}}T_{\mathrm{W}}$; and the relative error measurement, which estimates these transforms based on the registration data.




(a) (b)

Figure 7.15: (a) Normalized histograms of registration error magnitude for both relative and absolute motion comparison. These histograms are computed over all of the 8000 target points and all of the 352 correctly converged test poses. (b) Corresponding cumulative distribution functions.

rescaled to approximate the probability distribution of the registration error magnitude, as shown in figure 7.15(a), and reformatted as a cumulative distribution function (CDF) as shown in figure 7.15(b). From this CDF , we can estimate the probability that the magnitude of registration errors at an arbitrary point within the volume of interest will be below a particular threshold. For example, we estimate that absolute registration error at an arbitrary point will be below 1.5 mm with probability 0.72, and below 2 mm with probability 0.95. Similarly, we estimate that relative registration error will be below 1.0 mm with probability 0.91 and below 1.3 mm with probability 0.99.

## 7.4 Discussion

This chapter describes an anthropomorphic phantom study evaluating our X-ray/CT registrations system for application to image guided frameless stereotaxy. Evaluating 3D registration errors within 6 cm x 6 cm x 6 cm volume of interest inside the phantom cranium, and over 352 distinct poses, we find an RMS error magnitude of 1.3 mm. The actual error measurements vary considerably, with an observed maximum of 3.1 mm in one instance.

Some of the registration error is doubtless due to specific inaccuracies in the ground truth computation. In particular, we hypothesize that more accurate measurement of two specific coordinate transformations would reduce the size of the observed registration errors. We establish an upper bound on the magnitude of this reduction by computing a second set of *relative* results, and show that re-measurement of the two coordinate transforms would at best reduce RMS error magnitude

to 0.72 mm, and maximum error magnitude to 1.66 mm.

Finally, we show that the choice of error space profoundly affects the apparent registration accuracy, and argue for the use of error measurements with clear geometric interpretations.

# Chapter 8

# Post-operative Measurement of Acetabular Cup Position

The importance of postoperative feedback in computer assisted orthopedic surgery has increased dramatically with the introduction of computer assisted preoperative planning and treatment delivery methods[10]. In this chapter, we discuss the application of the X-ray/CT registration system to measurement of acetabular cup orientation following total hip replacement surgery. Postoperative measurement of implant orientation is important for several reasons: implant orientation has been shown to be predictive of postoperative outcome, and of complications such as dislocation[38][41]; accurate feedback permits the surgeon to refine his or her technique, in order to more effectively follow the preoperative plan; and reliable measurements allow information from follow-up studies to be correlated with the actual implant placement, ultimately leading to improved presurgical planning.

Postoperative measurements of acetabular cup orientation must be completely non-invasive. This makes it difficult to accurately estimate cup placement with respect to the pelvis. Implant orientation is traditionally measured using anterior-posterior (AP) X-ray images, and the resulting measurements have typically have very high variance [27]. Despite recent improvements in computer-assisted measurement technique, state of the art measurements still have large ($> 7^o$) error margins[28], in large part due to unknown pelvic flexion at the time of X-ray acquisition.

In our system we begin by using X-ray/CT registration to explicitly recover the pose of the pelvis, thus removing a major source of measurement error. This pose is expressed as a rigid transformation between the imager coordinate system and the coordinate system of the preoperative CT volume. Next, the pose of the implant is recovered with respect to the coordinate system of the CT volume using a projection-based algorithm similar to that of Sarojak [48]. After both of these registrations are complete, an anatomically based pelvis coordinate system is defined through the use

of stable anatomical landmarks. The pose of the implant is transformed into this coordinate system in order to provide clinically meaningful results.

We present a phantom studies in which pose estimates are generated using images of a high-density pelvis phantom. These estimates are compared with ground truth results obtained using the HipNav image guided system for total hip replacement surgery[10], and show RMS measurement errors on the order of $2^o$.

## 8.1 Problem Description

During postoperative evaluation of implant position, the pose of the acetabular implant must be determined with respect to the pelvis. AP and lateral X-ray images are acquired with known source-to-film distances. The approximate projection centers of each X-ray image are known, but the pose of the patient is not. In particular, the images are not known to be true AP or lateral views, and each image may be acquired with the patient either lying or standing. In addition to the X-ray images, a preoperative pelvic CT volume and a triangulated surface model of the implanted cup are available.

Although the AP and lateral X-ray images are not acquired simultaneously, it is assumed that the position of the cup with respect to the pelvis does not change between acquisitions. A schematic drawing of this scenario is shown in figure 8.1. There are three unknown rigid body transformations in this figure: the transformation from the coordinate system of the acetabular cup to the CT coordinate system, which is labeled ${}^{\mathrm{ct}}T_{\mathrm{cup}}$; the transformation from the CT coordinate system to the coordinate system of the X-ray imager at the time of AP image acquisition, labeled ${}^{\mathrm{S_0}}T_{\mathrm{ct}}$; and the transformation from the CT coordinate system to the coordinate system of the X-ray imager at the time of lateral image acquisition, labeled ${}^{\mathrm{S_1}}T_{\mathrm{ct}}$. We describe each of these coordinate transformations using the seven-element parameterization of section 2.1.2, for a total of 21 unknown parameters.

## 8.2 Approach

The problem is broken down into four steps:

1. All parameters are initialized to reasonable starting values using input from the user.

2. The pose of the pelvis is estimated with respect to each imager using the iterative 2D/3D registration scheme described in chapter 2.

3. The pose of the acetabular implant is estimated with respect to the coordinate system of the CT volume by simultaneously matching the projection of of the implant surface model to

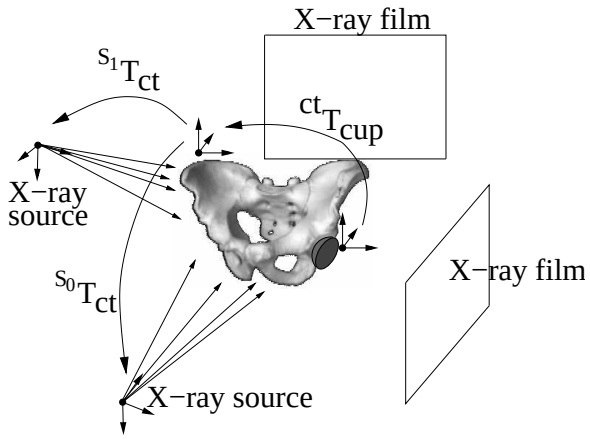



Figure 8.1: The pose of the acetabular implant is measured with respect to the pelvis using a pair of X-ray images. The position of each X-ray source at the time of image acquisition is known only approximately.

contours in each X-ray image.

4. The anatomically based pelvis coordinate system is defined by manually selecting landmarks in the CT volume, and the result from step 3 is further transformed so that the pose of the acetabular cup is represented in this coordinate system.

These four steps are discussed in sections 8.2.1, 8.2.2, 8.2.3, and 8.2.4 respectively.

### 8.2.1 Initialization

The 21 unknown degrees of freedom are initialized to reasonable starting values by user input. To initialize the coordinate transformations ${}^{\mathrm{S}_0}T_{\mathrm{ct}}$ and ${}^{\mathrm{S}_1}T_{\mathrm{ct}}$, the user indicates the approximate positions of at least three anatomical landmarks in each radiograph, and then enters the corresponding 3D coordinates from the CT volume. The initial transformations are then estimated using a point based 2D/3D registration.

The initial pose of the acetabular cup implant with respect to the CT volume is approximately known from the preoperative plan. In addition, the user provides the image positions of several points on the border of the cup in each image as shown in figure 8.2. These points are used to further constrain the position of the cup as described in section 8.2.3.

### 8.2.2 X-ray/CT registration

The pose of the pelvis with respect to the X-ray imaging apparatus is recovered for each image by iterative comparison between the input images and Digitally Reconstructed Radiographs (DRRs).

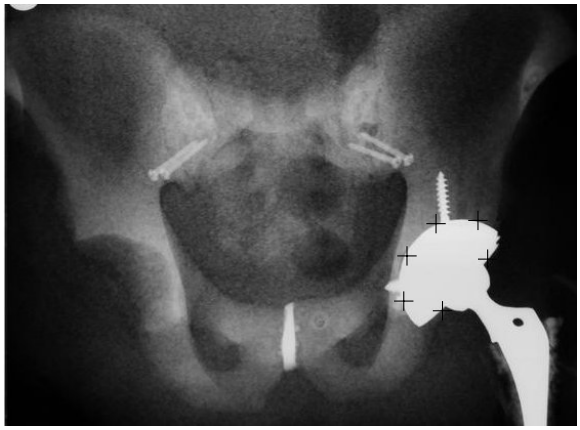


Figure 8.2: The user clicks several points on the boundary of the acetabular cup to initialize the contour-based registration process.

The DRRs are generated using the Transgraph-based technique described in chapter 3, and the images are compared using the VLNC correlation measure of section 2.2.3. The objective function of equation 2.32 is minimized using the quasi-Newton algorithm of Broyden, Fletcher, Goldfarb and Shanno, as described in [45]. The minimization is run once for each input image, to estimate the coordinate transformations ${}^{\mathrm{S}_0}T_{\mathrm{ct}}$ and ${}^{S_1}T_{\mathrm{ct}}$.

### 8.2.3 Determination of Cup Position

Once the pose of the pelvis has been estimated in both images, the pose of the acetabular cup implant is estimated with respect to the coordinate system of the CT volume. This is done by simultaneously matching the projection of the implant surface model to contours in the two X-ray images.

For a given pose, the silhouette of the surface model is projected into each image as a collection of points, and an error measure is computed based on the image positions of the projected points. This error measure is then minimized over the parameter space of the rigid body transformation ${}^{\mathrm{ct}}T_{\mathrm{cup}}$. The first subsection below describes how the silhouette is computed, the second describes a rough point-based registration used to approximate the actual cup position, and the third describes a final minimization which increases the registration accuracy.

#### Silhouette Generation.

To generate the silhouette of the cup, the vertices of the cup surface model are projected into each image. The projected vertices define a set of 2D triangles corresponding to the 3D triangles of the surface model. The silhouette is generated by culling those vertices which lie interior to any of the projected triangles. To speed this culling process, the projected triangles are organized into a

quadtree data structure, and each vertex is compared against only those triangles which lie in or intersect its cell in the quadtree. We denote the set of points which make up the silhouette of the cup in the AP image as $H_0$. Similarly, we denote the set of points which make up the silhouette of the cup in the lateral image as $H_1$.

**Approximate Solution.**

The pose of the cup is initially computed based on the image coordinates supplied by the user during manual initialization. We define the objective function

$$f(\gamma) = \frac{1}{|R|} \sum_{\boldsymbol{r} \in R} \min_{\boldsymbol{t} \in H_0(\gamma)} (\|\boldsymbol{r} - \boldsymbol{t}\|) + \frac{1}{|S|} \sum_{\boldsymbol{s} \in S} \min_{\boldsymbol{u} \in H_1(\gamma)} (\|\boldsymbol{s} - \boldsymbol{u}\|), \tag{8.1}$$

where $\gamma$ is the vector of parameters describing the rigid body transformation ${}^{\mathrm{ct}}T_{\mathrm{cup}}$, $R$ is the set of user-supplied initialization points in the AP image, and $|R|$ is the number of points in this set. $S$ is the set of user-supplied initialization points in the lateral image, and $|S|$ is the number of points in this set. $H_0(\gamma)$ is the set of points comprising the silhouette of the cup in the AP image at the pose specified by $\gamma$, and $H_1(\gamma)$ is the set of points comprising the silhouette in the lateral image. The notation $\|x\|$ denotes the magnitude of vector $x$.

The objective function in equation 8.1 reaches a minimum of zero when every user supplied point is exactly overlapped by one of the points on the silhouette of the projected model. In practice, this minimum is nearly met when the boundaries of the projected surface model lie close to the edges of the cup in the X-ray images. The objective function is minimized using the downhill simplex method of Nelder and Mead, as described in [45].

**Refinement of Approximate Solution.**

The minimization above gives a good approximation to the pose of the acetabular cup. There are, however, small inaccuracies. This is because the initialization points may not lie exactly on the boundaries of the implant in the X-ray images, and because these points may not match well with the points which make up the silhouette of the projected cup. Therefore we use this estimate only to initialize a more precise search.

The gradient of pixel intensity with respect to pixel coordinates $u$ and $v$ is computed for each

input image, and an objective function is defined

$$\begin{aligned} g(\gamma) &= G_0 - \frac{1}{|H_0(\gamma)|} \sum_{\boldsymbol{t} \in H_0(\gamma)} \left\| \left[ \frac{\partial}{\partial u} U_0(\boldsymbol{t}), \frac{\partial}{\partial v} U_0(\boldsymbol{t}) \right]^T \right\| \\ &\quad + G_1 - \frac{1}{|H_1(\gamma)|} \sum_{\boldsymbol{t} \in H_1(\gamma)} \left\| \left[ \frac{\partial}{\partial u} U_1(\boldsymbol{t}), \frac{\partial}{\partial v} U_1(\boldsymbol{t}) \right]^T \right\|, \end{aligned} \tag{8.2}$$

where $\gamma$ is the vector of parameters describing ${}^{\mathrm{ct}}T_{\mathrm{cup}}$, $|H_0(\gamma)|$ is the number of points in the cup silhouette projected into the AP image, and $|H_1(\gamma)|$ is the number of points in the cup silhouette projected into the lateral image, $\frac{\partial}{\partial u} U_i(\boldsymbol{t})$ is the first derivative of image $U_i$ with respect to image coordinate $u$, evaluated at image point $\boldsymbol{t}$, and $\frac{\partial}{\partial v} U_i(\boldsymbol{t})$ is the first derivative of image $U_i$ with respect to image coordinate $v$, evaluated at image point $\boldsymbol{t}$. $G_0$ and $G_1$ are chosen so that the value of $g(\gamma)$ is guaranteed to be non-negative

$$G_i = \max_{\boldsymbol{t}} \left( \left\| \left[ \frac{\partial}{\partial u} U_i(\boldsymbol{t}), \frac{\partial}{\partial v} U_i(\boldsymbol{t}) \right]^T \right\| \right), \tag{8.3}$$

where the maximum is taken over all of the pixels in $U_i$.

The objective function of equation 8.2 reaches a minimum when the points of the silhouette lie in high gradient regions of the image. Since the image contours of the acetabular implant have very high gradient, $g(\gamma)$ decreases as the silhouette becomes more closely aligned with the contours of the acetabular cup. As before, this function is minimized using the downhill simplex method of Nelder and Mead.

### 8.2.4 Pelvis Coordinate System

The pelvis coordinate system used by the HipNav system is defined with respect to several points on the pelvis. This coordinate system is different from the CT coordinate system in which we have located the acetabular implant. We define a 4x4 coordinate transformation ${}^{\mathrm{ct}}T_{\mathrm{p}}$ which transforms coordinates in the pelvis coordinate system to the CT coordinate system.

The anatomical landmarks used to define the pelvis coordinate system are shown in figure 8.3. The two pubic symphisys points and the locations of the two anterior iliac spines are identified by inspection in the CT volume. Point A is found by simply averaging the two pubic symphisys points. We define direction vectors $e_x$, $e_{\mathrm{y}}$, and $e_{\mathrm{z}}$ in the CT coordinate system which point in the same

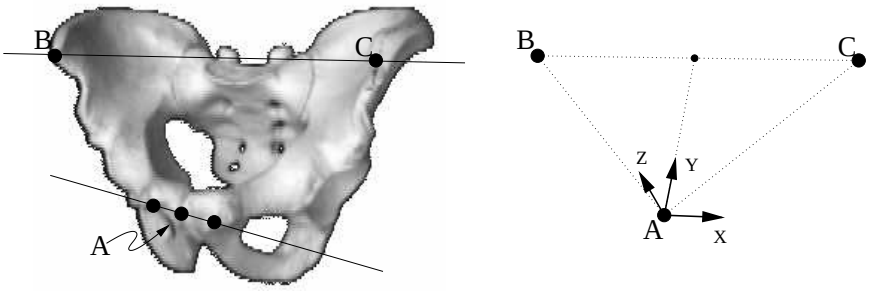



Figure 8.3: The pelvis coordinate system is defined relative to four anatomical landmarks. The Origin of the coordinate system lies at a point midway between the two pubic symphises. This point is labeled A in the figure. The X axis of the pelvis coordinate system is parallel to the line connecting the right and left iliac spines, which are labeled B and C. The Y axis lies in the plane of the points A, B, and C.

directions as the pelvis X, Y, and Z axes. These three vectors are simply defined as

$$e_{\mathrm{x}} = \frac{C - B}{\|C - B\|} \tag{8.4}$$

$$e_{\mathrm{y}} = \frac{(C - A) - ((C - A) \cdot e_{\mathrm{x}})\, e_{\mathrm{x}}}{\|(C - A) - ((C - A) \cdot e_{\mathrm{x}})\, e_{\mathrm{x}}\|} \tag{8.5}$$

$$e_{\mathrm{z}} = e_x \times e_{\mathrm{y}}. \tag{8.6}$$

The expression for ${}^{\mathrm{ct}}T_{\mathrm{p}}$ follows directly

$${}^{\mathrm{ct}}T_{\mathrm{p}} = \begin{bmatrix} e_{\mathrm{x}} & e_{\mathrm{y}} & e_{\mathrm{z}} & A \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{8.7}$$

and the pose of the cup with respect to the pelvis coordinate system is found using this transform

$${}^{\mathrm{p}}T_{\mathrm{cup}} = \left({}^{\mathrm{ct}}T_{\mathrm{p}}^{-1}\right)\left({}^{\mathrm{ct}}T_{\mathrm{cup}}\right). \tag{8.8}$$

## 8.3 Experiment

A preliminary study was conducted using phantom data to evaluate the performance of the registration algorithm. A 62mm diameter VerSys Acetabular cup (Zimmer, Inc.) was fitted to a high density pelvis phantom. A CT dataset was acquired, having an intra-slice pixel spacing of approximately 0.74mm, a slice thickness of 1mm, and an inter-slice spacing of 1mm.

The pelvis coordinate system was defined with respect to the left and right anterior iliac spines, and the left and right pubic symphisis points as described in section 8.2.4. These points were marked

on the model with 1 mm diameter fiducial markers. The fiducials were manually identified in the CT and used to compute the coordinate transformation between the pelvis coordinate system and the CT coordinate system.

Three series of images were acquired. In each series, AP films were taken with an approximate source-to-film distance of 40 inches, while lateral films were taken with an approximate source-to-film distance of 72 inches. In order to prevent the 1 mm fiducial markers from biasing the X-ray/CT registration, an image manipulation program was used to paint out any visible markers in the X-ray images.

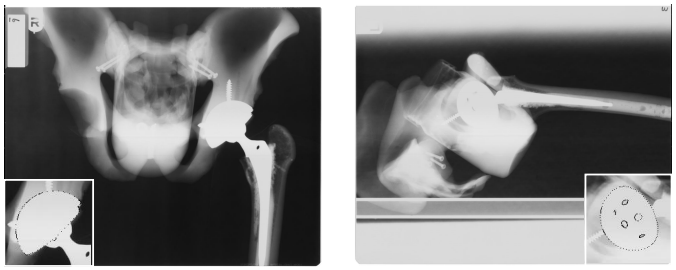
The first image series involved 3 AP images and 3 lateral images. Each image was acquired from a slightly different angle, and in four of the six images, household objects were placed in the field of view in order to simulate occluding patient anatomy as shown in Fig. 8.4(a). For the second and third series of images, a simulated torso, surrounding the pelvis, was constructed out of plastic film and filled with oats to simulate soft tissue. Small balloons were inserted into the oats to simulate the effects of bowel gas and soft tissue inhomogeneity. These two series differ in the arrangement of the soft tissue and in the placement of the bowel gas. In addition, the acetabular cup was removed prior to acquisition of the third series, and re-attached in a different orientation. As before, each image was taken from a slightly different angle. The AP images spanned roughly $20^o$ in flexion, while the lateral images spanned a range of roughly $15^o$ rotation around the superior-inferior axis. Typical images from the these series are shown in figures 8.4(b).

Ground truth measurements were performed using the HipNav system [10]. Repeated measurements of the implant orientation for the first two series had a mean abduction of $45.2^o$ and a mean flexion of $10.4^o$, with standard deviations of $0.11^o$ and $0.22^o$ respectively. After repositioning for the third series, the cup was measured to have an orientation of $52.6^o$ abduction and $48.9^o$ flexion.

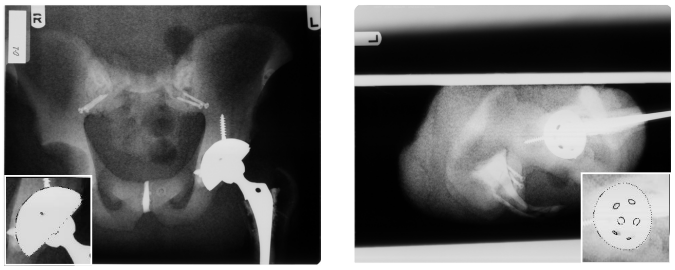
All of the films were digitized and resampled to resolution of 36 dpi, giving a final image size of 621x512 pixels. The center of projection for each image was assumed to lie at pixel coordinates (310, 255.5). No further attempt was made to calibrate the X-ray imaging system. The images from each series were grouped into pairs, each consisting of one AP image and one lateral image and the registration algorithm was run using each pair as input to recover the pose of the acetabular cup implant with respect to the coordinate system of the CT. This transformation was composed with the CT-to-pelvis coordinate transformation, and flexion/abduction measurements were calculated.

## 8.4 Results

Abduction and flexion measurements are presented in table 8.1. The recovered cup orientation matches the measured ground truth to within $2^o$ abduction and $3^o$ flexion in all except one of the trial cases. This incorrectly converged case is labeled 7(a) in the table, and is discussed in the



(a)



(b)

Figure 8.4: (a) A pair of input images from the first series of radiographs. The inset shows recovered cup position, and a peanut butter jar is visible in each image. (b) A pair of input images from the second series of radiographs, showing simulated soft tissue. In the lateral image, the superior boundary of the simulated torso runs almost parallel to the superior edges of the iliac crests. The bright line running superior-inferior in this image is a lexan plate to which the pelvis is attached.

| | Ground Truth | | Pose Estimate | | Error | |
|---|---|---|---|---|---|---|
| | Abduction | Flexion | Abduction | Flexion | Abduction | Flexion |
| Series 1 | | | | | | |
| Case 1 | 45.2$^o$ | 10.4$^o$ | 45.3$^o$ | 13.0$^o$ | 0.1$^o$ | 2.6$^o$ |
| Case 2 | 45.2$^o$ | 10.4$^o$ | 45.6$^o$ | 13.1$^o$ | 0.4$^o$ | 2.7$^o$ |
| Case 3 | 45.2$^o$ | 10.4$^o$ | 45.2$^o$ | 12.3$^o$ | 0.0$^o$ | 1.9$^o$ |
| Series 2 | | | | | | |
| Case 4 | 45.2$^o$ | 10.4$^o$ | 46.2$^o$ | 11.9$^o$ | 1.0$^o$ | 1.5$^o$ |
| Case 5 | 45.2$^o$ | 10.4$^o$ | 45.4$^o$ | 10.3$^o$ | 0.2$^o$ | -0.1$^o$ |
| Case 6 | 45.2$^o$ | 10.4$^o$ | 45.8$^o$ | 10.8$^o$ | 0.6$^o$ | 0.4$^o$ |
| Series 3 | | | | | | |
| Case 7(a) | 52.6$^o$ | 48.9$^o$ | 59.3$^o$ | -10.7$^o$ | 6.7$^o$ | -59.6$^o$ |
| Case 7(b) | 52.6$^o$ | 48.9$^o$ | 50.7$^o$ | 50.5$^o$ | -1.9$^o$ | 1.6$^o$ |
| Case 8 | 52.6$^o$ | 48.9$^o$ | 52.6$^o$ | 47.0$^o$ | 0.0$^o$ | -1.9$^o$ |
| Case 9 | 52.6$^o$ | 48.9$^o$ | 50.8$^o$ | 50.2$^o$ | -1.8$^o$ | 1.3$^o$ |

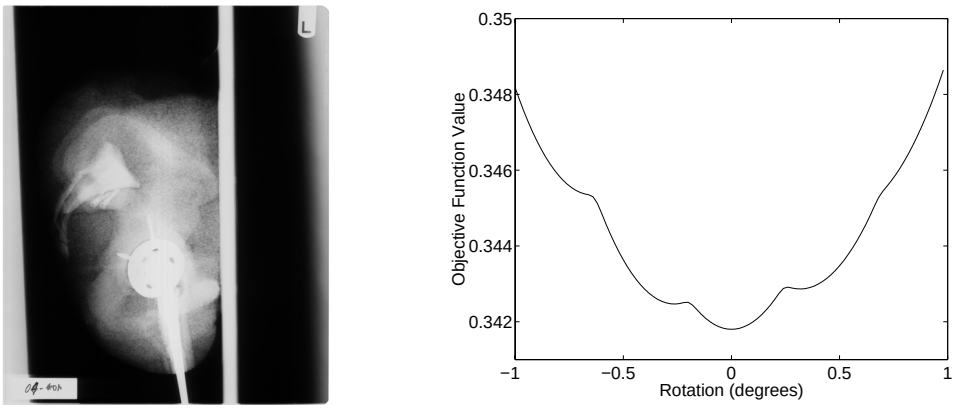
Table 8.1: Registration results for measurement of acetabular implant orientation. Erratic convergence in case 7(a) was resolved by using a lateral image having an oblique component (case 7(b)) as discussed in the text.

following paragraphs. RMS error among the correctly converged cases was 0.96$^o$ abduction and 1.76$^o$ flexion.

Although the X-ray/CT registration algorithm converges reliably for AP images, convergence was erratic for the lateral image in case 7(a), which shows large errors in estimated cup orientation. We attribute this erratic convergence to the bilateral symmetry of the pelvis, which leads to pose ambiguity when the view direction is very nearly lateral. The image in question was very nearly a direct lateral shot, and is shown in figure 8.5(a).

Since the pelvis is bilaterally symmetric, the projections of the left and right sides of the pelvis look similar. In a direct lateral image, these similar projections are very close to one another. This makes registration more difficult, since features on one half of the pelvis may match well with the image of the other half of the pelvis. If the spurious matches are good enough, they can lead to incorrect local minima in the image comparison function. To illustrate these local minima, a rotation axis was defined running vertically, inferior to superior through the center of the pelvis. A series of objective function values were computed in the neighborhood of the global minimum by rotating around the vertical axis as shown in figure 8.6. Figure 8.5(b) shows the resulting plot and local minima.

The erratic convergence was rectified by replacing the direct lateral image in case 7(a) with a slightly oblique view. The new image is shown in figure 8.7(a). The oblique component of the view makes the two sides of the pelvis project to positions which are offset from one another. Conse-




(a) (b)

Figure 8.5: In a true lateral image (a) the left and right halves of the pelvis project in such a way that similar features from the two sides are very close together. This similarity leads to local minima during registration, as features from the left and right sides are easily confused with one another. These local minima are seen by plotting the value of the objective function (b) while rotating the pelvis pose estimate as illustrated in figure 8.6. The vertical while line in (a) is an edge-on view of the lexan sheet to which the pelvis was mounted after CT acquisition. The white cloud and inhomogeneities surrounding the pelvis are simulated soft tissue.

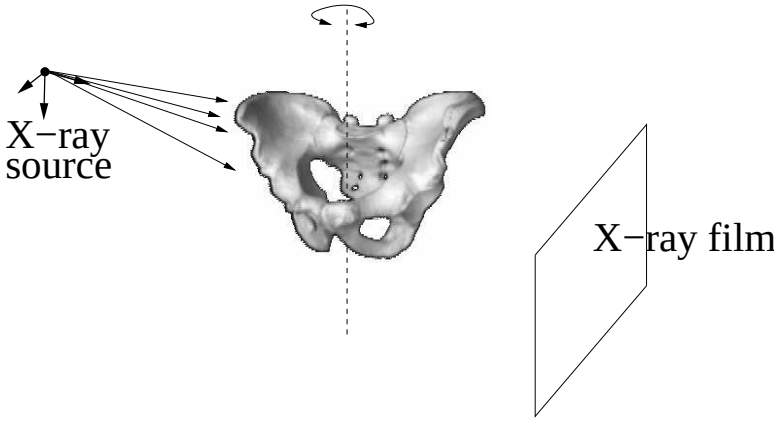
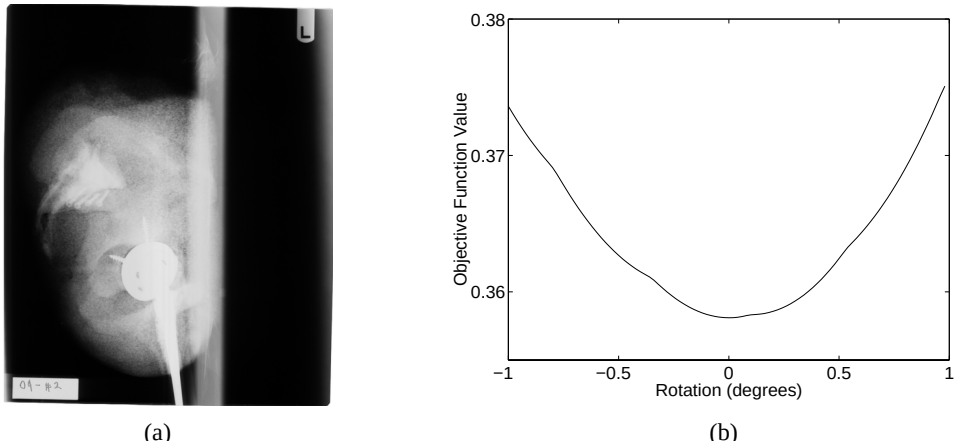



Figure 8.6: The pose estimate was rotated around a vertical axis running through the center of the pelvis. Objective function values were computed in the neighborhood of the global minimum, and are plotted in figure 8.5(b) for a true lateral image, and in figure 8.7(b) for a lateral image with a significant oblique component.




(a) (b)

Figure 8.7: Lateral images which have an oblique component (a) are much less vulnerable to pose ambiguity due to bilateral symmetry. The objective function value (b) is much more well behaved than true lateral images.

quently, spurious feature matches are less common, and the objective function is more well behaved, as shown in figure 8.7(b). Convergence with this new image is reliable, and revised registration errors are shown in case 7(b) of table 8.1.

## 8.5 Discussion

This chapter presents a registration procedure for recovering the orientation of the acetabular cup implant with respect to the pelvis following total hip replacement surgery. The algorithm is tested using data from a phantom study, giving registration results improve on the current state of the art by a factor of more than 3. The data further suggest that registration convergence is most reliable when lateral images are acquired with a slightly oblique component. We anticipate that these results will extend to experiments using real patient data, and such a study is currently underway.

# Chapter 9

# Conclusion

This chapter summarizes the work presented in the thesis, lists contributions, and suggests directions for continued work.

## 9.1 Summary

This thesis presents an algorithm for registering 2D X-ray images with 3D CT data. This algorithm was developed in the context of patient pose estimation for computer assisted medical applications. Our approach is to iteratively refine an estimate of patient pose based on comparison between the 2D X-ray images synthetic X-ray images known as DRRs. The DRRs are computed based on preoperative CT data, and the patient pose is updated until the DRRs and input images are maximally similar.

Image comparison is performed using two different metrics, which we call *sum of local normalized correlation* and *variance-weighted sum of local normalized correlation.* Both of these metrics are computed efficiently using recursive filters, and are robust to noise, inhomogeneity and clutter in the input images.

The most computationally expensive step of the iterative registration process is DRR generation. This is essentially a computer graphics operation in which the 3D CT volume must be used to generate simulated X-ray images. Two methods for accelerated volume rendering are presented. The first method uses no graphics hardware, speeds up DRR generation by over an order of magnitude, and permits efficient differentiation of DRR pixel intensity with respect to patient pose parameters. The second method affords nearly another order of magnitude speedup using consumer grade computer graphics hardware.

Phantom studies are presented which evaluate the registration algorithm for application to image guided stereotactic radiosurgery and post-operative measurement of acetabular implant orientation.

Frameless stereotactic registration results are comparable in accuracy to current systems which use immobilization devices, while the accuracy of implant orientation measurement significantly improves on the current state of the art.

## 9.2 Contributions

This work constitutes a concrete contribution to the literature in several respects:

1. We present an extension of image based rendering techniques to transmission imaging, enabling accelerated computation of DRRs using software based methods. Using this method, 256 pixel by 256 pixel DRRs can be computed based on arbitrarily large CT volumes at speeds of roughly 5Hz. This method further enables the efficient computation of pixel intensity derivatives with respect to patient pose parameters. These derivatives in turn greatly improve the convergence properties of our 2D/3D registration algorithm.

2. We present a new accumulation algorithm which runs on inexpensive, commonly available graphics hardware, and permits hardware accelerated rendering of DRRs. Using this method, full sized (512 pixel by 512 pixel) DRRs can be computed based on large (256x256x256) volumes at speeds of roughly 14 Hz, and smaller 256 pixel by 256 pixel DRRs can be generated at speeds of roughly 40 Hz.

3. We derive a two image comparison metrics, called *sum of local normalized correlation* and *variance-weighted sum of local normalized correlation* which permit accurate image comparison even with noisy radiographs which contain clutter and inhomogeneity.

4. We develop calibration models for geometric and intensity distortions in one class of fixed fluoroscopic imager, and present algorithms for recovering imager calibration parameters.

5. We develop a method for principled comparison of registration results with independently measured ground truth. We present registration results of sufficient accuracy for application to frameless stereotactic radiosurgery.

6. We apply our registration system to post-operative measurement of acetabular implant position, and demonstrate results in phantom studies which improve on the state of the art by a factor of more than two.

## 9.3 Future Work

The work in this thesis suggests several extension, which we address individually:

**Object ordered Transgraph indexing:** The Transgraph implementation presented in chapter 3 is organized in an intuitive and straightforward way. Recovering data from the Transgraph incurs an indexing overhead of 5 multiplications and 20 additions for each pixel lookup, plus 15 multiplications and 30 additions for bilinear interpolation. A lot of this overhead can be eliminated by first computing an intermediate image aligned with the $C_0$ Transgraph coordinate plane, and then projecting this image onto the imaging surface. This strategy is simply an extension of the shear-warp factorization [37] [32] to image based rendering. An additional shortcoming of the current implementation is that the data organization makes no effort to preserve locality of reference during image generation, and cache performance is consequently quite poor. We expect that reorganizing the data with an eye to cache performance would further reduce computation times.

**Evaluation of GeForce3 Hardware:** Chapter 5 introduces a rendering technique which requires graphics hardware which is still pre-release. One direction for future work is to implement this technique once the hardware becomes available.

**Extension of variance weighted sum of local normalized correlation:** The registration algorithm estimates patient pose by finding the minimum of the objective functions defined in chapter 2. Currently the search for this minimum is implemented using a quasi-Newton nonlinear optimization routine. Quasi-Newton methods iteratively build up an estimate of the inverse Hessian matrix of the objective function, and use this information to speed convergence. There are classes of function, for example Sum of Squared Difference functions, for which the inverse Hessian matrix can be trivially approximated in the neighborhood of the global minimum, and this enables methods like Levenberg-Marquardt nonlinear least-squares optimization to achieve nearly quadratic convergence [39]. We anticipate that, with suitable pre-processing, an objective function could be found with good resistance to image noise and image clutter, and also with an easily approximated inverse Hessian.

**In vivo studies of acetabular implant orientation:** Chapter 8 presents a phantom study which evaluates the X-ray/CT registration system for postoperative measurement of acetabular implant orientation. A similar study is underway using in vivo data from patients who have had bilateral total hip replacement. The preoperative CT for the second hip replacement will be analyzed to obtain ground-truth measurements for the study.

**Non-rigid registration:** This thesis addresses the problem of registering one or more rigid objects in X-ray images. It does not address the problem of deformation in patient anatomy. We anticipate that registration of non-rigid objects will be an exciting and fruitful research area.

# Appendix A

# Homogeneous coordinates

Throughout this document, we find it useful to represent points using 2- and 3-dimensional projective spaces. Projective geometry is a rich area, and we present only a few details here. For a more complete description, please refer to [14].

## A.1 Projective Spaces

Points in an n-dimensional projective space are represented by vectors with n+1 elements. A point in a three dimensional projective space is represented by a four element vector, and a point in a two dimensional projective space is represented by a 3 element vector. The mapping between points and vectors is not one to one; each point in a projective space can be represented by many different vectors. Vectors in a projective space are considered to represent the same point if they are scalar multiples of each other. That is to say, the 3 dimensional projective vectors [1, 2, 4, 1]T and [3, 6, 12, 3]T are considered equivalent. We call the n+1$^{\text{th}}$ parameter the *projective scale* of the vector, and represent it using the symbol $w$.

One way to visualize the relationship between points and vectors in projective space is to imagine that the points are mapped onto an n dimensional hyperplane described by the equation $w = 1$. Collinear vectors are considered equivalent, and represent a point corresponding to their intersection with this hyperplane. This is illustrated in figure A.1, which shows such an intersection in a 2D projective space. When a point is represented in this way, we say is is expressed in *homogeneous coordinates*.

We can make the correspondence between points in n-dimensional space and rays in n-dimensional

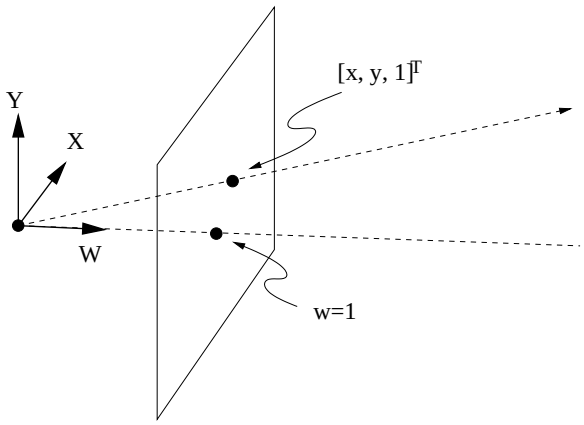



Figure A.1: The 2D point $[x, y]^T$ corresponds to the ray in 2D projective space which passes through $[x, y, 1]^T$.

projective space explicit by writing, for example,

$$\boldsymbol{p} = \begin{bmatrix} \alpha x \\ \alpha y \\ \alpha z \\ \alpha \end{bmatrix}, \tag{A.1}$$

where $p$ is a point in 3D projective space which corresponds to the 3D point $[x, y, z]^T$. It is conventional, however, to omit the free parameter $\alpha$ whenever possible, and instead write

$$\boldsymbol{p} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}. \tag{A.2}$$

## A.2 Homogeneous Transformation

We represent linear transformations in projective spaces using matrices, as we do in linear spaces. We transform a vector in the usual way: by left multiplying it using a transformation matrix. A linear transformation from n dimensional projective space to n dimensional projective space is represented with an n+1 by n+1 matrix. As is the case with projective vectors, two such transformation matrices are equivalent if they are scalar multiples of each other. As with vectors, it is conventional to fix the

projective scale so that the last element is equal to one whenever possible. We write

$$A = \begin{bmatrix} a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\ a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,0} & a_{3,1} & a_{3,2} & 1 \end{bmatrix}. \tag{A.3}$$

We call this a homogeneous transformation matrix, and use it to transform homogeneous coordinates. For example, in 3D homogeneous coordinates we write

$$\begin{bmatrix} \alpha x_1 \\ \alpha y_1 \\ \alpha z_1 \\ \alpha \end{bmatrix} = A \begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{bmatrix}. \tag{A.4}$$

In order to preserve the strict equality, it is necessary to explicitly represent the projective scale $\alpha$ on the left side of equation A.4. When using the more compact representation of equation A.2, it is conventional to omit this scale factor as well, writing

$$\boldsymbol{p}_1 = A\boldsymbol{p}_0 \tag{A.5}$$

and understanding the symbol "$=$" to represent *projective equality*, that is equality up to a scale factor.

Writing the elements of $A$ explicitly we have

$$\begin{bmatrix} \alpha x_1 \\ \alpha y_1 \\ \alpha z_1 \\ \alpha \end{bmatrix} = \begin{bmatrix} a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\ a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,0} & a_{3,1} & a_{3,2} & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{bmatrix}. \tag{A.6}$$

We note from this equation that $\alpha = a_{3,0}x_0 + a_{3,1}y_0 + a_{3,2}z_0 + 1$. This allows us to easily write the corresponding equation in 3D non-homogeneous coordinates

$$\begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} = \begin{bmatrix} \frac{a_{0,0}x_0 + a_{0,1}y_0 + a_{0,2}z_0 + a_{0,3}}{a_{3,0}x_0 + a_{3,1}y_0 + a_{3,2}z_0 + 1} \\ \frac{a_{1,0}x_0 + a_{1,1}y_0 + a_{1,2}z_0 + a_{1,3}}{a_{3,0}x_0 + a_{3,1}y_0 + a_{3,2}z_0 + 1} \\ \frac{a_{2,0}x_0 + a_{2,1}y_0 + a_{2,2}z_0 + a_{2,3}}{a_{3,0}x_0 + a_{3,1}y_0 + a_{3,2}z_0 + 1} \end{bmatrix}. \tag{A.7}$$

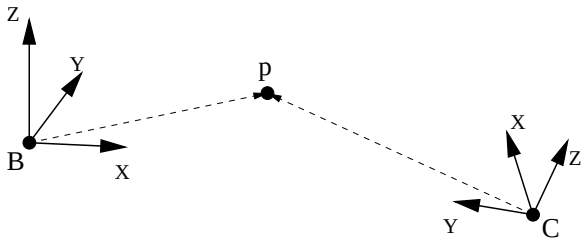



Figure A.2: The location of point $\boldsymbol{p}$ can be expressed with respect to both coordinate system $B$ and coordinate system $C$.

## A.3 3D Rigid Transformations

In this document, we are particularly concerned with a class of 4x4 matrices called 3D rigid transformation matrices. These are matrices which transform points from one coordinate frame to another. For example, figure A.2 shows a point, $p$, and two sets of 3D coordinate axes, which are labeled $B$, and $C$. The position of point p can be written with respect to either of the two coordinate systems. For example, it might have X, Y, and Z coordinates of 5, 2, and 2 respectively in coordinate system $B$, but X, Y, and Z coordinates 6, 7, and -1 respectively in coordinate system $C$. When we refer to a point which is represented in more than one coordinate system using homogeneous coordinates, we need to specify which coordinate system is used to define its coordinates. We do this using a left superscript containing the name of the coordinate system:

$$ {}^{\mathrm{B}}\boldsymbol{p} = \begin{bmatrix} 5 \\ 2 \\ 2 \\ 1 \end{bmatrix} \qquad {}^{\mathrm{C}}\boldsymbol{p} = \begin{bmatrix} 6 \\ 7 \\ -1 \\ 1 \end{bmatrix}. \tag{A.8} $$

Similarly, when a 4x4 transformation matrix takes coordinates from one coordinate system to another, we indicate this using a left superscript and a right subscript

$$ {}^{\mathrm{B}}\boldsymbol{p} = \left({}^{\mathrm{B}}T_{\mathrm{C}}\right)\left({}^{\mathrm{C}}\boldsymbol{p}\right), \tag{A.9} $$

where the 4x4 matrix transformation ${}^{\mathrm{B}}T_{\mathrm{C}}$ transforms points from coordinate system C to coordinate system B.

3D rigid transformations correspond to homogeneous transformation matrices which have the form

$$ {}^{\mathrm{B}}T_{\mathrm{C}} = \begin{bmatrix} & R & & \boldsymbol{t} \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{A.10} $$

where $R$ is a 3x3 rotation matrix, and $\boldsymbol{t}$ is a three dimensional translation vector. Because the bottom row of a rigid transformation is uniformly zero except for a 1 in the rightmost element, these transformations can be written without concern for the scale factor $\alpha$. We can see this by expanding equation A.9

$$\begin{bmatrix} 5 \\ 2 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} & R & & \boldsymbol{t} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 6 \\ 7 \\ -1 \\ 1 \end{bmatrix}, \tag{A.11}$$

which is a strict equality.

## A.4 Summary

This appendix presents only a very brief overview of homogeneous coordinates, introducing the ideas of projective scale, homogeneous transformations and projective equality. For a more thorough introduction to projective geometry, the reader is referred to [14].

# Appendix B

# Optically Tracked Pointers

During calibration and experimental procedures, it is frequently necessary to measure the position of fiducials, imager components, etc. with respect to a reference coordinate system. We do this using optically tracked pointers. This appendix describes the optical tracking device used, and how the pointers are constructed and calibrated.

## B.1 Optical Tracking Device

We use an optical tracking device called an Optotrak. The Optotrak is manufactured by Northern Digital, Incorporated, and consists of three 1D infrared sensitive CCD cameras. Each camera is equipped with a cylindrical lens, which causes the entire field of view to project onto a single line. The cameras are arranged so that if a feature is visible in all three cameras, its 3D position with respect to the Optotrak can be found by triangulation.

The Optotrak is used to to measure the position of infrared light emitting diodes (LEDs). When an LED is in the field of view, it causes a distinct intensity peak in the 1D image from each camera, and these peaks are used to triangulate the location of the LED. If more than one LED is in the field of view, the LEDs must be activated in turn so that intensity peaks from one LED do not interfere with the triangulation of the others. LED activation and camera measurements are synchronized by a piece of external hardware called a strober. The strober contains LED drivers, and timing circuitry to ensure that only one LED is active at any given time. Northern Digital's published specifications claim RMS LED tracking accuracies of 0.2mm or better along each axis [2].

Northern Digital supplies tracking markers, which are rigid assemblies, roughly 2"x4". Each tracking marker has an associated coordinate system, and houses 6 infrared LEDs at known positions within this coordinate system. The programming API for the Optotrak includes functions which calculate and return the 4x4 transformation matrix which relates the coordinate system of

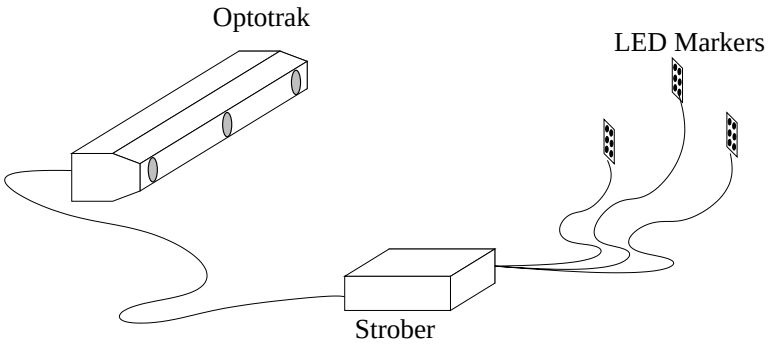



Figure B.1: The optical tracking system measures the position and orientation of LED markers.

each tracking marker with the coordinate system of the Optotrak itself. If desired, these transformations can be composed to find the coordinate transformation which relates the positions and orientations of two tracking markers. An illustration of the Optotrak system is shown in figure B.1.

## B.2 Pointer Construction

We construct each pointer by attaching a probe tip to a Northern Digital tracking marker. In our research we use two kinds of probe: sharp tipped probes are good for touching point features and surfaces, while cup tipped probes are good for locating spherical fiducials.

Each probe is calibrated as described in the next section. Sharp tipped probes are calibrated so that the location of the probe tip is known with respect to the coordinate system of the tracking marker, while cup tipped probes are calibrated so that when the tip is mated with a spherical fiducial of standard radius, the center of the fiducial lies at a known point in the coordinate system of the tracking marker. Figure B.2 illustrates both kinds of probes.

## B.3 Pointer Calibration

Our pointer calibration procedure involves two tracking markers. The first marker is the pointer be calibrated, while the second marker defines a reference coordinate system. The reference marker is rigidly attached to a calibration structure having an attached spherical fiducial, or a well defined punch-point which can be probed repeatably. We call this punch point or fiducial the *target* of the calibration. The probe is used to touch the target from many different angles, and at each angle we use the Optotrak to record the 4x4 matrix transformation which relates the coordinate system of the

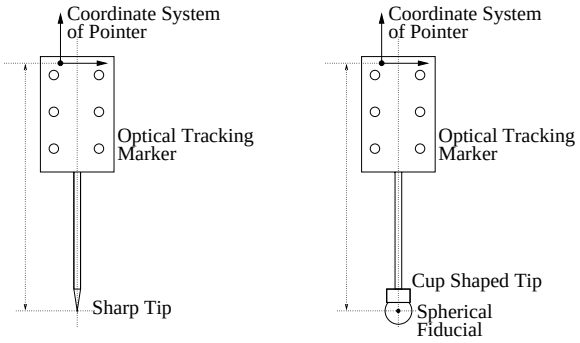



Figure B.2: Optically tracked probes are constructed by attaching sharp or cup-shaped tips to LED markers.

probe to the coordinate system of the reference marker.

Since the reference marker is rigidly attached to the calibration structure, the target does not move with respect to the coordinate system of the reference marker. We write the position of the target in the coordinate system of the reference marker using the 3D non-homogeneous vector $[x_0, y_0, z_0]^T$. Similarly, the probe tip is designed so that each time the target is probed the center of the fiducial or the location of the punch point is brought to the same place in the coordinate system of the pointer. We write the position of the target in the coordinate system of the pointer using the 3D non-homogeneous vector $[x_1, y_1, z_1]$. Since the two coordinate systems are related by a rigid transformation, the 4x4 matrix transformations returned by the Optotrak have the canonical form

$$T_i = \begin{bmatrix} & R_i & & \boldsymbol{t}_i \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{B.1}$$

where $T_i$ corresponds to the $i^{\text{th}}$ probing, $R_i$ is a 3x3 rotation matrix, and $\boldsymbol{t}_i$ is a 3 element translation vector. By definition, $T_i$ transforms points from the coordinate system of the pointer to the coordinate system of the reference marker, so we can write the 3D homogeneous equation

$$\begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{bmatrix} = T_i * \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{bmatrix}, \tag{B.2}$$

or equivalently, in non-homogeneous coordinates

$$\begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix} = R_i \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} + \boldsymbol{t}_i. \tag{B.3}$$

This equation can be rearranged to group the unknown parameters into one vector

$$\begin{bmatrix} R_i & -\mathrm{I} \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ x_0 \\ y_0 \\ z_0 \end{bmatrix} = -\boldsymbol{t}_i, \tag{B.4}$$

where I is the 3x3 identity matrix.

Combining equation B.4 over many observations gives an overconstrained system of linear equations which we solve using the Moore-Penrose pseudoinverse [43] [44]. The recovered 3D point $[x_1, y_1, z_1]^T$ is recorded and associated with the pointer.

In practice, we try to make the range of different angles from which we probe the target as wide as possible. We typically use between 70 and 100 observations, and see RMS residuals on the order of 0.2mm with a 6 inch probe tip.

# Bibliography


[1] Stereotactic radiosurgery and fractionated stereotactic radiosurgery. Blue Cross Blue Shield Association Medical Policy Manual, Policy Number 6.01.12.

[2] Northern Digital Optotrak product specifications, 2001. http://www.ndigital.com/optotrak.html.

[3] NVIDIA OpenGL extension specifications, March 2001. Copyright NVIDIA Corp. Available from http://www.nvidia.com/Developer.nsf.

[4] John R. Adler and Richard S. Cox. Preliminary clinical experience with the cyberknife: Image-guided stereotactic radiosurgery. In *Radiosurgery 1995*, pages 316–326, Boston, MA, June 1995. Stereotactic Radiosurgery Society.

[5] J. Amanatides and A. Woo. A fast voxel traversal algorithm for ray tracing. In G. Marechal, editor, *Proceedings of EUROGRAPHICS '87*. Elsevier, 1987.

[6] J. M. Balter, K. L. Lam, H. M. Sandler, J. F. Littles, R. L. Bree, and R. K. Ten Haken. Automated localization of the prostate at the time of treatment using implanted radiopaque markers: technical feasibility. *International Journal of Radiation Oncology Biology Physics*, 33(5):1181–1286, July 1995.

[7] B Cabral, N Cam, and J Foran. Accelerated volume rendering and tomographic reconstruction using texture mapping hardware. In *Proceedings, 1994 Symposium on Volume Visualization*, pages 131–132, 91–98, Washington D.C., October 1994. ACM Special Interest Group on Computer Graphics; IEEE Computer Society Technical Committee on Computer Graphics.

[8] F. Dachille, K. Kreeger, I. Bitter Chen, and A. Kaufman. High-quality volume rendering using texture mapping hardware. In *Proc. SIGGRAPH/Eurographics Graphics Hardware Workshop 1998*, 1998.

[9] Rachid Deriche. Fast algorithms for low-level vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(1):78–87, January 1990.

[10] A. M. DiGioia, B. Jaramaz, M. Blackwell, D. A. Simon, F. Morgan, J. E. Moody, C. Nikou, B. D. Colgan, C. A. Aston, R. S. Labarca, E. Kischell, and T. Kanade. The Otto Aufranc award. Image guided navigation system to intraoperatively measure acetabular implant alignment. *Clinical Orthopaedics and Related Reseearch*, 355:8–22, October 1998.

[11] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. Wiley, New York, 1973.

[12] George Eckel. *OpenGL Volumizer Programmer's Guide*. Number 007-3720-002. SGI Insight Developer Documentation Bookshelf, 1998.

[13] H. Erbe, A. Kreite, A. Jodicke, W. Deinsberger, and D.-K. Boker. 3D-ultrasonography and image matching for detection of brain shift during intracranial surgery. In *CAR '96 Computer Assisted Radiology. Proceedings of the International Symposium on Computer and Communication Systems for Image Guided Diagnosis and Therapy.*, pages 225–230, Paris, France, June 1996.

[14] Olivier Faugeras. *Three-Dimensional Computer Vision: A Geometric Viewpoint*. The MIT Press, Cambridge, Massachusetts, USA, 1993.

[15] J. M. Fitzpatrick, J. B. West, and C. R. Jr. Maurer. Predicting error in rigid-body point-based registration. *IEEE Transactions on Medical Imaging*, 17(5):694–702, October 1998.

[16] Janez Funda and Russell H. Taylor. On homogeneous transforms, quaternions, and computational efficiency. *IEEE Transactions on Robotics and Automation*, 6(3):382–388, October 1990.

[17] K. Gall, L. Verhey, and M. Wagner. Computer-assisted positioning of radiotherapy patients using implanted radiopaque fiducials. *Medical Physics*, 20(4):1153–1159, July 1993.

[18] K. G. A. Gilhuijs, K. Drukker, A. Touw, P. J. H. Van De Ven, and M. Van Herk. Interactive three dimensional inspection of patient setup in radiation therapy using digital portal images and computed tomography data. *International Journal of Radiation Oncology Biology Physics*, 34(4):873–885, March 1996.

[19] K. G. A. Gilhuijs, P. J. H. Van De Ven, and M. Van Herk. Automatic three-dimensional inspection of patient setup in radiation therapy using portal images, simulator images, and computed tomography data. *Medical Physics*, 23(3):389–399, March 1996.

[20] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen. The lumigraph. In *Computer Graphics Proceedings, Annual Conference Series*, pages 43–54, 528, New Orleans, LA, USA, August 1996. ACM SIGGRAPH.

[21] W. E. L. Grimson, G. J. Ettinger, S. J. White, P. L. Gleason, T. Lozano-Perez, W. M. Wells III, and R. Kikinis. Evaluating and validating an automated registration system for enhanced reality visualization in surgery. In *Proceedings of Computer Vision, Virtual Reality and Robotics in Medicine*, April 1995.

[22] A Gueziec, P Kazanzides, Williamson B., and R. H. Taylor. Anatomy-based registration of ct-scan and intraoperative x-ray images for guiding a surgical robot. *IEEE Transactions on Meical Imaging*, 17(5):715–728, October 1998.

[23] B. K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A (Optics and Image Science)*, 4(4):629–642, 1987.

[24] J. H. Hubbell. *Photon cross sections, attenuation coefficients, and energy absorption coefficients from 10 keV to 100 GeV.* National standard reference data series, 29. U.S. National Bureau of Standards, Washington, D.C., 1969. For sale by the Supt. of Docs., U.S. Govt. Print. Off.

[25] Coen W. Hurkmans, Peter Remeijer, Joos V. Lebesque, and Ben J. Mijnheer. Set-up verification using portal imaging; review of current clinical practice. *Radiotherapy and Oncology*, 58:105–120, 2000.

[26] Gelu Ionescu, S Lavallée, and J Demongeot. Automated registration of ultrasound with ct images: Application to computer assisted prostate radiotherapy and orthopedics. In *Proceedings of MICCAI '99*, pages 768–777, Cambridge, UK, September 1999.

[27] B. Jaramaz, M. DiGioia, T an Blackwell, and C. Nikou. Computer assisted measurement of cup placement in total hip replacement. *Clinical Orthopaedics*, 354:70–81, September 1998.

[28] B. Jaramaz, C. Nikou, and T. J. Levison. Cupalign: Computer-assisted postoperative radiographic measurement of acetabular components following total hip arthroplasty. In *Proceedings of MICCAI '99*, pages 876–882, Cambridge, UK, September 1999.

[29] H. E. Johns and J. R. Cunningham. *The Physics of Radiology*. Charles C. Thomas, Springfield, Illinois, 1983.

[30] L. Joskowcz, C. Milgrom, A. Simkin, L. Tockus, and Z. Yaniv. Fracas: A system for computer-aided image-guided long bone fracture surgery. *Journal of Computer-Aided Surgery*, 3(6), 1999.

[31] E. Kerrien, M-O. Berger, E. Maurincomme, L. Launay, R. Vaillant, and L. Picard. Fully automatic 3d/2d subtracted angiography registration. In *Proceedings, Medical Image Computing*

*and Computer-Assisted Intervention - MICCAI'99*, pages 664–671, Cambridge, UK, September 1999.

[32] P. Lacroute and M. Levoy. Fast volume rendering using a shear-warp factorization of the viewing transformation. In *Computer Graphics Proceedings, Annual Conference Series*, pages 451–458, Orlando, FL, USA, July 1994. ACM SIGGRAPH.

[33] Jean-Claude Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Massachusetts, 1991.

[34] S. Lavallée and R. Szeliski. Recovering the position and orientation of free-form objects from image contours using 3D distance maps. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(4):378–390, April 1995.

[35] L. Lemieux, R. Jagoe, D. R. Fish, N. D. Kitchen, and D. G. T. Thomas. A patient-to-computed-tomography image registration method based on digitally reconstructed radiographs. *Medical Physics*, 21(11):1749–1759, November 1994.

[36] Marc Levoy and Pat Hanrahan. Light field rendering. In *Computer Graphics Proceedings, Annual Conference Series*, pages 31–42, New Orleans, LA, USA, August 1996. ACM SIGGRAPH.

[37] Marc Levoy and Ross Whitaker. Gaze-directed volume rendering. *Computer Graphics*, 24(2):217–223, 1990.

[38] G. E. Lewinnek, J. L. Lewis, and R. Tarr. Dislocations after total hip replacement arthroplasties. *J Bone Joint Surg*, 60A:217–220, 1978.

[39] Donald M. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11(2):431–441, June 1963.

[40] Jerrold E. Marsden and Anthony J. Tromba. *Vector Calculus*. W. H. Freeman and Company, New York, 1981.

[41] D. E. McCollum and W. J. Gray. Dislocation after total hip arthroplasty: Causes and prevention. *Clinical Orthopaedics and Related Reseearch*, 261:159–170, 1990.

[42] M. J. Murphy. An automatic six-degree-of-freedom image registration algorithm for image-guided frameless stereotaxic radiosurgery. *Medical Physics*, 24(6):857–866, June 1997.

[43] R. Penrose. A generalized inverse for matrices. *Proceedings of the Cambridge Philosophical Society*, 51:406–413, 1955.

[44] R. Penrose. On best approximate solutions of linear matrix equations. *Proceedings of the Cambridge Philosophical Society*, 52:17–19, 1956.

[45] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipes in C - The Art of Scientific Computing*. Cambridge University Press, Cambridge, England, 1988.

[46] C. Rezk-Salama, K. Engel, M. Bauer, G. Greiner, and T. Ertl. Interactive volume rendering on standard pc graphics hardware using multi-textures and multi-stage rasterization. In *Proc. SIGGRAPH/Eurographics Graphics Hardware Workshop 2000*, 2000.

[47] V. Rudat, P. Schraube, D. Oetzel, D. Zierhut, M. Flentje, and M. Wannenmacher. Combined error of patient positioning variability and prostate motion uncertainty in 3D conformal radiotherapy of localized prostate cancer. *International Journal of Radiation Oncology Biology Physics*, 35(5):1027–1034, 1996.

[48] Mark Sarojak, William Hoff, Richard Komistek, and Douglas Dennis. An interactive system for kinematic analysis of artificial joint implants. In *Proceedings, 36th Rocky Mountain Bioengineering Symposium*, Copper Mountain, CO, USA, April 1999.

[49] Mark Segal and Kurt Akeley. *The OpenGL Graphics System: A Specification (Version 1.2.1)*. Silicon Graphics, Inc., Mountainview, CA, USA, 1999.

[50] G. W. Sherouse, K. Novins, and E. L. Chaney. Computation of digitally reconstructed radiographs for use in radiotherapy treatment design. *International Journal of Radiation Oncology Biology Physics*, 18:651–658, 1990.

[51] D. Simon, M. Hebert, and T. Kanade. Techniques for fast and accurate intrasurgical registration. *Journal of Image Guided Surgery*, 1:17–29, 1995.

[52] D. Simon, R. V. O'Toole, M. K. Blackwell, F. Morgan, A. M. DiGioia, and T. Kanade. Accuracy validation in image-guided orthopaedic surgery. In *Proceedings of the Second International Symposium on Medical Robotics and Computer Assisted Surgery*, pages 185–192, Baltimore, November 1995.

[53] R. Taylor, J. Funda, D. LaRose, Y. Kim, N. Bruun, N. Swarup, C. Cutting, and M. Treat. A passive/active manipulation system for surgical augmentation. In *Proceedings of the First International Workshop on Mechatronics in Medicine and Surgery*, Malaga, Spain, October 1992.

[54] R. H. Taylor, B. D. Mittelstadt, H. A. Paul, W. Hanson, P. Kazanzides, J. F. Zuhars, B. Williamson, B. L. Musits, E. Glassman, and W. L. Barger. An image-directed robotic system for precise orthopaedic surgery. *IEEE Transactions on Robotics and Automation*, 10(3):261–275, 1994.

[55] Roger Y Tsai. An efficient and accurate camera calibration technique for 3D machine vision. In *Proceedings CVPR '86: IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 364–374, Miami Beach, FL, USA, June 1986.

[56] J. Weese, T. M. Buzug, G. P. Penney, and P. Desmedt. 2D/3D registration and motion tracking for surgical interventions. *Philips Journal of Research*, 51(2):299–316, 1998.