(1)

DEPARTMENT OF THE AIR FORCE

AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GCS/ENG/91D-18

THREE-DIMENSIONAL MEDICAL IMAGE REGISTRATION USING A PATIENT SPACE CORRELATION TECHNIQUE

THESIS

Patrick Joseph Rizzuto Jr.
Captain, USAF

AFIT/GCS/ENG/91D-18

REPORT DOCUMENTATION PAGE

Form Approved
OMB No 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 1991	Master's Thesis

4. TITLE AND SUBTITLE
Three-Dimensional Medical Image Registration Using a Patient Space Correlation Technique

5. FUNDING NUMBERS

6. AUTHOR(S)
Patrick J. Rizzuto Jr., Captain, USAF

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology, WPAFB OH 45433-6583

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/GCS/ENG/91D-18

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The routine clinical use of three-dimensional data provided by modern medical imaging procedures is often impeded by the difficulty in accurately correlating the resultant volume datasets. These data are frequently obtained at different times using the same modality, or images of the same patient are sometimes produced using more than one imaging modality. In order to analyze the similarities and differences between such images, it is necessary for the medical imaging data to be spatially aligned using a process known as image registration. This research investigated a structure-based image registration technique based upon simple, three-dimensional relationships among user identified landmarks. An image registration system was developed to allow a user to identify anatomic landmarks or external markers anywhere within the entire volume of the medical imaging dataset. A graphical, user-centered interface design minimizes landmark placement error. Landmarks identified in images of one volume dataset are mapped to corresponding landmarks from another volume to determine a registration transformation. The transformation is then applied to the viewing parameters of a suitable volume visualization tool. Examples are shown using a surface rendering system.

14. SUBJECT TERMS
Image Registration, Medical Computer Applications, Computer Graphics, Image Processing, Theses

15. NUMBER OF PAGES
134

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used in announcing and cataloging reports. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling in each block of the form follow. It is important to **stay within the lines to meet optical scanning requirements.**

Block 1. Agency Use Only (Leave Blank)

Block 2. Report Date. Full publication date including day, month, and year, if available (e.g. 1 Jan 88). Must cite at least the year.

Block 3. Type of Report and Dates Covered. State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

Block 4. Title and Subtitle. A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

Block 5. Funding Numbers. To include contract and grant numbers; may include program element number(s), project number(s), task number(s), and work unit number(s). Use the following labels:

C	- Contract	**PR**	- Project
G	- Grant	**TA**	- Task
PE	- Program Element	**WU**	- Work Unit Accession No.

Block 6. Author(s). Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

Block 7. Performing Organization Name(s) and Address(es). Self-explanatory.

Block 8. Performing Organization Report Number. Enter the unique alphanumeric report number(s) assigned by the organization performing the report.

Block 9. Sponsoring/Monitoring Agency Names(s) and Address(es). Self-explanatory.

Block 10. Sponsoring/Monitoring Agency. Report Number. (If known)

Block 11. Supplementary Notes. Enter information not included elsewhere such as: Prepared in cooperation with...; Trans. of ..., To be published in .... When a report is revised, include a statement whether the new report supersedes or supplements the older report.

Block 12a. Distribution/Availablity Statement. Denote public availability or limitation. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NOFORN, REL, ITAR)

DOD	- See DoDD 5230.24, "Distribution Statements on Technical Documents."
DOE	- See authorities
NASA	- See Handbook NHB 2200.2.
NTIS	- Leave blank.

Block 12b. Distribution Code.

DOD	- DOD - Leave blank
DOE	- DOE - Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports
NASA	- NASA - Leave blank
NTIS	- NTIS - Leave blank.

Block 13. Abstract. Include a brief (Maximum 200 words) factual summary of the most significant information contained in the report.

Block 14. Subject Terms. Keywords or phrases identifying major subjects in the report.

Block 15. Number of Pages. Enter the total number of pages.

Block 16. Price Code. Enter appropriate price code (NTIS only).

Blocks 17. - 19. Security Classifications. Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contains classified information, stamp classification on the top and bottom of the page.

Block 20. Limitation of Abstract. This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.

AFIT/GCS/ENG/91D-18

THREE-DIMENSIONAL MEDICAL IMAGE REGISTRATION USING A PATIENT SPACE CORRELATION TECHNIQUE

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Computer Engineering

Patrick Joseph Rizzuto Jr., B.E.E.E.
Captain, USAF

December 1991

Preface

The purpose of this research was to develop a capability to examine and correlate large, three-dimensional medical image datasets, and help prepare a foundation for subsequent research in this area. My motivation for this research lies in a strong belief that three-dimensional visualization techniques offer the most promising solution to the data overload problem encountered in diagnostic medical imaging. As an engineer, I have learned that given the right tools, almost any problem is solvable. I fervently hope AFIT is able to employ the results of my efforts as one such tool in a continuing research program into this promising field.

In a fine example of Mills and Brooks' "Surgical Team" model (3:29), many people enabled me to bring this research to fruition. Of course, the most credit belongs to my wonderful wife Laura. She knew full well what she was getting into, and yet she still supported from the beginning our decision to go to AFIT. I also wish to thank my thesis advisor, Lt Col Marty Stytz, for introducing me to this problem, for the knowledge he gladly imparted upon me, and for showing me that research can be both hard work and fun at the same time. To my committee members, Lt Col Phil Amburn and Maj Dave Robinson, I give my thanks for their last minute efforts to make this thesis what it is, and for doing their darndest to keep me honest.

I am very grateful to Capts Joe Simpson, Rob Parrott, Patty Brightbill, and John Brunderman, as well as to Maj Dave Robinson, for the use of various pieces of their software. I'd still be coding if it weren't for their good work. Additional thanks go to Lt Col Amburn for the grounding he gave me in computer graphics and for providing the graphics lab resources and support necessary for me to accomplish this work. I especially want to thank my colleagues and friends, Patty and Rob, without whom this would have been just another AFIT project. The discussions, ideas, and comraderie we shared made this research the valuable learning experience it was for me.

To my darling children Patrick, Jillian, and Nicholas, I can't give thanks enough. You endured without knowing why, you revived me without knowing how, and you sacrificed without knowing what you were missing. You are the best kids a daddy could hope for!! Finally, and once again, I want to thank my beloved wife Laura, for sticking it out with me to the end, and learning to cheerfully say goodnight over a computer 'talk' session. You truly made the best out of the situation, not only for yourself, but for our family and the entire class. But most of all, thanks for knowing the difference between being my friend, my critic, my companion, my confidant, my taskmaster, my playmate, my wife, and mother to my children, and still being the best of all of them.

Patrick Joseph Rizzuto Jr.

Table of Contents

List of Figures

List of Tables

Table Page

Abstract

The routine clinical use of three-dimensional data provided by modern medical imaging procedures is often impeded by the difficulty in accurately correlating the resultant volume datasets. These data are frequently obtained at different times using the same modality, or images of the same patient are sometimes produced using more than one imaging modality. In order to analyze the similarities and differences between such images, it is necessary for the medical imaging data to be spatially aligned using a process known as image registration. This research investigated a structure-based image registration technique based upon simple, three-dimensional relationships among user identified landmarks. An image registration system was developed to allow a user to identify anatomic landmarks or external markers anywhere within the entire volume of the medical imaging dataset. A graphical, user-centered interface design minimizes landmark placement error. Landmarks identified in images of one volume dataset are mapped to corresponding landmarks from another volume to determine a registration transformation. The transformation is then applied to the viewing parameters of a suitable volume visualization tool. Examples are shown using a surface rendering system.

THREE-DIMENSIONAL MEDICAL IMAGE REGISTRATION USING A PATIENT SPACE CORRELATION TECHNIQUE

I. Introduction

1.1 Background

Visualiz^tion of multidimensional scientific data is a fast growing area in computer graphics. This practice exploits the fact that, in general, the human mind detects and analyzes complex relationships far more readily from visual images than from columns of numbers representing raw data. Medical imaging is one field greatly concerned with the collection and visualization of such scientific data. Medical practitioners are interested in techniques for creating accurate depictions of three-dimensional objects, such as human organs, on a computer display (40). Also important are tools and techniques for processing and manipulating this data to provide additional useful information, not directly discernable from an image or series of images, to further assist the clinician.

The past two decades have seen an increase in the use of three-dimensional medical imaging techniques, beginning with the earliest applications of x-ray computer-assisted tomography (CAT) scans to primarily provide images of bone. During this time, ever more sophisticated technologies, such as magnetic resonance imaging (MRI), have been brought from laboratory research to routine hospital use, greatly increasing the information available to practicing physicians (a review of these technologies is presented in (35)). Along with the medical community's use of these imaging capabilities has come the ability for radiologists and clinicians to make increasingly accurate diagnoses, as well as to plan much more precisely for following up the diagnoses with appropriate radiotherapy, surgical, or other treatment (23) (31).

Most medical imaging scanners represent a volume to be imaged (i.e., some portion of a patient) by creating a three-dimensional array of data sampled and stored as a series of two-dimensional cross sections. It is often difficult for the typical medical practitioner to mentally reconstruct the three-dimensional relationships among structures within such volumes by viewing these individual slices (5:65) (7:27). As previously alluded to, the

radiologist is frequently called upon to communicate his or her analysis and recommendations to the physician responsible for treating a patient. While skilled at forming and manipulating a mental model of the three-dimensional aspects of a case from these images, the radiologist may be less successful in communicating this mental model to the clinician based solely upon the two-dimensional images available. Yet such communication is vitally important to physicians, who must visualize for themselves the shape and size of the abnormality being evaluated or the treatment being considered. Three-dimensional images therefore represent a new medium for this communication and can thus greatly facilitate an improved, common understanding between the radiologist and the clinician, as well as allow the physician to discuss the case with colleagues on the basis of the same, common understanding of the three-dimensional relationships involved (23) (7).

Three-dimensional imaging has great potential in numerous medical areas. One such application is surgical planning, where preoperative three-dimensional images can provide a more realistic view of what the surgeon will encounter during a surgical procedure than can standard x-rays (7). In another application, radiotherapy frequently combines CT data with the data of other medical imaging systems to prepare detailed plans for the irradiation of tumors, while trying to ensure minimal damage to surrounding tissues (31). This last example is representative of a number of applications calling for the fusion of data from various imaging modalities taken at different times, and demonstrates a technique known as *image registration.* Image registration has the potential for becoming an extremely useful, standard diagnostic tool in a number of medical imaging applications, and is the subject of this thesis. A brief definition of image registration is included in the next section, and a more detailed discussion of various registration techniques follows in the next chapter.

1.2 Definitions

Although both ultrasound and single photon emission computed tomography (SPECT) are extremely useful imaging techniques, the majority of medical imaging references in this thesis are to CT, MR, and positron emission tomography (PET). Therefore, these three terms, as well as other related terminology used throughout this work, are described in more detail below.

1.2.1 Axial View. A two-dimensional view of a volume projected along the z-axis of the patient space coordinate system. Also known as a *transverse view.* The resulting images lie parallel to the x-y plane, and provide head to toe cross-sectional views of the patient.

1.2.2 Coronal View. A two-dimensional view of a volume projected along the y-axis of the patient space coordinate system. The resulting images lie parallel to the x-z plane, and provide either anterior or posterior cross-sectional views of the patient.

1.2.3 Computed Tomography (CT). Also referred to as *Computer-Assisted, Computer-Aided, Computerized-Assisted* or *Computerized Axial Tomography (CAT)*, computed tomography is a medical imaging technique that uses x-rays to produce accurate and detailed images of a patient's internal anatomy. The CT scanner's development is commonly seen as a milestone in medical history, and ushered in a new age of high technology medical procedures, especially other electronic medical imaging techniques such as MRI and PET. Conceptually, the CT scanner employs the same radiographic principles as first introduced by Roentgen at the start of the 20th century. In the medical version of this process, X-ray photons emitted from an x-ray source penetrate, and are attenuated by, the human body's various tissues. Image reconstruction techniques then mathematically approximate the x-rays' attenuation as they pass through different regions of the body. The calculated attenuation values are then used to build a photographic or electronic image representative of the patient's internal anatomy (35:27–32).

1.2.4 Density. This term originated with CT scans, and in that context represents a relative measure of the opacity of the body's tissues to x-rays. It has since come to represent a value, determined during scanner space sampling by one of the medical imaging modalities, associated with each of the volume's voxels.

1.2.5 Image Registration. Image registration is an image processing and computer graphics concept where two different images are correlated with one another in some fashion. According to Toennies *et al.*, the image registration process "... is a mechanism to ... represent in the same space ... two or more images of the same object from different modalities or from different time instances, using geometric properties that can be found in all images" (40:53). For medical imaging applications there are two basic classes of registration techniques. In the first, *structure-based* class, external markers, containing radiologically active compounds (safe to the patient when used properly), are sometimes fit to readily identifiable points of a patient's body. The external markers are resolvable by certain medical imaging modalities, and since they coincide with known anatomical features visible in scans from other modalities, images from different volumes may be registered by aligning the set of corresponding landmarks and markers (19:21 22). The

second, *surface-based* class of methods employ a variety of sophisticated statistical and geometric algorithms to automatically obtain a best-fit of one volume with another, without operator intervention (26).

In certain of these methods, data is projected from object space into screen space for an operator's use. This allows the operator to specify certain screen space coordinates, which may be readily recomputed back to object (and therefore patient) space coordinates, to be used for the image registration process. In all of these methods, a registration transformation is calculated in some image space to correlate the datasets. Various forms of rigid body differences between the datasets may then be compensated for, such as might be encountered when a patient's orientation within the scanner has changed between scans (whether intentionally or not). Thus, two different volume's of data, collected at times T_1 and T_2, would appear in screen space as if the patient were positioned in scanner space at those times in exactly the same way (40:54-56). Also, certain complex structural deformations (27:452) or other distortion effects (possibly introduced by the medical imaging scanner itself) may also be compensated for in the registration process (32:667) (19:23) (21:302). The resultant geometric transformation may therefore be described as a three-dimensional translation and rotation about one or more axes (rigid body transformations), or a scaling or shearing of the dataset in one or more planes (affine transformations) (9:207).

Independent of the image registration technique used, the products of this procedure can then be used in a number of clinically useful ways. A *comparative analysis* measures relative changes in a particular region's structure or function occurring during some elapsed period of time (40:53). A *composite analysis* combines dissimilar, yet complementary, data to produce a synergistic effect whereby the composite image provides information regarding some aspect of the patient's body not obtainable from the individual images by themselves (26:20-21) (16:817 818). Additional information regarding image registration techniques and examples of practical applications are presented in Chapter 2.

1.2.6 Image Space. Image space is an abstract three-dimensional space that contains the common reference systems for the image registration process. The data for each scan, referenced by coordinates in the unique object space associated with each volume scanned, must be registered in image space before the data may be manipulated or analyzed.

1.2.7 Landmarks. Also known as *Fiducial* or *Fiduciary* markers, landmarks is a general term that represents two types of certain three-dimensional points in patient space. *Anatomic Landmarks* are unambiguous, easily recognizable body structures, organs, or other organic feature identifiable in a particular imaging modality (e.g., the xyphoid process at the sternum's base or the sutures between skull bones). *External Markers* are artificial substances affixed to the patient's body and are observable by a particular imaging modality (e.g., vitamin E capsules used in MR images). Throughout this thesis, "landmarks" may be used interchangeably with "anatomic landmarks and/or external markers" unless a specific form is explicitly referenced.

1.2.8 Magnetic Resonance Imaging (MRI). Magnetic resonance imaging, previously known as *nuclear magnetic resonance (NMR)*, provides exquisitely detailed images of the body's internal soft tissues. This technology exploits the fact that certain atomic nuclei, such as hydrogen, possess a quantum mechanical characteristic known as spin. Since these atomic nuclei also possess a positive electric charge, they generate a magnetic field as they spin. Thus, they act as magnetic dipoles, and will therefore align themselves along the field lines of any external magnetic field. In addition, each nuclei's spin axis precesses about the axis of the external magnetic field at a frequency related to the external field's strength. If, while under the external field's influence, the nuclei are also exposed to a burst of radio frequency (RF) energy tuned to the nuclei precession frequency, then those nuclei will precess in phase and emit a coherent RF signal of their own at the same precession frequency. That emitted RF signal's strength is therefore proportional to the concentration of the atomic nuclei in a particular portion of the body's volume. With the proper combination of varying magnetic fields and RF signal pulses, the locations of these concentrations of nuclei can be pinpointed within the body. These concentration values are then used to build a photographic or electronic image representative of the patient's internal soft tissue anatomy (35:35–40).

1.2.9 Object Space. Also referred to as *Scanner Space*, object space encompasses the three-dimensional volume sampled by a medical imaging system. Scanner space (defined by a coordinate system affixed to the scanner) is where data is obtained from various points within patient space (defined by a coordinate system rigidly affixed to some part of the patient). These data may be transformed into an object reference system associated with that specific scan. Object reference systems are then manipulated to register their corresponding medical images (40).

1.2.10 Patient Space. The three-dimensional space defined by a coordinate system rigidly attached to some bony point, usually the skull, of the patient being medically imaged. In this orthogonal system, the z-axis lies parallel to the patient' longest (head-to-toe) axis, the y-axis lies parallel to the patient's front-to-back axis, and the x axis lies parallel to the patient's side-to-side axis (36:38).

1.2.11 Positron Emission Tomography (PET). Positron emission tomography is a medical imaging technique that uses positron emitting radioisotopes attached to biochemical tracers to provide radiographic images of certain biological processes. PET imaging is based upon the property of beta (β^+) decay. In this process, relatively unstable isotopes of certain atoms (such as oxygen, carbon, or nitrogen) tend to break down into more stable atomic forms through the emission of β^+ particles, also known as positrons. A positron is the antiparticle of an electron, i.e., they share the same physical characteristics except that the positron has a positive (+1) electric charge while the electron has a negative (-1) electric charge. As a radioisotope decays, a positron is released from the atom's nucleus, and is immediately acted upon by the other charged particles, especially the electrons, surrounding it. The mutual attraction between a positron and a nearby electron eventually causes the two to come into contact, destroying both particles and converting all of their mass into energy. This energy takes the form of two weakly energetic (511 keV) gamma (γ) rays traveling in opposite directions. These γ rays may then be detected and used to localize the radioisotope's approximate location in three-dimensional space (35:44 -[illegible].

1.2.12 Sagittal View. A two-dimensional view of a volume projected along the x-axis of the patient space coordinate system. The resulting images lie parallel to the y-z plane, and provide either right or left side, cross-sectional views of the patient.

1.2.13 Screen Space. Screen space is the two-dimensional space encompassed by the display surface of an image display device, such as a cathode-ray tube (CRT). The individual pixels composing the display screen are accessed by an integer valued u v coordinate system.

1.2.14 Surface Rendering. Surface rendering is a group of computer graphics techniques for displaying surface representations of three-dimensional objects contained within a volume dataset. In these techniques, the user must specify the object to be visualized (actually, only the surface of the desired object is visualized) by providing a range of density values representative of the object of interest. Every voxel is then sampled, and a binary

classification decision is made as to whether or not the desired surface passes through the voxel. Only those with a value falling within the specified range contribute to the final image displayed. Methods such as Lorenson and Cline's *Marching Cubes* algorithm fit two-dimensional geometric primitives (e.g., triangles) to the estimated surface, which may then be shaded and displayed using conventional polygon rendering techniques (38) (18) (11).

1.2.15 Volume Rendering. Also known as *Volumetric Rendering*, volume rendering is a computer graphics technique for creating images of three-dimensional objects by visualizing each voxel individually and projecting it onto a display surface. Of the several classes of volume rendering tools, the most popular use either *Ray Casting* or ordered voxel projection (e.g., *Back-to-Front (BTF)* (11)) to directly visualize voxels composing the volume dataset. References that provide additional details of these techniques are (18), (17), (5), (23), and (37). An important feature of volume rendering techniques is that they create the impression of transparent surfaces, allowing one to view an object's interior. In this approach, very little preprocessing is performed on the volume dataset; these direct visualization techniques serve primarily to provide an "enhanced presentation of the original data" (17:829). Thus, very fine or thin anatomical structures may be appreciated without the addition of nonexistent objects or the deletion of meaningful information from the dataset.

1.2.16 Voxel. A voxel is a three-dimensional volume element, shaped as a rectangular parallelpiped, and is analogous to the term *pixel*, which represents a two-dimensional picture element of a display screen. A voxel is formed when space is divided by three sets of mutually perpendicular parallel planes. When the inter-plane spacing for each set of planes is identical (i.e., the voxel has a cubic shape, all sides having the same dimension), that dissection of space is known as the *Cuberille* model (11:52–53).

1.3 Problem

Significant advances in computer and associated electronic imaging technology have made CT, MRI, and other medical imaging modalities, such as PET and SPECT, routinely available for research and clinical use. Nevertheless, the ability to extract accurate and diagnostically useful information from these three dimensional images has not kept pace with the ability to produce the images themselves. Continued development is required to

produce new techniques to better display, manipulate, and measure the important information contained in these images (29). One such area of research addresses the fundamental problem of relating some aspect of the body's function to its structure. For example, it has long been recognized that three-dimensional images of the brain produced by CT, MR, PET, and SPECT scans contain complementary anatomical and physiological information, information that has become increasingly important for neurologic research, diagnosis, and treatment (16) (26). The routine clinical use of advanced three-dimensional volume visualization techniques on these data is impeded by the difficulty in accurately aligning several, multimodality volume datasets. Such image registration is a prerequisite to performing and displaying the results from various analyses of the volumes.

The goal of the research recorded in this thesis was to demonstrate a new image registration technique based upon simple, three-dimensional relationships among several user identified landmarks. Some correlation techniques for registering medical images are computationally intensive, such as the surface-based methods. Other methods, such as those from the structure-based category, are frequently based upon obtaining at least three pairs of matching locations on images derived from the volumes, although "[c]learly the greater the number of accurately specified landmarks, the better the result" (40:55 56). This latter comment is supported by the fact that New York University radiologists use 7 to 12 pairs of landmarks (19:24). In this case, a skilled user, perhaps even expert in anatomy, precisely and interactively identifies these landmarks. This thesis shows the feasibility of performing image registration using a simpler process of identifying landmarks and exploiting their three-dimensional linear relationships in a new way. Coupled with the use of external markers during medical imaging sessions, this technique can reduce the complexity and computational expense of the image registration process.

1.4 Research Objectives

A three-dimensional medical image registration system was designed and developed as part of this research effort. In so doing, I achieved the following objectives:

- Produced a system capable of displaying a number of two-dimensional medical images from volume datasets. The data was obtained from standard MR and CT imaging equipment, and was previously filtered so density values stored in each voxel represented 8-bit gray-scale values. The datasets, in addition to storing the usual density information gathered during a scanning session, also contained data regarding the location of external markers affixed to the patient.

- Incorporated a simple user interface to allow anatomic landmark and external marker identification in the two-dimensional medical images.
- Registered two different volumes by using simple three-dimensional relationships to determine the geometric transformation required to orient one volume with respect to another, based upon structures identified in patient space. A user manually selects those landmarks from various medical images extracted from the volume datasets.
- Displayed three-dimensional renderings of the registered volumes, demonstrating corresponding views of the differently oriented objects within the volume dataset.
- Displayed registered images of the same modality taken at different times, demonstrating the differences between the images, with the capability to either highlight those differences in one image or remove all common features in both images, thus displaying just the differences between both volumes.
- Displayed registered images of different modalities, demonstrating in a single, composite image those areas in an image of one modality corresponding to an area designated in some other imaging modality.

1.5 Assumptions

The following assumptions were made throughout the entire research effort:

- The image registration system developed and described by this thesis would be eventually incorporated within an overall three-dimensional medical image processing system under continuing development at the Air Force Institute of Technology (AFIT). Therefore, additional design features were incorporated into the image registration software to allow it to interface with other software being developed for this overall system.
- Previously developed software (e.g., a volume visualization package) would be available and would function correctly for use during this effort. This allowed my research to focus on image registration, and I did not have to repeat the designs and deal with the issues associated with developing this software, since it already existed.
- The medical image datasets to be registered by this system must contain the same number of voxels in each dimension. In cases where this assumption does not hold, various interpolation techniques exist that may be used to perform a resampling of the data to ensure the volumes have the same dimensions. However, this research

did not address the resampling question, and any necessary changes to the datasets must have been made prior to this system's use.

1.6 Scope and Limitations

The constraints for this thesis effort were as follows:

- This system employs a structure-based image registration method. The system does not use surface matching or stereotactic techniques, which are popular alternatives to the image registration process discussed in this thesis.
- This system requires a user to interact with the volume datasets to identify landmarks to be used for image registration.
- The identification of landmarks often requires the use of external markers to assist a user to correlate certain anatomical features with observed physiological processes. This system is therefore frequently unable to perform retrospective registration on medical images, since the data must be specially acquired to include any external markers required for the registration process.

1.7 Summary

Current radiological diagnostic practices rely upon the use of two-dimensional x-ray films or images to demonstrate certain views taken through a patient's body. An improved way to visualize volumes, capable of demonstrating even subtle interrelationships between structures, is to exploit the inherent three-dimensional nature of modern medical imaging procedures. One technique gaining much interest in this area is known as image registration, which provides the capability for accurately aligning medical images taken at different times or using different imaging modalities. Practical applications of this procedure include providing physicians with the ability to take more precise measurements critical for radiotherapy and surgical treatment planning, as well as to better understand complementary functional and structural information regarding some organ or portion of a patient's body.

1.8 Thesis Organization

The remainder of this thesis is organized into four chapters following this introductory chapter. Chapter II provides a review of the current literature on the subjects of

visualizing three dimensional volumes, medical tomographs, and image registration. Chapter III presents an analysis of the image registration system's requirements and presents the system's design. Chapter IV describes details of the system's implementation and overviews the image registration system's functionality and operation. Chapter V presents conclusions drawn from this implementation effort and actual use of the image registration software on actual medical imaging data, and recommends areas for future research efforts. Appendix A contains the products of the analyses performed for this project. Appendix B contains C++ class specifications for the major software system components, demonstrating the system's overall architecture.

II. Literature Review

This chapter presents background information on the image registration process. As an introduction to my research, this chapter provides a framework for understanding the design decisions and implementation details raised in later chapters. In order to motivate this discussion, I briefly describe image registration in clinical practice. Following this is a more detailed examination of several popular techniques. Existing systems frequently present two-dimensional or three-dimensional images, before and after they have been registered, to the operator's display for interactive use. It is therefore appropriate to also overview several user interface issues in my review of the pertinent literature.

2.1 Image Registration Applications

To date, the majority of applications for the different medical imaging modalities discussed in the previous chapter have been for the purpose of clinical analyses, where the goal has been to "... get quantitative measures of structures in an absolute, comparative, and composite fashion" (40:53). In the following sections I briefly discuss the nature of comparative and composite analyses and their utility to modern medical practice.

2.1.1 Comparative Analysis. The goal of a comparative analysis is to use the products of the same medical imaging modality to provide a user with the ability to detect and measure changes in an object's structure over time. Significant changes constitute those differences between diagnostic medical images (demonstrating a patient's initial pathological condition, for example, following a traumatic injury, during diagnosis of a disease, or resulting from a deformity) and those images recorded during and following the course of the patient's medical or surgical treatment. As an example of this, a CT scan series of a victim's skull, taken immediately after an accident, would perhaps show severe craniofacial bone damage. Later, those images might be compared with another CT scan series taken following the patient's reconstructive surgery to evaluate initial post-operative success. Additional CT scans might also be taken, as the patient's treatment and recuperation progresses, to measure other significant changes (such as bone graft loss due to reabsorption) and to assess the therapy's long-term outcome.

Differences in medical images due to patient movement during the imaging session, dissimilar patient orientations, or revised scanner resolution settings from one imaging session to the next, fall into the category of clinically insignificant changes. These changes

do not provide medically useful information pertinent to the analysis at hand. Nevertheless, it is vitally important for the diagnostician to understand and account for these differences to ensure an accurate and meaningful analysis (40:53).

2.1.2 Composite Analysis. As discussed in Section 1.2.5, CT and MR images provide anatomic, structural information while PET and SPECT radionuclide images help the clinician to determine physiologic, functional information. PET and SPECT scans measure some characteristic of an organ's function, such as metabolic activity rate or blood flow, by monitoring the uptake of radioactively labeled substances introduced into a patient's body. Thus, a tumor's benign or malignant nature may be better understood by studying images showing the concentration of radioactively tagged glucose found in the tumor body and the network of blood vessels supplying the tumor. Unfortunately, the same PET and SPECT scans that provide this insight to a tumor's functioning provide almost no detailed information on the tumor's location. On the other hand, MR and CT images, while unable to provide the same physiologic interpretation as just discussed, can reproduce in detail the tumor's conformation and location with respect to other organ and body structures (31) (16).

The images created from these varying modalities thus contain complementary information, "... greatly enhanc[ing] the information content of each and hence their diagnostic value" (19:20). The process of composite analysis, then, is used to "... create integrated multimodality images by mapping features from one image onto an image obtained with another modality" (16:817). Building upon the previous section's illustrative example, Levin *et al.* (16:820) describe a case where a CT scan taken of a patient's skull following a head injury demonstrated a fracture but did not show any apparent damage to the brain immediately adjacent to the fracture. A later MR study indicated the presence of brain damage, but the damaged area could not be localized with respect to the skull's external surface. Only after a composite image was created, overlaying portions of the CT image on the MR image, was the relationship between the skull fracture and the underlying brain damage fully appreciated. As Levin *et al.* point out, such an analysis would certainly be more useful than either of the individual images by themselves, allowing a surgeon to accurately plan a neurosurgical strategy for repairing this damage.

2.2 The Image Registration Process

Having presented some clinical applications to demonstrate image registration's potential in medicine, this section discusses the process itself. The objective of this section

is to provide a greater appreciation for the wide variety of solutions to this problem and the lack of unanimous support for a single image registration method. To do this I first provide an overview of several well known image registration techniques, and a comparison of the two techniques most closely related to my own research. This is followed by a more detailed description of each technique.

2.2.1 Overview. Before discussing individual image registration techniques, it is useful to introduce them as a group and place them in context with one another. Maguire *et al.* (19:21) assert there are five major approaches to image registration, summarized as follows:

1. Techniques relying upon the identification of unambiguous anatomical structures visible in the medical images under study, as characterized by the work of Toennies *et al.* (40) and Maguire *et al.* (19).
2. Techniques relying upon the presence of discernable surfaces in the medical images under study, as characterized by the work of Pelizzari and Chen *et al.* (26) (16) (12). In this category, Maguire *et al.* make a distinction between principal axes and moment matching methods, which, for the purposes of this paper, are considered synonymous.
3. Stereotactic techniques using a rigid frame affixed to the patient (usually the patient's head), as characterized by the work of Vandermeulen *et al.* (42), Schad *et al.* (31), and Rhodes *et al.* (28).
4. Techniques using an anatomic atlas as an intermediate reference system, as characterized by the work of Evans *et al.* (6), Fox *et al.* (10), and Bajcsy *et al.* (1).
5. Techniques using external markers visible to the medical imaging modality being employed, placed on certain readily identifiable locations of the patient's body, again as characterized by the work of Maguire *et al.* (19).

On the other hand, Pelizzari *et al.* (22:259) consider only structure-based (combining the use of anatomical landmarks and external markers), stereotactic-based, and surface-based methods. They further classify the latter of these categories into principal axis and surface matching/surface fitting optimization techniques. In my discussion of these image registration methods, rather than adhere strictly to either of these classifications, I have chosen to merge the two groups' positions for two reasons. First, even though the anatomic atlas category also employs stereotactic frames (see Section 2.2.6), the use of

such atlases as an intermediate step sufficiently distinguishes this technique from other stereotactic approaches to warrant separate consideration. Second, external markers are very often used in conjunction with anatomic landmarks, so the combination of these two conceptually similar categories simplifies the overall classification. Therefore, the use of anatomic landmarks and external markers is considered in one section on structure-based techniques, and another section briefly discusses the use of atlases. Figure 2.1 illustrates the interrelationships among these varied approaches, and each is discussed in greater detail later in this chapter.

It is important to note here that there is no clear consensus as to whether one technique is superior to another. On the other hand, there is much evidence to suggest that in many ways they perform complementary functions, and several of these techniques can be used in conjunction with one another. For example, researchers at the University of Michigan developed their U-MPlan radiation therapy treatment planning system to use both surface and structure identification techniques. The anatomic objects to be visualized and the desired planning operations to be performed determine the selection of a particular method (21).

2.2.2 Comparison of Structure-Based and Surface-Based Techniques. Before presenting the detailed mechanics of the four forms of image registration identified in the previous section (described in Sections 2.2.3 – 2.2.6), it is beneficial to summarize key characteristics of some of these techniques, see Table 2.1. The landmark mapping technique, discussed later in Chapter 3, is included in this table for completeness and to provide a basis for comparison between my method and other established methods. This summary deals only with the structure-based and surface-based methods, since they are the most directly related to my own research. I compare these two categories based upon the following criteria.

2.2.2.1 Data Displayed to User. Medical imaging data must frequently be presented on a display screen for the user to provide input to the system. The data displayed typically takes one of two forms:

- *Slice Data.* Medical image data, normally acquired as a series of transverse slices, is displayed one transverse slice at a time. This orthogonal view, however, is not always satisfactory. Since the patient's orientation may change between scanning sessions, a set of points visible on one scan's slice may not all be visible in the corresponding

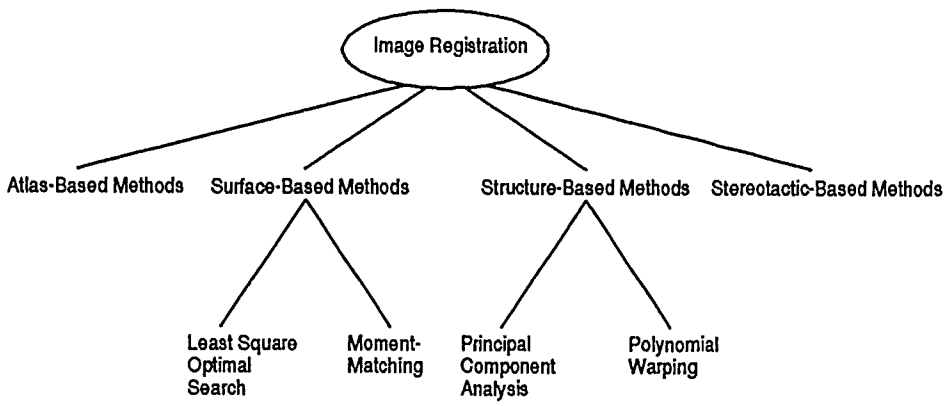

Figure 2.1. Taxonomy of Image Registration Techniques

slice from the other scan. In these cases, it is necessary to construct and display *oblique* (off-axis) slices, creating a two-dimensional projection of the scanned volume from any angle.

- *Surface Rendering.* Medical image data, organized as a stack of two-dimensional slices composed of pixels, may be internally represented as a volume within the computer system. From this volume data set, three-dimensional computer graphics algorithms can create a surface rendering (e.g., of the skin or brain surface). Once displayed, such renderings may be manipulated by the user to provide arbitrary views of the surface in question. This allows a study of the surface's conformation and the identification of certain structures lying on that surface.

2.2.2.2 Registration Primitives. The following three primitive types are typically used by the structure-based and surface-based image registration techniques:

- *Anatomic Landmarks.* Points in patient space corresponding to certain structures or locations in the patient's body (e.g., the ear canal's external opening, the tip of the sacrum at the spine's base). These points must be identified by the user based upon the data displayed.
- *External Markers.* Points in patient space created by placing artificial objects on the patient's body in close proximity to known anatomical structures (such as the umbilicus or the tip of the sternum) not discernable by the desired imaging modality. These artificial objects, visible to the desired imaging modality, then allow the user to identify the corresponding anatomic structures. Again, these points must be identified by the user based upon the data displayed.
- *Surface Contours.* The closed curve outline of a three-dimensional surface intersected with a two-dimensional plane. Contours may be manually defined (by a user employing an interactive pointing device), semi-automatically defined (requiring some user input), and completely automatically defined.

2.2.2.3 Structure-based Algorithms. Two algorithms are employed by the structure-based techniques, each performing image registration using anatomic landmarks and/or external markers. Each of these algorithms is discussed in greater detail in Section 2.2.3.

- *Principal Component Analysis.*
- *Polynomial Warping.*

2.2.2.4 Surface-based Algorithms. Two algorithms are employed by the surface-based techniques, each performing image registration using surface contours. Each of these algorithms is discussed in greater detail in Section 2.2.4.

- *Least Square Optimal Search.*
- *Moment Matching.*

2.2.2.5 Ease of Use. Based upon my research into the previous four categories, I established an arbitrary, relative ranking of the methods for identifying to the system those primitives to be used for image registration. In this ranking, techniques that allow a user to most easily, accurately, and consistently select a particular registration primitive from the data displayed warrant a ranking of '1.'

- *1* – Primitives (surface contours) are automatically or semi-automatically selected. Techniques in this category require very little user interaction, but some manual correction is occasionally required.
- *2* – User selects primitives (anatomic landmarks and/or external markers) directly from a volume rendering. Techniques in this category provide the user with a very good idea of what portion of the volume they are examining, allowing the user to select a specific voxel displayed in context with all other voxels composing the volume.
- *3* – User selects primitives (anatomic landmarks and/or external markers) from surface renderings. Techniques in this category provide the user with a good idea of what portion of the volume they are examining, allowing greater confidence in their landmark selection.
- *4* User selects primitives (anatomic landmarks and/or external markers) from slices. Techniques in this category require the user to form a mental picture of the volume from the slice(s) displayed, complicating the task of identifying landmark locations within the volume.

2.2.2.6 Registration Accuracy. Also based upon my research, I established an arbitrary, relative ranking of the image registration technique's ability to correlate corresponding primitives in two separate images. In this ranking, techniques that provide the most accurate mapping or estimate of one volume's registration with a second warrant a ranking of '1.'

Table 2.1. Summary of Image Registration Evaluation Factors.

Team	Data Displayed to User		Registration Primitives			Structure-Based Algorithms		Surface-Based Algorithms			
	Slice Data	Surface Rendering	Anatomic Landmarks	External Markers	Surface Contours	Prin. Comp. Analysis	Polynomial Warping	Least Squares Optimal Search	Moment Matching	Ease of Use[a]	Registration Accuracy[b]
Fellingham *et al.*	Yes	No	No	No	Yes, Automatic	—	—	No	Yes	1	2
Landmark Mapping[c]	Yes	No	Yes	Yes	No	No	Yes[d]	—	—	4	1
Maguire *et al.*	Yes	No	Yes	Yes	No	No	Yes	—	—	4	1
Pelizzari *et al.*	Yes	No	No	No	Yes, Automatic[e]	—	—	Yes	No	1	2
Tiede *et al.*	No	Yes	Yes	No	No	No	Yes	—	—	3	3
Toennies *et al.*	No	Yes	Yes	No	No	Yes	No	—	—	?	3

[a]Arbitrary, relative rankings, see Section 2.2.2.5.
[b]Arbitrary, relative rankings, see Section 2.2.2.6.
[c]This entry represents the image registration technique discussed in this thesis.
[d]The Landmark Mapping technique is most similar to this category of structure-based algorithms.
[e]Until recently a seed point was required.

- *1* – Techniques in this category perform registration using uniquely identified landmarks selected directly from slice data. This allows for the possibility of a theoretically perfect, one-for-one mapping of corresponding landmarks between volumes.
- *2* – Techniques in this category perform registration using statistical correlation of surfaces defined by a large number of points (generally more than 100). The surfaces are automatically generated from surface contours, which are subject to the accuracy of the contour following algorithm.
- *3* – Techniques in this category perform registration using statistical correlation of a relatively small number of uniquely identified landmarks (generally less than 20). In this case, the landmarks are selected by the user based upon their location on a surface rendering, which is subject to the accuracy of the surface estimation algorithm.

2.2.3 Structure Identification and External Markers Techniques. This section presents a detailed description of two structure-based image registration techniques. Since my own research falls into this category, this information will provide a better understanding of the design decisions and implementation described in following chapters. In each case, I outline the sequence of events followed to register images in this manner, starting with the data displayed to the user, the primitives used for image registration, and the registration algorithm itself. These two methods are primarily differentiated in this section based upon the image registration algorithm employed, as introduced in Section 2.2.2.3.

2.2.3.1 Principal Component Analysis Algorithm. At the Hospital of the University of Pennsylvania, Toennies *et al.* have devised an image registration system based upon a mathematical model of the entire scanning process. Their system registers images in an environment where the user interactively specifies anatomic landmarks, and then uses the landmarks to determine the correct transformation to correlate the images so one may be overlaid on another. Their efforts, as reported to date, have been confined to CT scan comparative analyses (although the principle may be extended to other modalities and applications), and their primary need for image registration is to compensate for patient positioning differences from scan A to scan B. This compensation, they assert, can always be represented in terms of a translation, followed by a rotation, of the data set representing the patient's body in scan B with respect to scan A. The following summarizes the process's four steps (additional details may be found in (40:55-57) (39:381-383)):

1. High quality renderings of some surface (such as the skull) contained within each of the volumes to be registered are displayed side by side to the user. These surface renderings must be of sufficient fidelity that fine anatomic structures, such as the skull's sutures and ridges (frequently used landmarks), are visible. The user reviews both renderings and locates corresponding locations in each, selecting those to be used for the image registration process with a pointing device.

2. The landmarks are used to form an object reference system for each volume using the *Principal Component Analysis* method. The centroid of the landmarks becomes the object reference system's origin, and mutually orthogonal, principal axes are then calculated. The use of principal component analysis tends to smooth out errors introduced into the set of landmark pairs. Even so, if one or more of the landmark pairs as specified by the user are significantly misidentified, the resulting object reference systems will be inaccurate. In this case, the offending landmarks must be determined and eliminated from further consideration, in a process they refer to as cross-validation.

3. Once the two object reference systems have been established, they are used to determine the registration transformation necessary to correlate one volume with the second. This is simply the geometric transformation, consisting of a translation followed by a rotation, required to align one object reference system with the other.

4. After the two volumes have been registered, the transformed volume is "redigitized." The previous steps had the effect of registering the overall scene, but not the individual voxels composing the volume. As some of the volume's voxels obtained from scan B might not have been captured in object space during scan A, those voxels must be removed from the final, registered image.

Inaccuracies may be introduced into this process in at least three ways. First, using the object reference systems for image registration, rather than the individual landmarks themselves, does not remove all misidentified landmarks. Those remaining result from accuracy errors, too small to be detected in the cross-validation process, in the user's specification of a landmark's location. In other words, it is difficult to select exactly which voxels correspond to the same anatomic point in both volumes, especially when that selection is made from a rendered approximation of the actual surface. Second, precision errors may be introduced when each landmark is transformed from screen space (where they were selected by the user) to object space (where they are used to construct the object

reference system). Third, inaccuracies may also be introduced during the generation of the surface renderings, depending upon the particular surface estimator used.

2.2.3.2 Polynomial Warping Algorithm. As discussed earlier, different image registration techniques are frequently used in conjunction with one another. The New York University group of researchers, Maguire *et al.*, employs a structure matching technique that includes the use of flexible external markers placed at strategic locations along the patient's body. These markers literally serve to tie together the functional and structural imaging modalities; for example, CT and PET scans may be registered by identifying a marker, visible on the PET scan but not on the CT scan, that coincides with a known anatomic landmark visible only on the CT scan. Once data from the two scans has been obtained, image registration proceeds as follows (additional details are in (19:22-24)):

1. Slice data from the scanner is projected directly onto a display screen. Before the data may be displayed, it is manipulated so each slice from scan A contains the same number of pixels as the corresponding slice from scan B, and also so a pixel in each slice from scan A has the same x–y dimensions as a pixel in the corresponding slice from scan B.
2. The next step is to ensure corresponding slices from the two scans lie in the same plane. It is a practical impossibility (without special fixtures, see Section 2.2.5) to exactly align the patient so the slices obtained from scans taken at two different times, or using two different medical imaging modalities, share the same plane. It is therefore necessary to compensate for these differences by creating an *oblique* projection of a slice from one of the two scans. The user does this by first selecting a pivot point in the x–y (axial) plane, then specifying rotations about the y– and x–axes. The system then builds the new, oblique slice from the data stored in the three-dimensional stack of original slices (24:610).
3. Once the slices are coplanar, the user selects individual anatomic landmarks, or external markers placed in close proximity to known anatomical structures, present in both slices. After a sufficient number (typically ten) have been selected, each landmark of one slice is cross-correlated with the points surrounding the corresponding landmark selected in the other slice. This determines if the landmark selected in the second slice represents the best match with the first slice's landmark. If not, the algorithm discards that point in favor of a nearby, more closely correlated point.

4. The actual image registration algorithm uses the polynomials

$$x_i = a_0 + a_1 u_i + a_2 v_i + a_3 u_i v_i + a_4 u_i^2 + a_5 v_i^2 + \cdots \quad (2.1)$$

$$y_i = b_0 + b_1 u_i + b_2 v_i + b_3 u_i v_i + b_4 u_i^2 + b_5 v_i^2 + \cdots \quad (2.2)$$

to relate the set of landmark coordinates (x_i, y_i) from the first slice with the corresponding (u_i, v_i) coordinates from the second slice. The polynomial coefficients are determined using a least-squares best-fit approximation based on linear regression. These coefficients define the transform necessary to reposition the pixels in the second slice so they are aligned with those of the first slice.

A key factor of this method is the use of a single oblique slice extracted from each volume dataset to be registered. This requires the user to select all the landmarks, later to be used to register the images, from the available structures lying on a single plane. In effect, this group breaks the registration process into two steps. First, they determine that portion of the transformation required to display corresponding oblique slices from each volume, each possessing anatomic landmarks and/or external markers that may be selected. Second, following landmark selection from those available on the slices, they determine the remainder of the transformation required to align the landmarks in that plane.

Another group, Tiede *et al.* (32), uses a combination of both the preceding techniques to perform image registration. Like the Philadelphia group, they build a three-dimensional volume representation of the scanned objects from the two-dimensional slice data. They then display surface renderings of the two volumes for the operator to select landmarks. Once suitable landmarks have been selected, however, they, like the New York group, use polynomial warping (which they refer to as polynomial matching) to obtain the registration transformation. Extending the linear equation system to account for the third dimension, they also obtain the polynomial coefficients from a least-squares error best-fit algorithm.

A common characteristic of these structure-based methods is that a considerable amount of user interaction is required, primarily for the purpose of landmark selection or to specify parameters for constructing an oblique slice. In addition, to successfully and accurately register images, this user interaction must be performed by a highly skilled

radiologist or anatomist. Finally, the two methods used to actually determine the registration transformation, either principal component analysis or polynomial warping, entail significant computational expense (19:26).

On the other hand, key benefits of this technique are its accuracy and flexibility. Manually selecting and matching landmarks "... provides the most accurate transformation possible, with uncertainties given only by scan parameters (pixel size, slice spacing, artifacts, etc.)," although this is only true "... if the landmarks can be accurately located [in both scans]" (12:469). Also, a great number of medical imaging sessions are ordered for diagnosing pathological conditions of the thorax or abdomen, where it may be difficult (or impossible, in the case of a physiological modality scan such as SPECT) to always locate the same anatomical landmark in both images. The application of external markers, with a known relationship to almost any anatomical feature of the patient's body, thus allows an effective image registration. A downfall, however, is that this eliminates from consideration those data acquired before it was known external markers would be required (i.e., previous medical imaging scans) (16:818) (26:26).

2.2.4 Surface Identification Techniques. This section presents a detailed description of two surface-based image registration techniques. Since these methods are fairly widely used, it is useful to note the important similarities and differences between them and the structure-based techniques. As in the previous section, I outline the sequence of events followed to register images in this manner, starting with the data displayed to the user, the primitives used for image registration, and the registration algorithm itself. These two techniques are primarily differentiated in this section based upon the image registration algorithm employed, as introduced in Section 2.2.2.4.

2.2.4.1 Least Square Optimal Search Algorithm. Probably the most popular and widely used image registration technique is Pelizzari's *et al.* optimizing best-fit approach to surface-matching, developed at the University of Chicago. In one sense, this technique may be thought of as an extension of the previously described structure-identification method, where a large number of landmarks is used (12:469) (41:448). Their process determines a the registration between models specified by hundreds of points on a surface instead of a handful of landmarks (16:820). This, however, overly simplifies this group's process of generating surface contours from two different scans, then optimizing the fit of one with respect to the other, thus deriving the rotation and translation transformations

required to correlate the data sets. The following briefly describes the method's highlights (additional details are in (26:21–22) (16:819–820) (22:260–261)).

1. The first step in the process involves the creation of an outline of the desired surfaces to be used for image registration. The series of slices composing each scan's data set are individually processed by a number of algorithms designed to extract the closed curve, or contour, formed by the intersection of the three-dimensional surface with the slice plane (12:468). The sequence of contours forming the surface outline is sometimes referred to as a "ringstack" (8:321).
2. One of the two ringstacks, usually that corresponding to the scan with the greater spatial resolution or possessing the most data samples, is considered the "head" model in an analogy that likens the surface-matching image registration process to placing a custom-fitted, rigid "hat" onto the previously mentioned "head." Some 200 - 300 points belonging to the other ringstack are then extracted to serve as "hat" points. The models are also corrected for known differences in scanning parameters, such as scan pixel size.
3. The ultimate objective of the actual image registration process is to seek the optimal transformation of the "hat" to the "head" surface, measured in terms of a residual error function. The process begins by calculating and storing the root mean squared distance from a selection of points on each slice's "head" surface contour. Then, for every estimate at a new set of transformation parameters, the algorithm iteratively:
    - Determines the root mean squared distance from the "head" centroid to every "hat" point.
    - Compares that "hat" point distance with the value of the corresponding "head" surface distance, projected along the same ray connecting the centroid with the "hat" point.
    - The difference between these two distances is the residual error value. This value is accumulated for all "hat" points in that trial, after which the accumulated error is used to define the next geometric transformation to be used.
4. The cycle is repeated until the residual error function can no longer be reduced, at which point the algorithm has converged upon the optimal surface match, thus defining the registration transformation.

Early contour extraction methods employed by this group occasionally required some user interaction. In cases where contours were of the interface between the brain's external surface and the skull's inner surface (the 'inner table') (41:449) (12:469), it was necessary to specify a seed location from which to begin boundary following (14:46). In other cases, where the contours followed the external skin surface, it was sometimes necessary for an operator to correct erroneous contour segments in the vicinity of the ears, nose, and mouth (16:819). Recent improvements to their algorithm, such as employing 'expert knowledge' about CT, MR, and PET images and fuzzy reasoning (in the case of MR images), have allowed the team to completely automate this portion of the image registration process (22:260).

2.2.4.2 Moment Matching Algorithm. In a departure from the Chicago group's technique, Fellingham *et al.* (8:323–324) (12:468) describe the approach taken in the CEMAX-1000 medical imaging system, produced by CEMAX Medical Products, Inc. That system, while employing surface-based techniques, uses a different method to determine the registration transformation. The appropriate surface contours (such as the brain/skull interface) are automatically extracted from each volume's two-dimensional slices, as was the case in the previous method. Their algorithm then uses these contour data to determine the objects' centers of mass, which Fellingham *et al.* also refers to as the objects' first order moment. Their algorithm also determines from the surface contours the orientation of what this group terms the 'tensors of inertia,' or second order moment, for each object, from which they compute the 'principal axes of inertia.' The two volumes' centers of mass are related by a translation and the two sets of principal axes by a rotation, and from this the necessary transformation is obtained.

The success of surface-based techniques is predicated on the assumption that common closed surfaces may be detected and used to generate the necessary contours for each scan. This is fairly well demonstrated in scans involving the skull and brain (especially in CT scans, which clearly portray bony structures). However, while this method (with modifications) can be applied, in principle, to organs other than the brain (22:259), structure-based techniques are more successful with the soft tissues and complex anatomy found in the abdomen (19:21). Also, in the case of registering PET images with CT or MR images using these methods, the image registration's accuracy is typically on the order of several millimeters. While this can lead to a noticeable image misalignment of one to two display screen pixels, many researchers and clinical practitioners deem this reasonable (16:820) (26:21) (14:46).

In this technique's favor is its ability to perform automatic and retrospective (after the fact) registrations. Structure-matching techniques involve the specification of anatomical landmarks, requiring a human expert to manually select those structure points, a time-consuming and error prone process. The automated surface-tracking method just described avoids such use of a skilled anatomist's valuable time, and since an entire surface is used, it is less susceptible to error from misplaced individual surface points (14:46) (16:820) (41:449). This method also benefits from its avoidance of external markers and stereotactic frames. These techniques are considered impractical in clinical situations because such special efforts must go into acquiring the medical images. Also, the use of external markers implies an intent to use those devices for registration sometime in the future, an intention not always known at the time the images are taken. This retrospective approach therefore allows image registration of medical imaging data acquired without such special procedures (26:26) (16:818).

2.2.5 Stereotactic Frame Technique. The image quality and resolution of medical images generated by MR, PET, and SPECT have steadily and dramatically improved since their introduction. Most image registration techniques, however, currently lack the precision and accuracy for many diagnostic and therapeutic applications. Prime examples of this are in the fields of radiation treatment planning (see (31)) and neurosurgery. Also referred to as the *fixed frame technique*, this method of image registration is accomplished by affixing a rigid wood or plastic frame to the patient's head. This serves both to immobilize the head and provide a fixed reference system for use both in image registration and actual treatment. Using these frames, the required accuracy can be achieved while allowing the correlation of data from multiple imaging modalities.

As one example, Computer Assisted Stereotactic Neurosurgery (CASN) uses CT, MR, and other medical imaging modalities to plan and pre-operatively simulate neurosurgical procedures. Intraoperative information may also be provided to the neurosurgeon regarding the placement of probes and electrodes introduced into the patient's brain (42). When attached to the patient's skull, the stereotactic frame provides a common reference system (*stereotactic space*) against which medical imaging scans are registered, providing the submillimeter accuracy required for these procedures. Using this technique, the correlated CT and MR images can visualize cerebral blood vessels and other vital brain areas that must be avoided when performing biopsies, electrically stimulating or destroying tissues deeply seated within the brain, or when placing radioactive seeds for certain types of radiotherapy. With the advent of this technology, these risky procedures have benefitted

from greatly increased accuracy while also reducing the patient's time in surgery. Among the results are fewer complications (such as intracranial bleeding) and, since the operative access points through the skull can be made much smaller, less chance of infection (28).

2.2.6 Anatomic Atlas Technique. The last image registration technique to be examined uses a series of MR images to build a generalized atlas of the brain, defining a number of standard regions of interest by their idealized shapes and relative locations. This atlas is then used as an intermediate reference system against which other clinical evaluations are compared (10). In this process, the patient undergoes MR and PET studies while wearing a stereotactic frame, as described in the previous section, which is then rigidly attached to each examination table. Following the medical imaging sessions, the patient's MR images are registered with slices from the reference atlas by overlaying atlas templates of each region of interest onto the patient's MR image. A radiologist or neuroanatomist, interacting with a graphical display system, then performs a variety of manual operations on each region's boundaries, shaping it to match that structure's unique conformation for that patient. Since the MR and PET data sets were obtained while wearing the stereotactic frame, they may be related to one another and consequently to the region of interest templates customized for that patient. It is therefore possible to overlay the modified regions of interest onto images of the PET data, very accurately localizing key brain metabolic characteristics to specific structures. This capability is important when determining the extent to which a tumor has invaded a particular portion of the brain, or when assessing stroke induced cerebral damage (6). A variation of this method includes earlier work by Bajcsy *et al.* (1), in which image processing techniques were employed to perform edge detection on the MR images, easing the operator's task of fitting and deforming the regions of interest to the patient's MR data.

2.3 User Interfaces

Image registration is inherently a visual process. This statement is especially true for structure-based techniques, which rely so heavily on the user's accurate selection of corresponding landmarks in displayed images. While some applications may register and use data directly, without displaying an image to the operator or clinician, this is not generally the case. This section examines several general user interface issues, providing a basis for several of my design decisions presented in Chapter 3.

2.3.1 Purpose of a User Interface. The user interface's purpose is to aid communication and interaction between the computer and its human user. The communication forms a dialogue between the user and the machine, and is intended to hide machine specific concepts and idiosyncrasies from the user. By abstracting the system's operation in this way, the user can focus on *what* the machine is to do, rather than be concerned with *how* the machine accomplishes its task (20:49). More importantly, the user interface simplifies the user's understanding of the system through the use of well known metaphors, such as windows, pull-down menus, and control panels. This generally results in increasing the user's productivity by allowing them to provide direction to, and receive responses from, the computer in an easier, more consistent form. Key to this is the requirement that the user interface allows both the computer and the user to communicate with one another using vocabulary drawn from the problem domain (34:259–261).

2.3.2 User Interface Design. A system's user interface is more frequently becoming the primary factor in how a user views an application's usefulness. The following are generally accepted as true (34:258) (9:Chapter 9):

- Well-designed and implemented user interfaces can add flexibility and increase user productivity.
- Poorly designed or implemented user interfaces can allow erroneous data entry, the omission of vital data from the program's display, cause user frustration, or loss of productivity. Under such circumstances, a potentially valuable application may receive diminished use, or may even be abandoned.

Until relatively recently, user interface design issues were neglected. Now, however, the user interface is recognized as a critical component of system design. In fact, the amount of design effort devoted to the user interface has grown steadily as system designers have come to recognize its important role. There are now many cases where the lines of code to implement the user interface outnumber those for the application (20:50).

In general, user interface design must be *user-centered*, focusing on the user's background, training, and level of expertise with computer systems. User-centered designs embody the following principles (34:259–260) (20:52):

- They are designed to meet the user's needs and abilities, and do not force the user to change the way they perform their work. As one example of this, the terminology

used in the design comes from the user's normal language, with consistent meanings for objects being manipulated by the computer under the user's direction.

- They provide users with the feeling that they control the computer's actions, as well as navigational aids to understand where they are in the program or with respect to the data being operated upon.
- They use consistent and predictable menu formats, data display conventions, and operator command procedures. They provide for clear and concise user requests and system responses.
- They provide built-in, readily available, user-friendly help facilities. The amount of help available from the application is user and context sensitive, matching the information provided with the user's level of expertise and the application procedure being attempted.

Such designs reduce the emphasis on the technological principles underlying human-computer communications. This recognizes that the computer's capabilities are simply a means to an end, reinforcing the view of the computer as a tool, an extension of a human's ability to interact with the information being presented to them (34:260) (20:57) (9:Chapters 8 and 9).

2.4 Conclusions

Considerable debate persists over the relative merits of the various image registration techniques. I, however, believe that *all* of these methods are valuable, complementary tools available for the clinical diagnostician's evaluation and use. For example, neurological studies might best be conducted using Pelizzari's *et al.* automatic surface-matching techniques. On the other hand, the thorax and abdomen's complex and asymmetrical structures demand the more flexible application of external markers coupled with a structure-matching approach as advocated by Maguire *et al.*

The processing requirements for advanced computer graphics techniques (such as volume rendering) and the quantities of data produced by modern medical imaging systems tax today's most powerful computers. Despite coming advances, the growing amount of data available, and the medically useful operations that will be performed on those data, will probably continue to press computers and software to their limits. This promises to add to the already high burden placed upon the user's of these systems, pointing up the necessity for a well-designed user interface.

2.5 Summary

The literature review to support this research covered two areas. First, the clinical value of techniques such as composite and comparative analyses were examined, and the various forms of image registration were compared. Second, owing to the user interface's importance in as visual a process as image registration, issues were presented centering about the necessity for strong design emphasis on this frequently neglected system component. In the next chapter, the requirements definition for, and the design of, an image registration system are detailed.

III. Image Registration System Design

This chapter describes the design methodology employed to incrementally devise an image registration system. The first section introduces the analysis and design process used during this effort, and includes a description of the symbology used to represent certain design aspects. Next I provide a brief architectural perspective of the overall system design. I then present a number of major design decisions encountered and resolved during this development effort. Finally, I discuss in greater detail the resulting image registration system's four major components.

3.1 Introduction

The object-oriented paradigm was followed throughout this development's analysis, design, and implementation. The overriding motivation for this decision was the requirement to build an image registration tool amenable to further development and evolution (see Section 1.4). In general, as a system matures, its required functionality may change and it may even have to interact with new objects. The original objects themselves and their interrelationships, however, tend to remain relatively stable. An object-oriented approach may therefore make the image registration system inherently more maintainable than if it had been constructed using classical, functional techniques (4:97). The results of the object-oriented analysis effort are recorded in a series of Coad diagrams. Figure 3.1 illustrates the symbology used in these diagrams.

3.2 Overall System Architecture

The next two sections present the results of the context and problem analyses conducted to learn more about the image registration problem. Once design and implementation had begun, however, deficiencies in my first analyses became apparent. Revisiting the analysis resulted in a number of significant alterations, not only in the organization of the entities described by the Coad diagrams, but in their substance as well. Examples are:

1. The design decision to display the volume data as slices, not as surface renderings, upon which landmarks would be identified (see Section 3.3.1).
2. The realization that landmarks and a viewing environment were attributes of each volume to be registered.

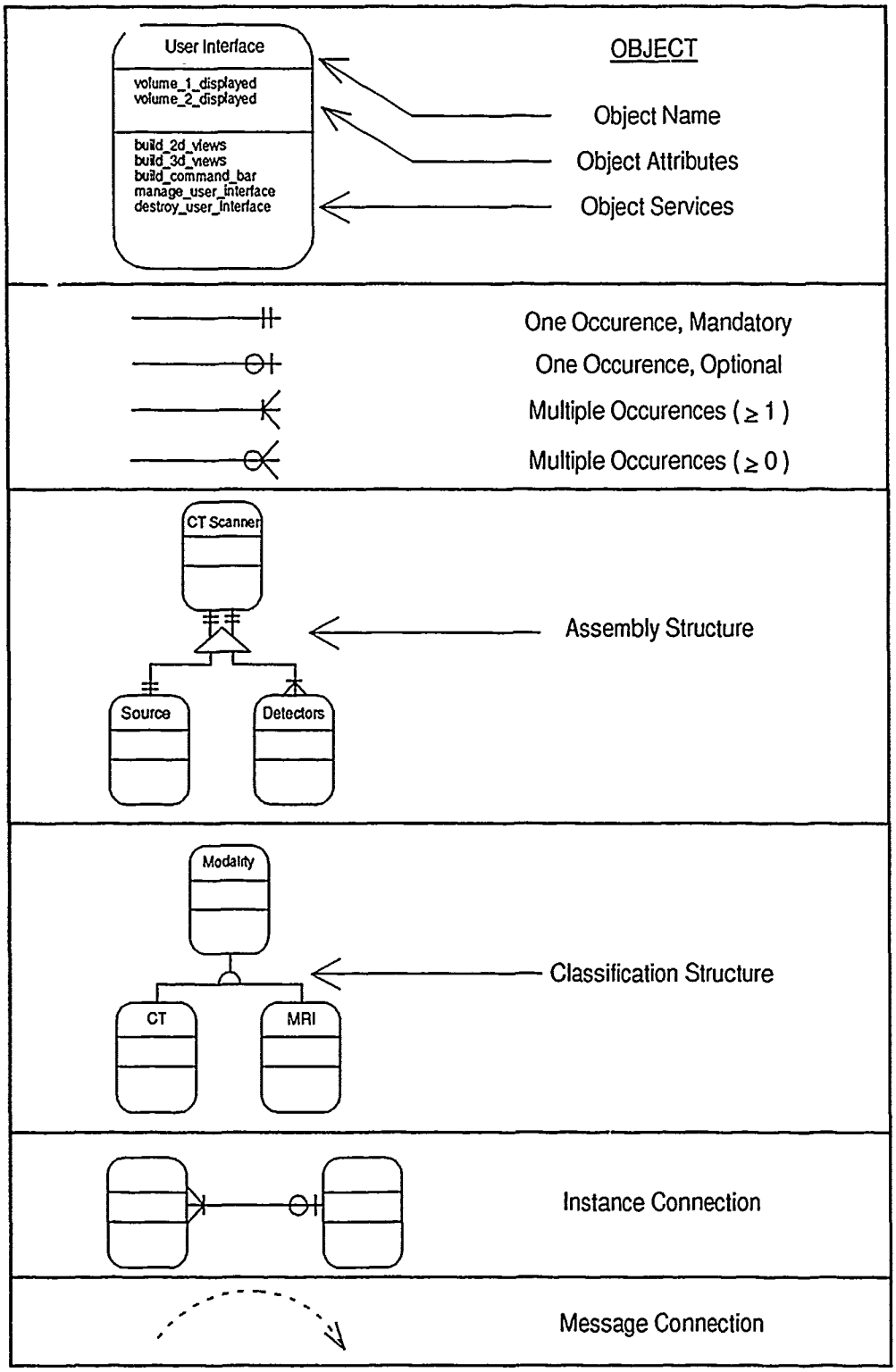

Figure 3.1. Object-Oriented Analysis Notation

3. The realization that specifying how the volume visualization subsystem was to display its final image was really an implementation issue, not a design issue, and therefore inappropriate to consider at this level. In addition, since one goal of the overall image registration system was to be interoperable with any number of post-processing tools, such as a surface or a volume renderer, the design essentially had to be flexible enough to take what it could get.

Based upon the substantive nature of these changes, it was appropriate to re-baseline the system using the updated set of design requirements. To provide a record of this development's evolution, the context diagram resulting from the first analysis is included at Appendix A.2 .

3.2.1 Context Analysis. A context analysis is generally performed to define a proposed system's purpose, and to identify significant system level requirements (30:6). The context analysis for this development was conducted primarily to bound the image registration problem and to isolate the required interfaces to and from the system. For example, prior to this analysis, the user interface was considered as separate and wholly secondary to the remaining image registration system components. One observation during the analysis, however, was that the user interface was very much involved with landmark selection. An adequate user interface, capable of satisfying the requirements for this image registration tool, was not available. It was therefore necessary to proceed with the user interface's development concurrently with that of the image registration subsystem. The context diagram for the overall image registration system is shown in Figure 3.2. The entire needs product derived from this analysis is in Appendix A.3.

3.2.2 Problem Analysis. An object-oriented problem analysis is generally performed to identify the entities (objects and classes of objects) that form the problem space (4:46). A major feature of object-oriented analysis and design is that they use the vocabulary from the problem space to identify these entities. The image registration system's classes, objects, and methods were therefore defined using standard medical image processing terminology. In the first step of this process, the context diagram served as the basis for constructing the subject layer, as indicated in Figure 3.3. In this diagram, the entities and interfaces forming the image registration system and its environment were used to define the most general classes modeling aspects of the problem space. Of particular note is the central role occupied by the user interface, providing further justification arguing for its

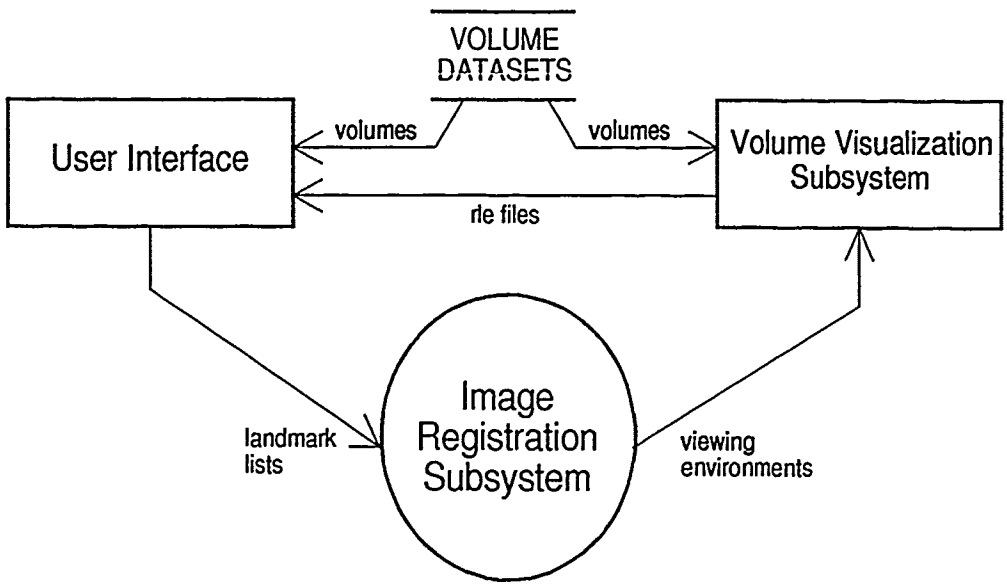

Figure 3.2. Image Registration System Context Diagram

importance to the overall system. In other words, without a well-designed user interface, the image registration system's utility would have been diminished.

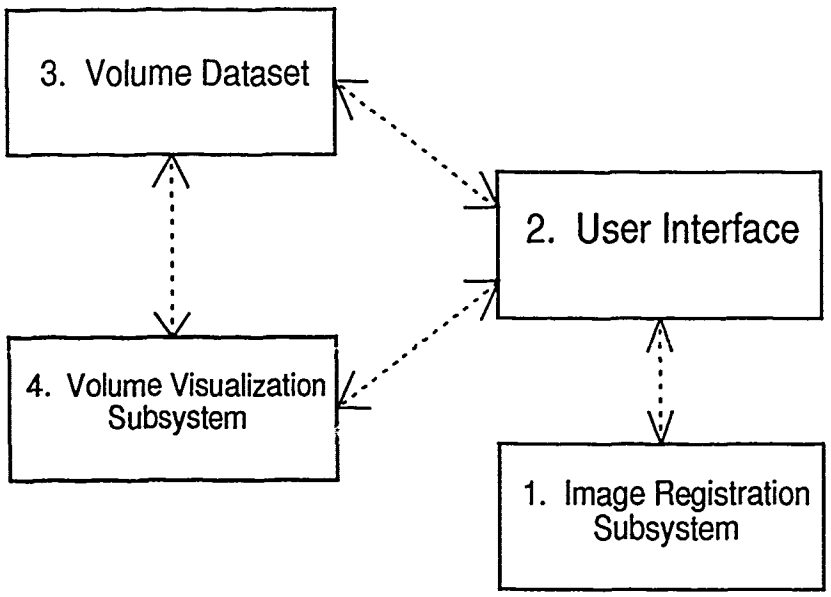

Figure 3.3. Image Registration System Subject Layer

Having prepared the system's subject layer, the remaining analysis tasks were to identify the system's other objects, their attributes and services, and the structures and relationships they form with one another. Attempting to capture all the objects and their relationships in a single diagram, however, would only serve to create a confusing analysis product. In this case, it made more sense to represent the total analysis product as a hierarchy of diagrams, with a single top-level diagram supported by a number of subordinate diagrams, as required. The top-most diagram, shown at Figure 3.4, therefore strongly resembles the subject layer, but with each general class possessing substantially more detail regarding its attributes and behavior. Each subordinate diagram, shown elsewhere in this chapter, likewise provides the additional objects and relationships associated with a particular top-level class.

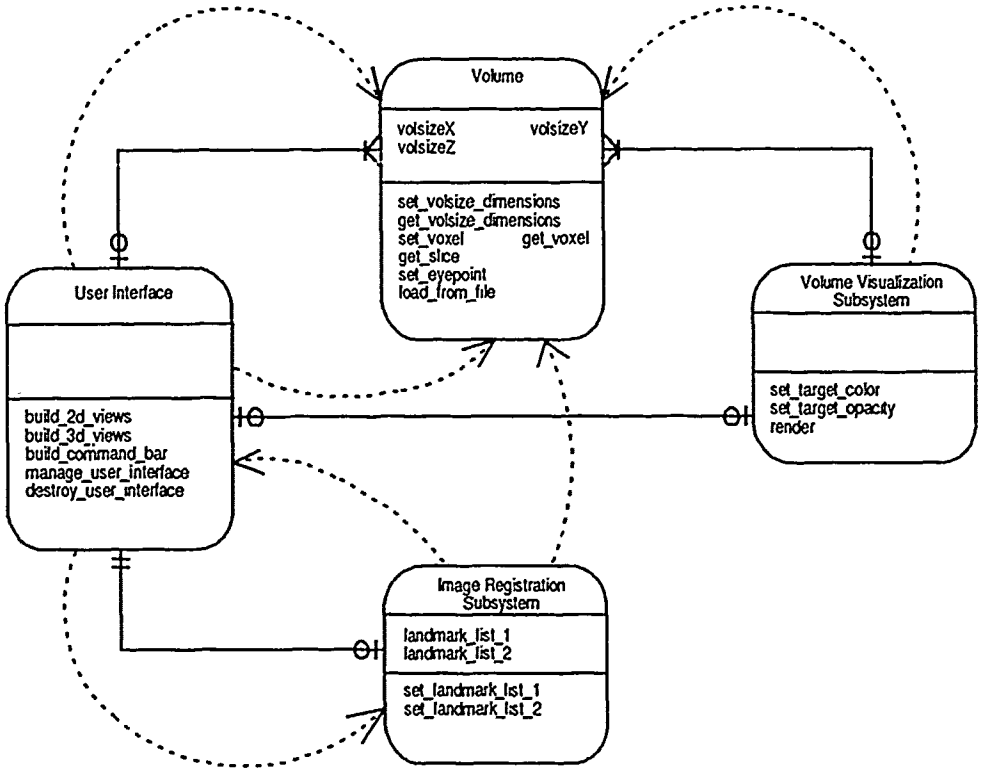

Figure 3.4. Image Registration System Top-Level OOA

3.3 Design Decisions

Once the analyses had been performed, I began the design process, incrementally adding information about each class's characteristics, relationships, and functionality. During this phase of the development, several system options were considered, generally centering around the user interface. Its primary responsibilities, displaying volume data to the user and determining landmarks within that volume, became the driving force behind much of the system's design. The following sections summarize a few of the decisions made.

3.3.1 Data Displayed as Slices. I chose to display the volume data to the user (for later landmark selection) as a series of single slices, as discussed in Section 2.2.3.2, rather than by presenting renderings of some surface contained within the volume (see Section 2.2.3.1). Several factors influenced this decision.

- Surface renderings possess inherent inaccuracies resulting from the estimation of one particular surface within a volume.
- The lack of an immediately available, proven surface rendering package.
- Displaying slice data is easier than generating a surface rendering.

3.3.2 Landmarks Selected from Multiple Slices. As mentioned earlier, the objective of this research was to demonstrate image registration using a number of three-dimensional landmarks identified from various voxels within the volumes to be registered. Crucial to this technique was the requirement that the user be able to unambiguously locate and select any particular voxel within those volumes. I felt this flexibility would enable the user to more accurately select landmarks, and thereby increase the overall accuracy of the image registration. Having already decided the data was to be displayed in slice format, I concluded the system must be capable of:

- Displaying the data so that the user can precisely localize any anatomic landmark or external marker to one specific voxel. In order to meet this requirement, the user interface must display images with sufficient three-dimensional context so that the user can navigate through the volume. By observing a particular voxel from different viewpoints, the user can more easily decide if it most accurately represents the desired landmark. Also, the user is not confined to a single slice in the search for landmarks, thus ensuring those finally selected represent the best possible selections from throughout the entire volume.

- Accurately transforming the desired landmark's screen space coordinates into object space voxel coordinates. Since the slice data displayed represents a one-for-one mapping from screen space pixels to object space voxels, this transformation is extremely accurate and fast. These object space coordinates, the voxel's address in the three-dimensional array of volume data, are then used to actually determine the registration transformation.

To satisfy these requirements, I decided to select a *MultiPlanar Display (MPD)* technique as the most meaningful method to display the medical image slice data. Such multiplanar displays may be thought of as projecting a three-dimensional object, contained within a glass box, onto the box's six faces. If the box is unfolded, all faces lie on the same plane, creating a top view, a front view, a bottom view, a right and a left side view, and a back view.

Because of the display screen's limited viewing area, and the need to simultaneously display views of both volumes to be registered, only four of these six views are presented for each volume, as shown in Figure 3.5. These views determine the slices to be extracted from each volume and projected onto the dedicated viewing areas of the display screen. The four views chosen represent the following standard projections of medical imaging data, as further illustrated in Figure 3.6.

- A coronal view, corresponding to the front view.
- An axial view, corresponding to the bottom view.
- A right sagittal view, corresponding to the right side view.
- A left sagittal view, corresponding to the left side view.

3.3.3 Use the Entire Volume Dataset. The datasets used by this system are saved as a series of files in secondary storage, each file containing the data for a single transverse slice as scanned by a particular medical imaging system. There are two possible ways this data may be loaded into the computer and used for image registration. As explained in Section 2.2.3.2, some researchers, such as Maguire *et al.*, load only one slice of data into main memory at a time, performing the image registration based upon the data present in that one slice. Other groups, mentioned in Sections 2.2.3.1 and 2.2.4.1, load all the slice files into main memory at the same time, using the data to construct an internal volume data structure.

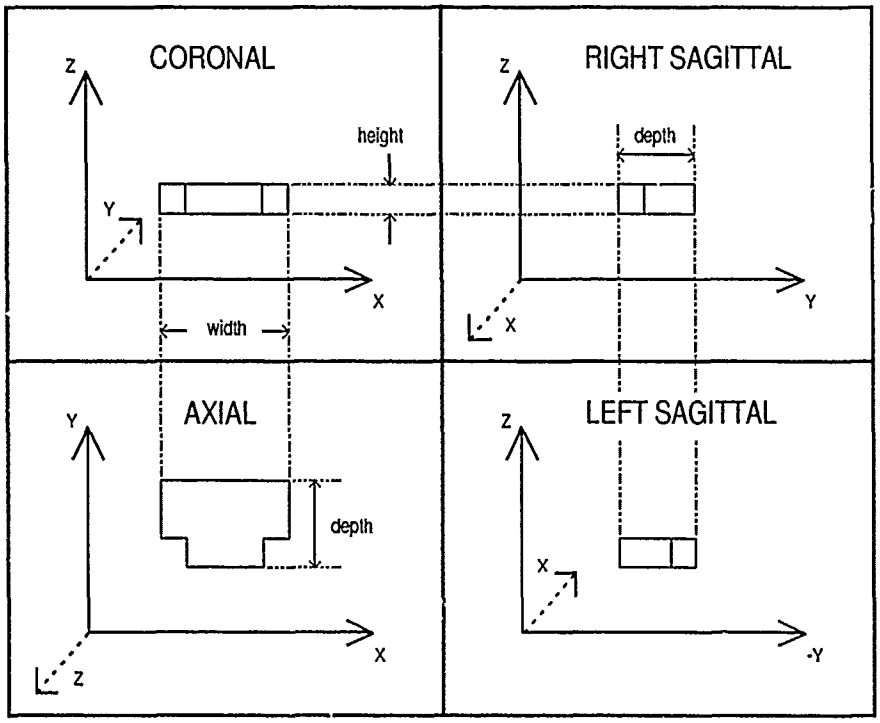

Figure 3.5. Two-Dimensional View Layout

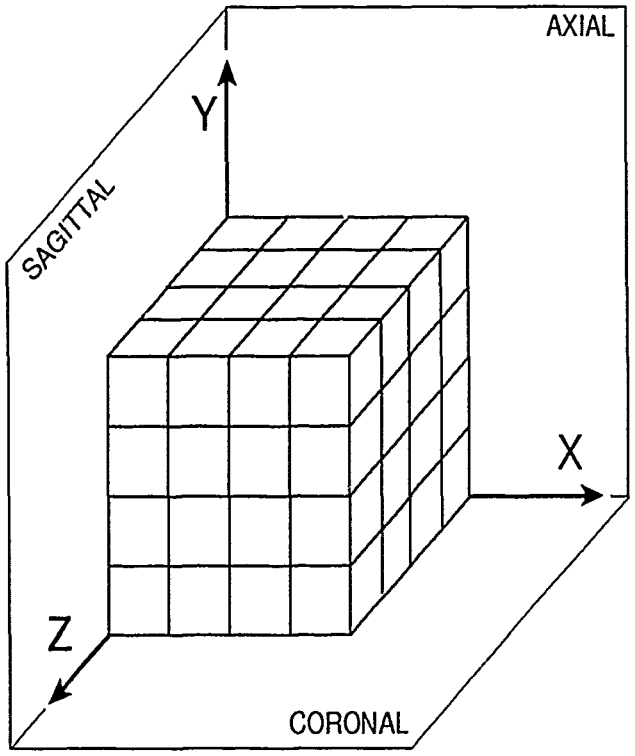

Figure 3.6. Orientation of Standard Medical Projections

I chose to follow the latter of these two practices. This decision was driven primarily by user interface considerations. As stated earlier, a major goal of this development was to provide the user with the flexibility to select the necessary landmarks from a number of different views of the volume. To reduce the amount of time required to select the landmarks, it was necessary to display a number of these slices, interactively determined by the user, as quickly as possible. This required that the entire volume data set, from which all these slices are extracted, be loaded into main memory.

3.3.4 Image Registration Technique. The last major design decision made for this effort was the selection of the actual image registration technique. As stated in Section 1.4, a major goal for this research was to use simple three-dimensional relationships between landmarks to derive an image registration transformation. This registration transformation can then be used to modify the object space eyepoint associated with the volume to be registered. The new eyepoint represents the point in object space where an observer must be positioned so that three-dimensional views of both the reference and registered volumes are aligned.

I chose to approach this problem in a manner similar to the computer graphics technique of mapping a two-dimensional texture onto a three-dimensional object, as discussed by Paul Heckbert (13:57–60) (43:236–237). In this technique, a composite mapping is performed from two-dimensional texture space to two-dimensional screen space. The projection of the three-dimensional object onto screen-space is assigned values, pixel by pixel, mapped directly from texture space. That mapping is determined from the following equation:

$$\begin{bmatrix} u & v & q \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix} \begin{bmatrix} a & d & g \\ b & e & h \\ c & f & i \end{bmatrix} \tag{3.1}$$

In the above, q and i are global scaling factors that may arbitrarily be set to 1. This leaves eight unknowns, a - h, that may be determined by solving an 8 x 8 system of linear equations. These equations are formed using four texture coordinates $\{(u_i, v_i) \mid 0 \leq i \leq 3\}$ corresponding to four screen coordinates $\{(x_j, y_j) \mid 0 \leq j \leq 3\}$ from the projected three-dimensional object.

It is possible to perform image registration with as few as three landmarks, using methods such as Principal Component Analysis (as discussed in Section 2.2.3.1). Also, by

using the angles between each landmark, in addition to their displacements with respect to one another, the registration transformation matrix's individual components may be determined. However, the goal of this research was to focus on performing image registration using only simple linear relationships between the landmarks. I therefore extended the texture mapping technique's use of four *ticpoints* and applied it to the image registration problem in a method hereafter referred to as *Landmark Mapping*. In this method, I chose to use a set of four landmarks $\{(x_i', y_i', z_i') \mid 0 \leq i \leq 3\}$ mapped onto another set of four landmarks $\{(x_j, y_j, z_j) \mid 0 \leq j \leq 3\}$ by the following relationship:

$$\begin{bmatrix} x_i' & y_i' & z_i' & w_i' \end{bmatrix} = \begin{bmatrix} x_j & y_j & z_j & w_j \end{bmatrix} \begin{bmatrix} a & e & i & m \\ b & f & j & n \\ c & g & k & o \\ d & h & l & p \end{bmatrix} \tag{3.2}$$

w, w', and p all represent global scaling factors that may be arbitrarily set to 1. Also, m, n, and o represent factors that, for affine mappings, are set to 0 (13:59). The resulting 4 x 4 matrix

$$\begin{bmatrix} a & e & i & 0 \\ b & f & j & 0 \\ c & g & k & 0 \\ d & h & l & 1 \end{bmatrix} \tag{3.3}$$

is the registration transformation. Its remaining coefficients may be empirically determined using the two sets of four landmarks to solve the 12 x 12 system of linear equations:

$$\begin{bmatrix} x_0 & y_0 & z_0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & x_0 & y_0 & z_0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x_0 & y_0 & z_0 & 1 \\ x_1 & y_1 & z_1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & x_1 & y_1 & z_1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & x_2 & y_2 & z_2 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x_2 & y_2 & z_2 & 1 \\ x_3 & y_3 & z_3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & x_3 & y_3 & z_3 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x_3 & y_3 & z_3 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \\ i \\ j \\ k \\ l \end{bmatrix} = \begin{bmatrix} x'_0 \\ y'_0 \\ z'_0 \\ x'_1 \\ y'_1 \\ z'_1 \\ x'_2 \\ y'_2 \\ z'_2 \\ x'_3 \\ y'_3 \\ z'_3 \end{bmatrix} \tag{3.4}$$

This equation is in the standard format $Ax = B$, which is readily solved using such Gaussian elimination techniques as *LU Decomposition.* Once this algorithm determines the registration transformation, it may be used to complete the image registration process, as discussed earlier.

Constructing the registration transformation matrix in this way allows the introduction of distorting scaling and shearing (affine) transformations, as well as rotation and translation (rigid body) transformations. Some researchers, such as Toennies *et al.*, attempt to limit their transformations to only rotations and translations (40). On the other hand, researchers such as Maguire *et al.* and Ratib *et al.*, actively seek the scaling and shearing components (19) (27). A simplifying assumption made in Section 1.5, that all datasets used by this system must possess the same dimensions, eliminates a major source of scaling (as might be encountered by registering volumes based upon 256 x 256 and 512 x 512 voxel axial slices). However, I chose not to further constrain the registration transformation to only rotations and translations for two reasons. First, such a restriction would require extensive modifications to the above algorithm to prevent the introduction of scaling and shearing components. Second, this would allow for a greater level of experimentation as to the effects of these affine distortions and an analysis of their impact on this technique's clinical utility.

3.4 Major System Elements

In the following sections, each component defined in the subject layer (Figure 3.3) is described in greater detail. As previously stated, two of the four individual components are further described by subordinate diagrams, introducing additional objects and relationships not shown on the problem analysis's top-most diagram. In the case of the image registration and volume visualization subsystems, the objects, attributes, and behavior noted on the top-level diagram possess the right level of detail.

3.4.1 Image Registration Subsystem. The image registration subsystem's primary function is to abstract the details of image registration from the remainder of the system's entities. It was therefore designed to present a common interface to the other components, regardless of the actual methods it incorporates to perform its functions. The image registration subsystem's interface requirements are met as follows:

- *Input.* The input to the image registration system is a list of landmarks, one from each volume to be registered.
- *Output.* The output from the image registration system is a matrix representing the transformation required to register one volume with a reference volume.

Services offered by the image registration subsystem:

- Allow the user to select an image transformation method from those available to the image registration subsystem. The primary method to be used for this research will be the landmark mapping technique discussed in Section 3.3.4. Even so, the subsystem was designed to accommodate additional techniques.
- Apply the selected image registration technique to the lists of landmarks input to the subsystem.
- Allow the user to process the volume visualization system's RLE images, resulting from the requested registration of both volumes, through composite or comparative analysis routines. This creates a single image that demonstrates the desired analysis of the registered volumes.

3.4.2 Volume. The image registration system forms an internal volume data structure for each set of medical imaging slice files.

3.4.2.1 Attributes. A volume is composed of a three-dimensional array of voxels, and so has as an attribute the array's x, y, and z *dimensions*. Each voxel has as an attribute a *density* value, determined by the medical imaging scanner, associated with the corresponding point in patient space. In addition, a slice may be extracted from the volume, and is composed of a two-dimensional array of voxels lying on a plane orthogonal to one of the volume's major axes. Therefore, along with its *dimensions*, the slice also has as an attribute its *orientation* with respect to the volume.

In addition, each volume to be registered has a viewing environment and a list of landmarks associated with it. Each landmark in the list has as attributes the *voxel's address* (the x, y, and z coordinates of the matrix position occupied by that voxel) and the landmark's *type* (e.g., anatomical landmark, external marker) and *name* as identified by the user. The viewing environment has as an attribute the eyepoint's three-dimensional *location* in object space with respect to the volume's origin.

3.4.2.2 Methods and Relationships. Each of these objects possess methods to modify or retrieve their various attributes. In addition, the volume object has a method to load the data from the computer's secondary storage into the data structure. The voxel, slice, and volume objects, as well as the volume's landmark list and view environment objects, are related as illustrated in Figure 3.7.

3.4.3 User Interface. The image registration system's user interface was designed to assist the user by performing the following tasks.

1. Data Display. Projections of the two volumes to be registered, along the three major axes in scanner space, are displayed to the user in four dedicated two-dimensional views, as discussed in Section 3.3.2. The user can choose any slice, from the entire sequence of slices along each axis, to display in these two-dimensional views. A rendering of the entire volume can be displayed, as discussed in Section 1.2.15, in a dedicated three-dimensional view.
2. Landmark Identification. Each slice displayed in one of a volume's two-dimensional views may contain any number of readily recognizable anatomical structures or external markers. The user's task is to unambiguously correlate a potential landmark in one view with the same structure or marker observable in views from the other volume. Once the user decides a particular landmark is acceptable for image registration, the corresponding voxel must be identified to the system. The user does this

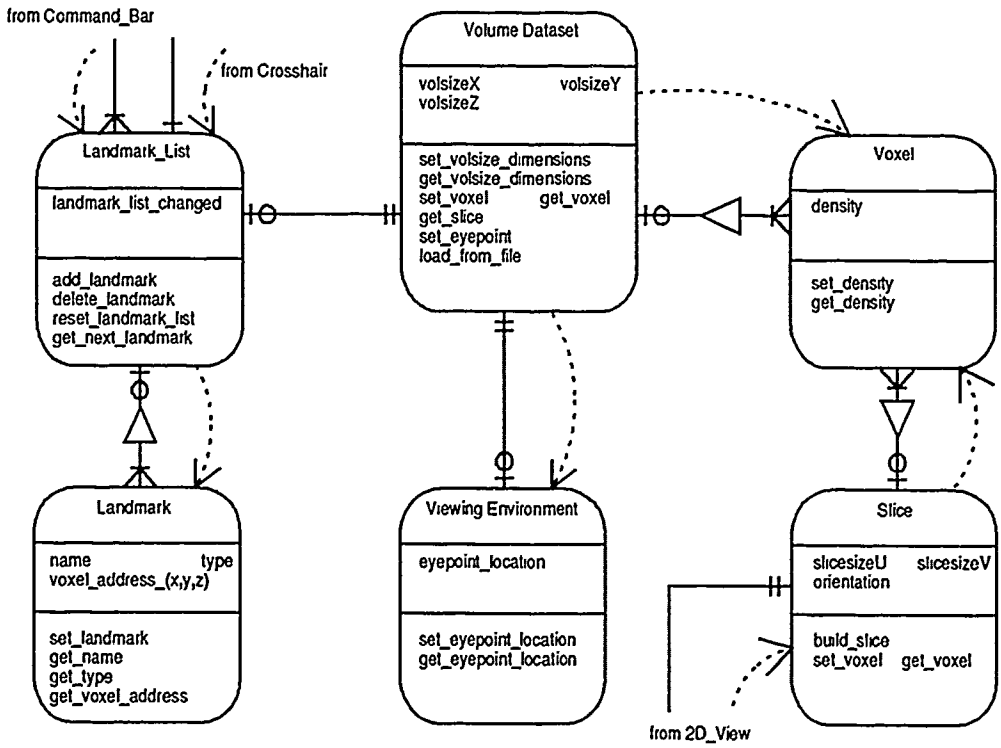

Figure 3.7. Volume Class OOA

by overlaying a crosshair onto the landmark and designating the voxel's screen space coordinates using the system mouse.

This dual purpose is therefore reflected in the user interface's attributes, methods, and relationships.

3.4.3.1 Attributes. To carry out the above functions, the user interface is associated with the two volumes to be registered. For each of the two volumes, it generates and manages four two-dimensional views (as described in Section 3.3.2) and a single three-dimensional view. Each two-dimensional view has as attributes the *orientation* of slices it can display and the *sequence number* of the slice currently being displayed. Every view is associated with a crosshair display screen cursor, controlled by the system mouse, possessing the attributes of the display screen *pixel coordinates*, and the corresponding *voxel coordinates*, currently being pointed to. While the crosshair controls the user's interaction with a view, menus control the activities occurring within the view. The final component, the command bar, controls the overall image registration system and displays information regarding the landmark lists as they are developed.

3.4.3.2 Methods and Relationships. The user interface possesses methods to build the two-dimensional and three-dimensional views, as well as to build the command bar. Once these objects have been constructed, another method manages the user interface. Finally, when the user has finished with the image registration system, another method destroys the user interface.

Each user interface view, and the command bar, is a separate window controlled by, and uses a number of methods to communicate with, the hardware platform's window manager. Each object possesses a method to construct and destroy itself, opening and closing the windows with which the user interacts with the system. Between those two events, each object has a method to interact with the system mouse, thus allowing the user to control the activities occurring within each view or the command bar. Each object also has methods to modify and refresh, as needed, the images and text displayed within the view or command bar. Finally, each object has a method to capture an image of itself and save that image, as a file on secondary storage, for later display.

In addition, each two-dimensional view also provides methods to allow the user to navigate through the volume along the axis perpendicular to that view's orientation, displaying slices sequentially or at random. Each three-dimensional viewing area also provides

methods to allow the user to display the RLE images created of that volume by the volume visualization subsystem. The crosshair and menus associated with each two-dimensional view provide methods to add landmarks to, and delete them from, the landmark list associated with each volume being registered. The user interface (and its views, command bar, crosshair, and menus), as well as the hardware platform's mouse and keyboard, are related as illustrated in Figure 3.8.

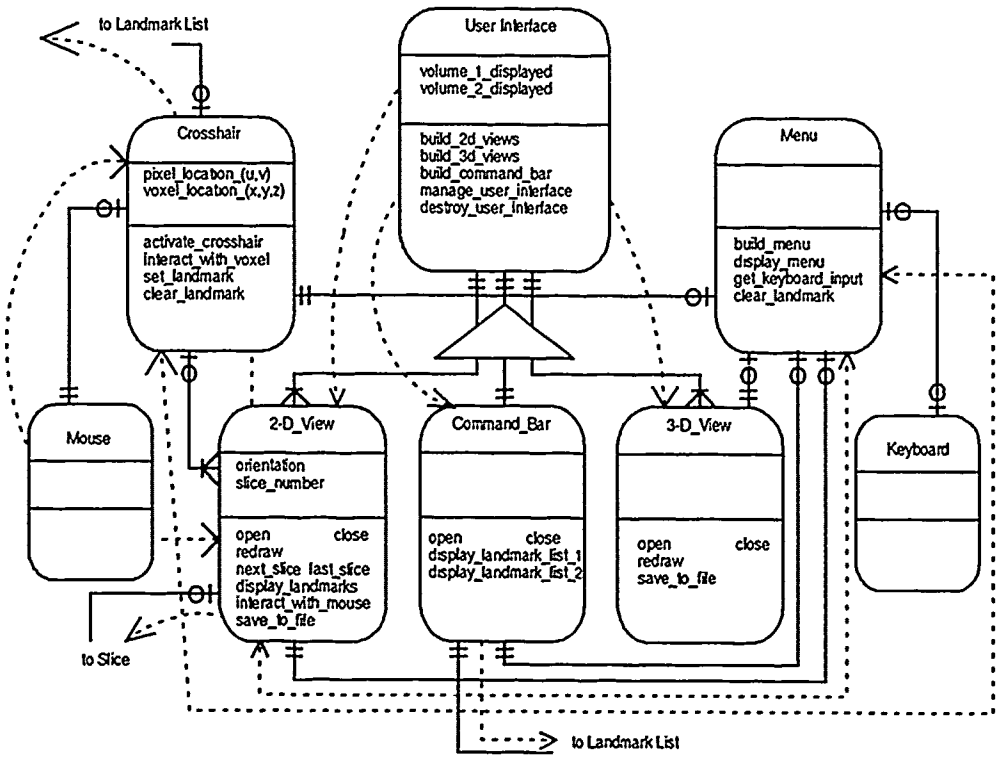

Figure 3.8. User Interface Class OOA

3.4.4 Volume Visualization Subsystem. As the final step in the process, the registration transformation generated by the image registration subsystem must be applied to the viewing parameters for one of the volumes being registered. As indicated in Sections 1.4 and 3.2, this would allow corresponding views of the registered volumes to be displayed as three-dimensional surface or volume renderings. The image registration system was designed to use a surface rendering tool developed by Capt Parrott (25) and a volume rendering tool developed by Captain Brightbill (2) under separate AFIT research efforts. A static interface between these tools is the only one possible at this time. Both visualization

tools were designed for, and implemented using, the Sun and SPARC architectures, which is incompatible with the architecture used for this research (see Section 4.2.1 for a more complete discussion of this system's platform requirements). The volume visualization tools' interface requirements are met as follows:

1. *Input.* The image registration system builds a command line argument from the same file information used to initially load the datasets for this system. This is done for each volume used for the registration process.
2. *Input.* The image registration system also constructs a control file, containing default parameter settings, for each volume used in the registration process. These default values include opacity and color values, as well as an initial eyepoint position, to be used by the volume visualization tool to build the desired images. The output of the image registration subsystem, the calculated registration transformation, is then used to modify the registered volume's eyepoint position.
3. *Processing.* Using the command line arguments previously constructed, the image registration system executes operating system routines to initiate the volume visualization tool's processing of each volume.
4. *Output.* The output from each volume's rendering is saved as an RLE file onto secondary storage for later use.

Services offered by the volume visualization subsystem:

- Allow the user to select either the surface rendering or volume rendering tools for displaying the final images of the registered volumes.
- Alert the user when the resultant RLE files have been created and stored by the selected volume visualization tool.

3.5 Summary

This chapter presented a brief overview of the analysis and design process used to better understand and design a solution for the image registration problem. Following a brief overview of the system's design, I discussed the major design decisions made for this development effort. Finally, I described the image registration system's design from the standpoint of its four major components. The design advanced in this chapter served as a starting point for the solution's implementation as presented in the next chapter.

IV. Image Registration System Implementation

4.1 Overview

This section discusses the process taken to translate the analysis and design products presented in the previous chapter into an operational system. The chapter begins with a brief description of the hardware platform and software development environment and philosophy under which this system was developed. I next outline the sequence of events taken to successively add functionality to the system. The chapter concludes with an overview the system's input requirements and operation. This chapter also demonstrates the user's interaction with the system and the results of the image registration process.

4.2 Implementation Environment

4.2.1 Hardware Platform. The image registration system was developed for use primarily on the Silicon Graphics, Inc. (SGI), Power Series 4D/310GTX computer graphics workstation located in the AFIT computer graphics laboratory. In addition, the system is operational on the 4D/85GT or the Personal Iris 4D/35 also available in the lab, although at greatly reduced performance.

4.2.2 Software. Whenever possible, I used existing software, or software being developed for other applications, rather than creating my own packages to provide the same functionality (e.g., an already developed doubly linked list data structure for the list of landmarks, and a modified LU decomposition algorithm for the image registration subsystem). In addition, I modeled many aspects of the software generated for this research after a widely used general purpose rendering (GPR) package previously developed in the AFIT computer graphics laboratory. Several ideas from that system were incorporated into my own design, such as abstracting viewing environment parameters into a separate class. These reused design elements and software components will reduce the amount of effort required of future researchers to understand this system's design. In turn, this will hopefully improve their results as they use, modify, or extend this software for new applications.

C++ was chosen as the computer language best suited for this development. The following factors were taken into consideration for this decision.

1. There are many computer graphics software tools available that have already been written in C, a language similar to and compatible with C++. Many of these tools are

already available on the hardware systems for which the image registration software was developed.

2. There is a great quantity of medical image processing software already available in the medical and academic communities, the preponderance of which is written in C and C++.
3. C++ is quickly becoming accepted as the *de facto* standard for many object-oriented design and development efforts, the approach chosen for this project.
4. C++ is the AFIT computer graphics laboratory's standard high-order programming language.

4.2.3 Implementation Philosophy. I implemented the image registration system using an incremental, top-down approach. An initial system was assembled from the specifications derived for each of the major classes of objects identified from the previous analysis and design efforts. The primary goal of this step was to establish inter-object relationships and each object's service methods, allowing me to resolve any interface issues before adding the complexity of functional code. Following that, additional layers of functionality were successively added to the system. Between each increment, I performed some limited testing to ensure the desired features had been correctly implemented and previously functioning software had not been corrupted.

My overall goal was to secure an operational system as quickly as possible. This allowed me to experiment with different ideas and new functionality on a continually working system. In addition, the program's earlier versions were used extensively in researching various aspects of the volume datasets being used for this and other AFIT research efforts. Most importantly, this allowed me to demonstrate the user interface and obtain feedback regarding changes to the data's presentation or necessary additions. This practice is commonly referred to as *rapid prototyping*, and is widely used when there is some uncertainty as to the system's true requirements (15:272 273). Thus, I was able to satisfy the final system's requirements through a logical succession of progressively more advanced iterations.

4.3 Image Registration System Development

The image registration system's development took shape over six distinct phases. The first and second phases were concerned with obtaining a minimally functional user

interface and displaying medical image data. The third and fourth demonstrated the ability to interact with the system and identify landmarks. The final two phases implemented the actual image registration processing and displayed the results to the user as surface or volume renderings. The following sections recount these activities in greater detail.

4.3.1 User Interface Construction. The first step in the image registration system's development was to build the axial, coronal, and sagittal two-dimensional views discussed in Section 3.3.2. At this point, three-dimensional viewing areas for each volume and a command bar to control system operation were created but were not yet functional. The cornerstone to this portion of the design was the availability of a general purpose window manager built for another AFIT research effort by Captain Simpson (33). His object-oriented window management system integrated together the SGI's extensive graphics library of customized routines for performing low-level window, mouse, and cursor control. The window management system provided a *Text Window* class, which created a window capable of displaying static and dynamic text strings. I derived from this class a *User Interface Window* class that combined its parent's capabilities with attributes and methods common to my user interface's objects. It was from this user interface window sub-class that I derived the two- and three-dimensional view and the command bar objects. C++ specifications for these classes are in Appendix B.1.

Another aspect of this phase was the integration of the high level classes identified from the analysis and design effort with the main driver program. This simple sequence of instructions first constructed instances of the volume class by loading into memory the requested slice data located in secondary storage. Then, in addition to constructing the previously mentioned user interface object, it created rudimentary objects from the image registration and volume visualization subsystem classes. Once this initial system was fully operational, it formed the skeleton upon which the remaining image registration system elements would be attached.

4.3.2 Displaying Slice Data. Following development of the basic user interface objects, it was necessary to begin displaying actual data to the user through the interface. This involved two separate steps, obtaining slices of data from the volume object, and displaying the slice in the appropriate two-dimensional view. In the first step, each two-dimensional view has associated with it a slice object storing the data to be displayed within the view. Each slice has an orientation attribute that dictates how the slice may copy data from the volume. In other words, a coronal two-dimensional view only displays a slice of

data extracted from a coronal plane through the volume. The specific orthogonal plane of voxels accessed is determined by the two-dimensional view's slice number attribute, which tracks the view's depth in the volume as the user moves it along the viewing axis. The two-dimensional view's depth is controlled using mouse buttons, with which the user can increment or decrement the view's slice number and cause either the next or the last slice to be displayed. In addition, the user has the option of directly requesting a slice from elsewhere in the sequence. C++ specifications for these classes are in Appendix B.2.

I arbitrarily chose to select and display the slice one-third of the way into the volume along each view's axis every time the user interface is created. In addition, I also decided to maintain in memory the slice just before and just after the slice being displayed. This would allow the view to rapidly display the next or last slice in the sequence while the system copies the new next or last slice. This was a trade-off in that no user interaction would be possible until the new slice had been extracted, but at least the user would be able to spend that time examining the desired slice for landmarks. The alternative would have been for the user to watch a blank screen or the old slice while waiting for the system to extract and display the desired slice.

The second aspect of this phase was the actual display of slice data in the two-dimensional views. Knowing the view's dimensions as well as those of the slice, referenced in screen space by (u, v) coordinates, it is possible to center the slice within the view. In addition, for those slices that exceed the view's dimensions, the data on the slice's fringes is clipped away before being displayed. Each voxel in the slice is then sequentially accessed, and its density value is used to set the color of the corresponding display screen pixel. Implicit to this operation is the simplifying assumption that the volume's voxels have the same dimensions as the display screen's pixels. At this point in the development, the user interface was able to display four orthogonal views for each volume, allowing the user to completely examine the volume dataset's appearance.

4.3.3 Interacting with Slice Data. The next phase in the implementation was to provide the ability to interact with the slice data being displayed within each view. The first step was to present indicators in each view to orient the user within the volume. Each two-dimensional view's slice number is used to display the common voxel coordinate lying in the slice plane (e.g., the z coordinate in the axial view). The mouse-driven system cursor's position is regularly sampled, and its screen position is translated into object space coordinates, thus providing the remaining coordinates for the voxel lying immediately

beneath the cursor. Thus, as the user moves the mouse, the cursor moves across the slices displayed in each two-dimensional view, and a voxel address display for each view is continually updated.

The second element of interaction was to allow the user to select a single voxel from the slice being displayed in each two-dimensional view. To acknowledge the selection, the view displays the density value for the voxel immediately beneath the cursor when the user depresses the right mouse button. While the mouse button is held down, the voxel address display and density value for that view are frozen regardless of any mouse motion. With this addition to the system, it was now possible to begin sampling individual voxels contained anywhere within the volume dataset.

4.3.4 Identify Landmarks. The next step was to capitalize on the functions provided by the user interface to create and track landmarks in each of the two volumes to be registered. Two new objects were introduced and another was modified to support this system function. The first new object was a crosshair associated with each view to replace the SGI's system cursor. The crosshair was designed to assist the user to locate and select a single desirable voxel from its corresponding pixel. The second new object was a list of landmarks associated with each volume. Each landmark added to the list represents a single voxel within that volume, and stores that voxel's coordinates and a short, user supplied identifier. C++ specifications for these classes are in Appendix B.3.

Each two-dimensional view's crosshair provides a menu driven interface with the landmark list associated with the same volume as the view being interacted with. Menu options allow the user to add the landmark to the list, to delete a previously designated landmark from the list, or to clear the landmark list entirely. Also at this time, the command bar was upgraded to display the coordinates and names for the first few landmarks on the lists being developed for both volumes. Once landmark lists were available, it was possible to begin adding to the two remaining system level entities required for the image registration system's operation, the image registration and volume visualization subsystems.

4.3.5 Perform Image Registration. The image registration subsystem presents a common interface and set of functions to the rest of the system, yet the actual registration may be performed by more than one technique. I incorporated this encapsulation to enable users of the image registration system to experiment with a number of different types of image registration techniques. As a first attempt, however, I implemented the landmark

mapping technique introduced in Section 3.3.4. C++ specifications for these classes are in Appendix B.4.

The landmarks for each volume are used solely according to their position in their respective lists by the image registration subsystem. In other words, the landmark mapping method uses the first four landmarks of each landmark list, taken in sequence, to represent four corresponding structures in the two volumes. The two sets of four landmarks are then used by the landmark mapping algorithm to calculate the image registration transformation.

4.3.6 Render Registered Volumes. Like the image registration subsystem, the volume visualization subsystem presents a common interface and set of functions to the rest of the system, yet it is designed to permit the use of a number of surface and volume rendering tools. I incorporated this encapsulation for two reasons. First, this flexible interface will allow new techniques to be incorporated into the system more quickly and with less possible damage to other software components. Second, this will allow a choice from a collection of renderers depending on factors important to the user, such as required rendering speed, resolution, and transparency or shading effects. As a first attempt, however, I implemented an interface to Capt Parrott's surface rendering system as discussed in Section 3.4.4. C++ specifications for these classes are in Appendix B.5.

The registration transformation returned from the image registration subsystem is passed to the volume visualization subsystem along with the eyepoint location from which the user wishes to view the volumes. The eyepoint parameter for the second volume is then modified by the transformation matrix. This determines the position in object space where the eyepoint needs to be located so the second volume's rendering is correctly aligned with that of the first volume. The eyepoints for each volume are then incorporated into control files containing start-up parameters used by the selected rendering tool. After the renderer has completed its processing for each volume, it saves the resulting image to secondary storage. The user interface can then display those images in each volume's three-dimensional viewing area.

4.4 Overview of System Operations

In the previous section, I discussed the details behind the image registration system's implementation, describing the individual entities and functions used to build the system. That presentation of the system's components, however, would be incomplete without

providing some idea of the image registration process in actual practice. Therefore, in this section, I briefly outline the system's input requirements and the sequence of events taken to register two volumes of medical imaging data.

In the accompanying photographs, the medical imaging data displayed is of a CT scan of a human female thorax, stored as a series of 30 transverse slices. Each slice is composed of 240 voxels in the x–dimension and 164 voxels in the y–dimension. The dataset was provided by Vital Images, Inc., of Fairfield, IA.

4.4.1 System Input. The image registration system requires only two input sources, the two volume datasets to be registered and a control file for each volume describing the dataset's format. The control files are referenced in the command line used to initiate the program's execution. The system operates upon volume datasets organized as a series of transverse slices. Each slice occupies a separate file located on the computer's secondary storage, and is named simply with its sequence number within the series of slices composing the volume dataset. Each volume dataset is located in a separate subdirectory.

4.4.2 System Execution. Once the above input requirements are satisfied, the program is executed. The first step in the process is the construction of a volume object for each dataset being registered. Following this, the user interface constructs its two-dimensional views, three-dimensional views, and command bar. At this time, the two-dimensional views display the slices of data initially extracted from each volume, as shown in Figure 4.1.

Note the single voxel address coordinate displayed to the user prior to any interaction with the data. This value represents the coordinate of all voxels lying on the slice being displayed, and is one less than the slice number. This difference stems from the traditional practice of numbering slices starting from '1,' but the equally traditional practice of assigning the first voxel in the volume the coordinates [0, 0, 0].

In Figure 4.2 the user has begun interacting with the slice data presented in the first volume's axial two-dimensional view. An important point to note is the crosshair's transparent center, allowing the voxel being interacted with to show through. This is a useful feature when trying to identify boundaries or other abrupt changes in the slice.

The next step is for the user to determine if there are desirable landmarks present in the slice displayed by the two-dimensional view. If not, then the user has the choice of selecting the next or the last slice in the sequence or requesting a slice from a completely

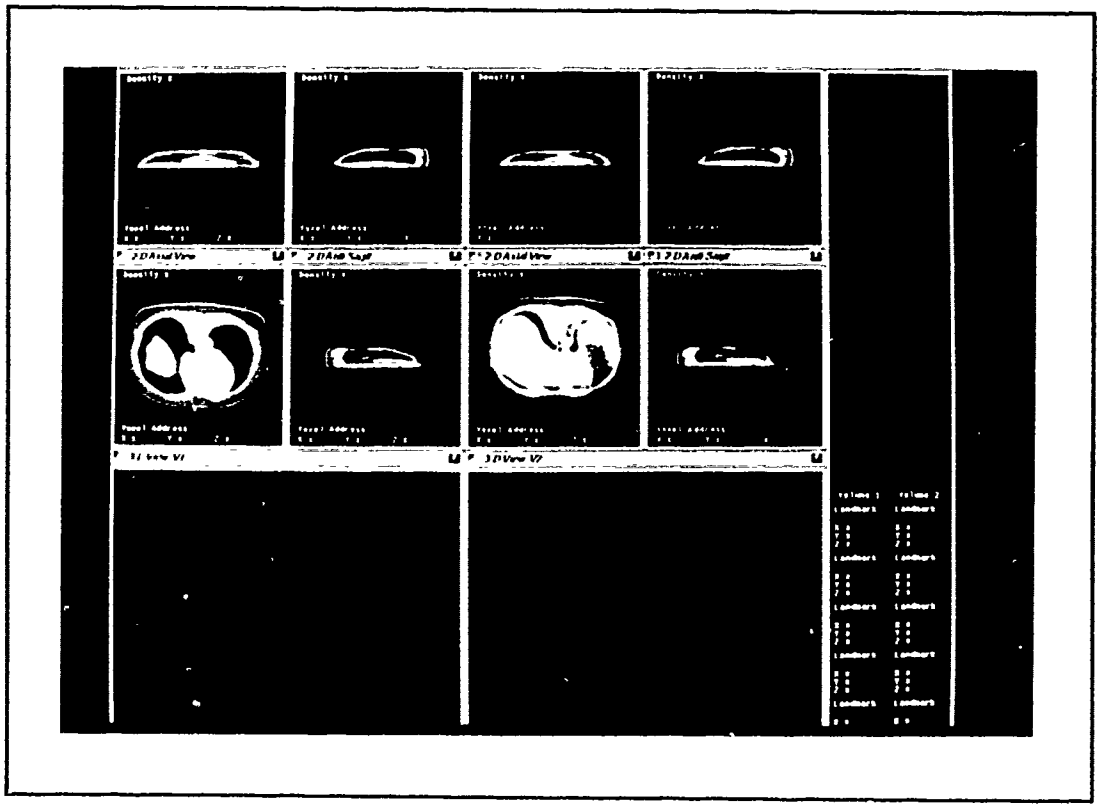

Figure 4.1. Initial User Interface Display

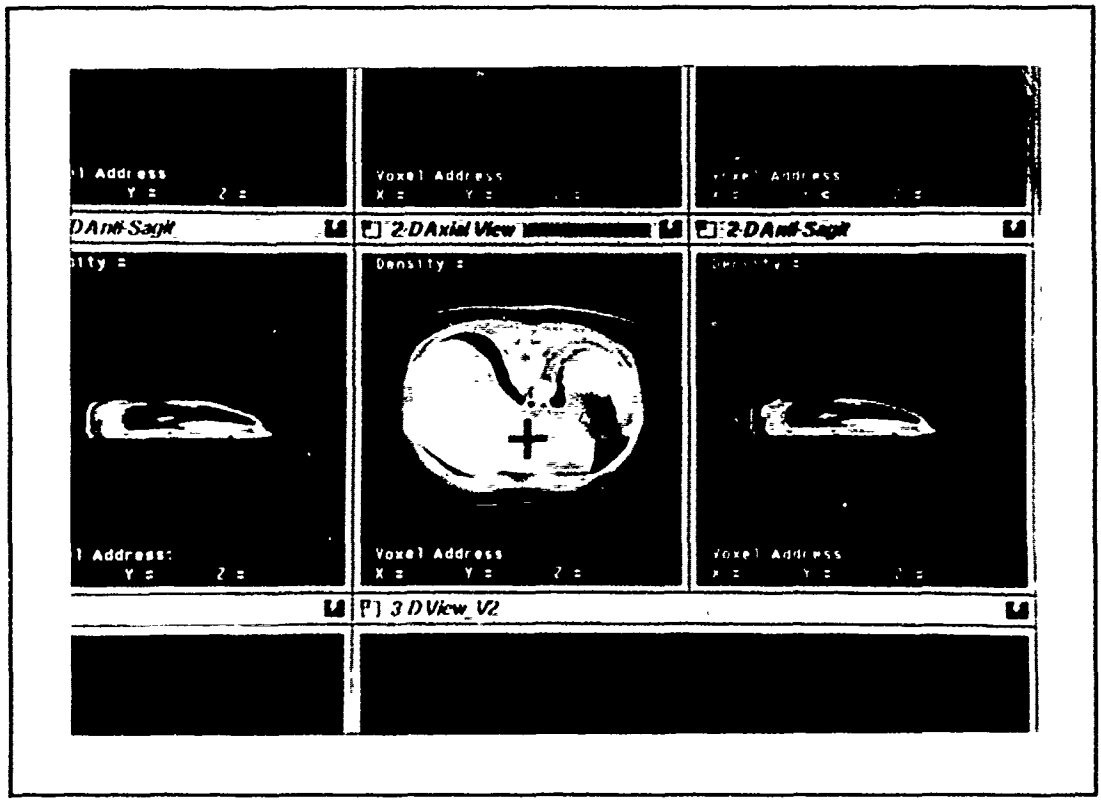

Figure 4.2. Closeup of Crosshair

different portion of the volume. Once the user has identified a potential landmark, the desired voxel may be selected, as shown in Figure 4.3. In this case, the user has selected a voxel from the next slice after the slice shown in Figure 4.2. The voxel coordinates shown tell the user that the selected voxel also lies in the coronal and right sagittal plane currently being displayed (note the corresponding X coordinate in the right sagittal view and the corresponding Y coordinate in the coronal view). The selected voxel's density value is also displayed to assist the user when trying to select the corresponding landmark in the other volume.

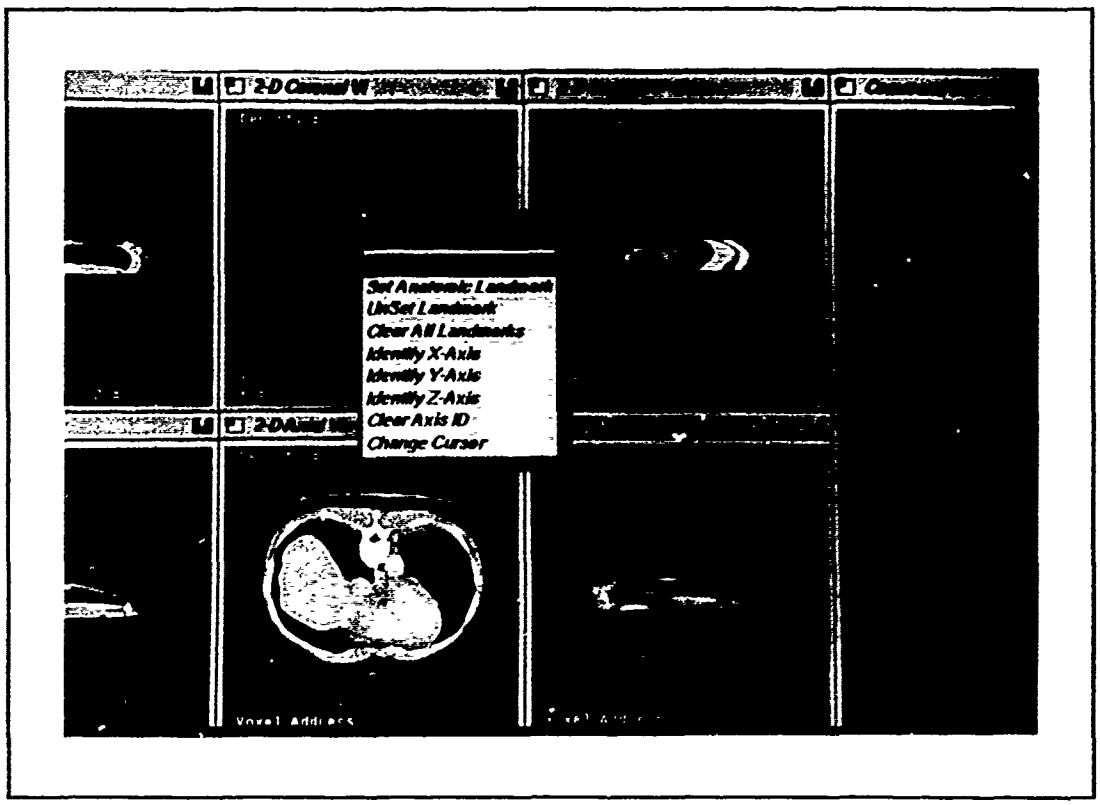

Figure 4.3. Selection of a Voxel

At this point the user has the option of designating this voxel as a landmark, as indicated in Figure 4.4, or releasing the voxel in search of another. If selected, the landmark is highlighted on the view in which it was selected, as well as the any of the other views it also happens to be visible in. The landmark is also entered into the landmark list for that volume, as shown in the command bar. It is generally most effective for the user to then identify the corresponding landmark in the second volume. By selecting landmarks

in pairs, the user can more quickly remove from one volume's list those landmarks that cannot be easily matched in the other volume.

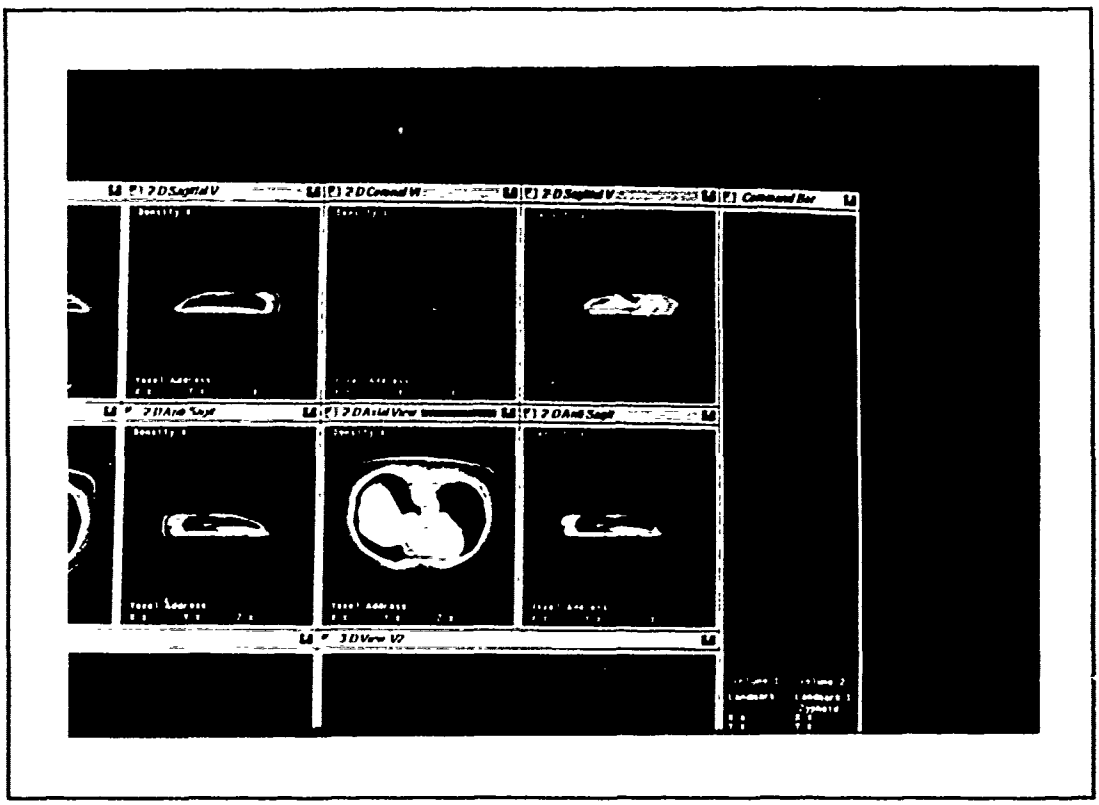

Figure 4.4. A Designated Landmark

Figure 4.5 illustrates a typical image registration session after a complete set of four landmarks has been selected for each volume. At this point, the user requests the image registration subsystem to process the two landmark lists and calculate the registration transform necessary to align the two volumes. Once the transformation has been determined, the volume visualization subsystem processes the two volumes and generates a surface or volume rendered image for each. The final step of the process displays these images in the volumes' three-dimensional viewing areas, as shown in Figure 4.6 (surface renderings shown). An example of a composite analysis based upon these registered volumes is also demonstrated in Figure 4.7.

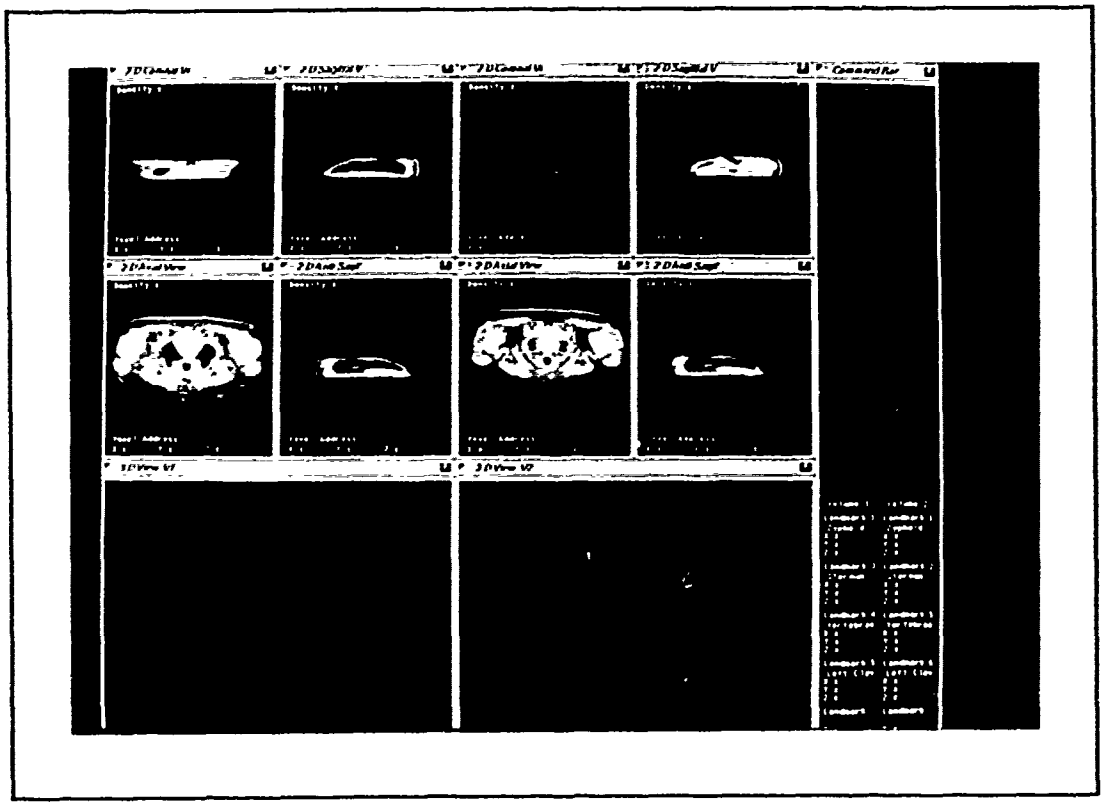

Figure 4.5. All Landmarks Selected

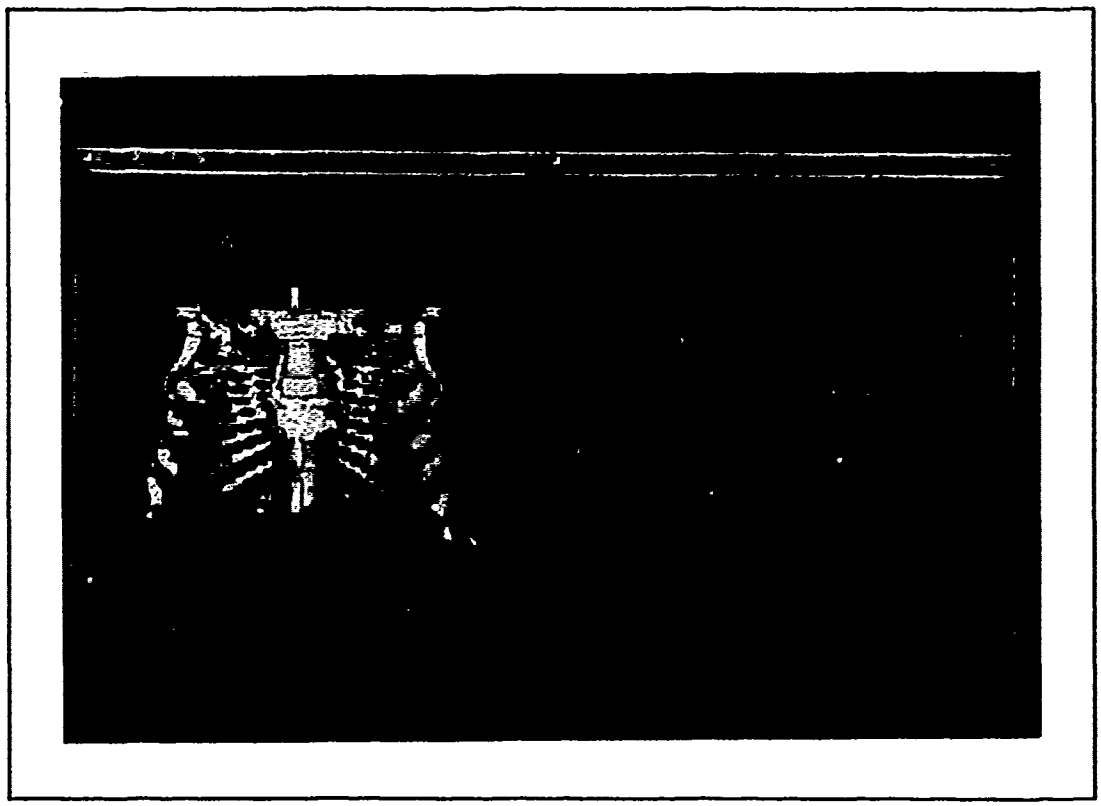

Figure 4.6. Renderings of the Registered Volumes

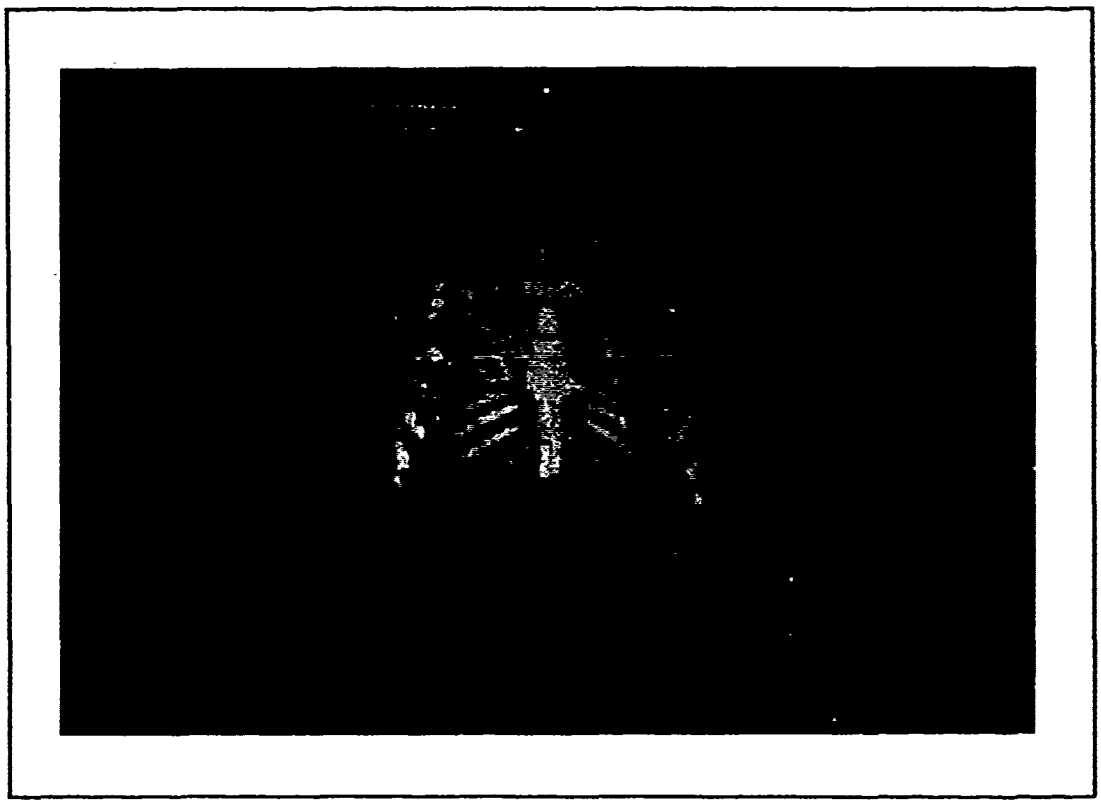

Figure 4.7. Composite Analysis of Registered Volumes

4.5 Summary

This chapter summarized the activities undertaken in order to implement the image registration system. After briefly describing the environment in which this development took place, I presented some of the highlights of the top-down incremental approach used to build the system. The system's operations were then discussed, accented with photographs demonstrating an actual image registration session. The next chapter will discuss some of the conclusions drawn from this research and outline several recommendations for additional research in this area.

V. Conclusions and Recommendations

This research effort's original requirement was to develop a system to register medical images using a structure-based correlation technique based upon simple, three-dimensional relationships among several user identified landmarks. The scope of the effort was limited to the use of a structure-based image registration method, applied to medical image volume datasets following user interaction with the data. This chapter will present some concluding remarks about the success of this system's implementation, and will recommend several potentially fruitful directions for future research in this area.

5.1 Research Conclusions

The image registration system developed as part of this effort can perform useful service in continuing research to improve the capabilities and acceptability of three-dimensional imaging in clinical situations. The following sections discuss the results of the two major components of the development effort, the image registration subsystem and the user interface.

5.1.1 Image Registration. The image registration technique implemented during this research performed successfully, and was demonstrated in this thesis using sample CT imaging data. Two different volume datasets were registered using the three-dimensional locations of a series of user identified landmarks. In a process analogous to texture mapping, a landmark mapping algorithm was employed to determine the registration transformation between the two volumes. This transformation was used to modify viewing parameters associated with the volume to be registered. The volume visualization subsystem, using the modified viewing parameters, was then able to display corresponding views of the volume datasets.

The success of the structure-based, landmark mapping technique described herein is significant for two reasons. First, it confirms the diagnostic possibilities of this form of image registration, especially when applied to images possessing complex anatomy such as found in the thorax and abdominal cavity. In these situations, the use of automatic and/or retrospective surface-based registration techniques may not be feasible for some time. Thus, structure-based techniques such as this may offer the only solution for certain cases when medical image registration is to be conducted.

Second, it demonstrates a simple and potentially more accurate method for determining the exact three-dimensional relationship between the landmarks identified in each volume. Certain alternative structure-based methods employ best-fit approximation techniques to determine the registration transformation, which may introduce additional inaccuracies into the process. However, objective conclusions regarding this image registration system's accuracy can not be made at this time. It remains for additional testing to determine the relative accuracy of these and other methods, especially in terms of detecting and characterizing any distortion effects caused by affine transformation components.

5.1.2 User Interface. The original requirements for the image registration system's user interface were completely satisfied. This development effort produced a system capable of displaying multiple, orthogonal two-dimensional medical images from volume datasets. The system incorporates a crosshair and menu driven method to allow anatomic landmark and external marker identification directly from medical imaging data displayed within a series of two-dimensional views. Following image registration, the system displays three-dimensional surface or volume renderings of the registered datasets.

The user interface's design and simplicity not only enhances its own operation, but also benefits the entire image registration process. First, the user interface appears to hold potential for increasing the accuracy of landmark selection over other structure-based methods. This system accepts landmarks identified from anywhere within the volume dataset, whereas other applications confine the user's selections to certain slices or surfaces extracted from the volume. It may be argued that the best landmarks for image registration are not always visible on any one particular slice or surface. Thus, the user interface's flexibility provides the user with a much wider assortment of completely independent possible landmarks from which to choose.

Second, the ease and reliability with which recognizable and unambiguous landmarks may be selected with this system reveals several possibilities for improving structure-based image registration methods. One such prospect is diminishing the reliance upon the expertise of an anatomist for landmark selection. The straightforward type of interface developed for this effort would allow selections from a series of standard anatomical landmarks and external markers. Such selections could, in general, be made by a skilled technician, saving the expert's time for truly unique and difficult cases. Another and related benefit would result from the expanded and routine use of external markers in medical imaging sessions. Standard marker locations would further reduce the amount of user effort required to select landmark.

On the other hand, one problem noted with this user interface design is that the coronal and sagittal views may not present an immediately recognizable display of the volume. Certain medical imaging modalities produce data with gaps (i.e., noncontiguous slices with some interslice spacing), corresponding to regions of patient space that were not scanned. Since this system currently does not estimate these missing voxels, the internal volume representation is built by stacking the noncontiguous slices one atop the other, without leaving space where data were not gathered. When looking at sagittal or coronal slices of such data, it appears as if the volume were compressed along the axial dimension, and often shows abrupt surface discontinuities. Regardless, I believe this problem is more than outweighed by the benefit of having these views, which greatly increase the user's ability to maintain their spatial perspective as they move around the volume in search of landmarks.

Another potential problem with this design is the possibility of the user being unable to positively identify one single voxel as representative of a particular anatomic landmark or external marker. For example, external markers displayed on a medical image, depending upon the imaging modality involved, may have relatively large dimensions (on the order of 5 to 10 millimeters) compared to a voxel's dimensions. It is therefore possible for a single landmark to be represented by more than one voxel. Nevertheless, I believe the system's design minimizes this problem by providing a number of different views of the volume. In the above ambiguous case, the user is generally able to decide on the same single voxel, in both groups of voxels composing the landmark in each volume, as being the most representative of the desired landmark. Also, the flexibility of being able to select landmarks from all over the volume allows the user to bypass questionable landmarks in favor of those more readily identifiable in views of both volumes.

5.1.3 Overall Implementation Results. The top-down, rapid prototyping approach taken for this development was very much a mixed blessing. On the one hand, it was extremely beneficial to have a working, useful system early in the implementation phase. The system's earlier versions, although limited to displaying slices of data from the volume dataset, allowed the user to sample the density values from voxels located anywhere within the volume. This capability was frequently used not only by myself but also by other researchers pursuing their own studies of the medical imaging datasets available to us.

On the other hand, the sequential, top-down nature of the implementation led to a rather ironic end result. The image registration software was arguably the most important

portion of software to be developed for this research, yet it wasn't implemented until very late in the process. This primarily stemmed from the structure-based image registration method's visual nature. It was necessary to identify potential landmarks available from the data with sufficient three-dimensional context so the user had confidence in what was being examined. It was equally necessary to be able to interactively "point and click" on landmarks for the system to actually use in image registration. Obviously, late development of such critical software had the expected result of severely constraining extensive test and evaluation of the image registration subsystem. In addition, it was not possible to extend that code to perform other desirable functions, such as implementing some form of error checking to identify possibly misplaced landmarks.

5.2 Recommendations

The image registration system is useful in its present form, although it may certainly be improved. Such improvements are considered in the following suggested enhancements to the current implementation. I also recommend additional research areas for image registration, in general, at the Air Force Institute of Technology.

5.2.1 Suggested Enhancements. The first improvements to the system as implemented should be performed on the image registration subsystem. This component should test landmark pairs, taken in order from the two volume's landmark lists, to determine if the user's selections actually represent the best choice. Some procedure, such as the cross-correlation technique discussed in (19), would automatically examine the voxels surrounding the one selected in the volume to be registered. In the event a better match is determined, the appropriate landmark is automatically shifted to the new voxel. Another improvement would be for the image registration subsystem to expand its use of landmarks beyond the first four on the landmark list. Tests could be performed to calculate all the possible transformations using various combinations of landmarks from the entire list. In this way, the subsystem would statistically determine the optimal transform, and possibly disqualify landmark pairs (presumably poorly matched) that skew the results beyond some threshold criterion.

To date, limited testing conducted on the image registration system has not uncovered any instances of poor or distorted alignments through the introduction of shearing or scaling components into the registration transformation. However, the restriction that all datasets to be used with the system must possess the same dimensions is too limiting for

practical use. This is especially true for registrations involving datasets from physiologic modalities (i.e., PET and SPECT), which typically offer less resolution than CT or MR. The assumption that all datasets have the same number of voxels should be eliminated, and the system modified to perform resampling of the datasets (e.g., tri-linear interpolation) when necessary, thus making its operation more flexible. Additional testing would then be necessary to evaluate any changes to the registration transformation's accuracy.

A related improvement would be the elimination of the simplifying assumption that voxels projected onto the two-dimensional views possess the same dimensions as the display screen's pixels. The user interface and image registration subsystem both require modification to account for these real-world differences. A benefit of this change is that it would immediately be useful for implementing a zoom capability within the two-dimensional viewing areas. This feature would assist the user in landmark placement by providing a magnified view of the voxels being considered for selection. Safeguards would be necessary to maintain the relationship between the display screen pixels and the corresponding voxel being sampled. Again, additional testing would then be called for to determine the impact of such changes on the resulting registration transformations.

Another area for improvement involves the operations performed to center and clip slice data to the two-dimensional view. These operations would be performed more efficiently by using homogeneous coordinate transformations and other standard three-dimensional viewing practices. In addition, scaling operations to implement the previously mentioned zoom feature would be much simpler. This will, however, complicate the software responsible for interacting with the mouse and cursor, forcing additional transformations to get from the cursor's screen space coordinates to the desired image space coordinates. Thus, improvements such as this must proceed in a systematic manner, otherwise more harm than good may be done to the existing system.

In general, the user interface performed successfully during this research. Nevertheless, there is still room for improvement, especially in terms of making the design even more user centered. The system is a first product that now needs to be demonstrated, and have actual users (e.g., radiologists and technicians) experiment with it. This experimentation will provide at least the following two benefits.

1. Gain knowledge about desired additional functionality and changes to the user interface's look and feel. In addition, gather information for other improvements, not just regarding the user interface, but for the entire image registration system. For

example, one proposal has already been suggested to provide magnification of the slice data displayed in the two-dimensional view, possibly allowing easier selection of the desired voxel.

2. Gather objective, expert assessment of the quality and utility of the the image registration system's product. This would be especially important when evaluating other techniques, and might help determine if the arbitrary rankings set forth in Sections 2.2.2.5 and 2.2.2.6 are meaningful.

A final area for improvement exists in the exploitation of those system operations that may be executed in parallel with one another. The following represent just two suggestions to address these possibilities within the image registration system:

1. At a coarse level of parallelism, the user interface's response times would be greatly increased by extracting each view's slices from the volume while allowing the user to interact with the rest of the system.
2. At a fine level of parallelism, slice and volume operations could exploit the inherently parallel nature of the large, regular arrays of voxels used by the system. The possibilities extend from file operations responsible for loading the volume dataset to operations responsible for displaying the data to the user.

Basically, the existing system demonstrates the feasibility of the user interface and the landmark mapping image registration technique. Now that the system's effectiveness has been shown, the engineering effort needs to be directed towards making the system more efficient.

5.2.2 Additional Research. Recommendations for additional image registration research fall into two categories:

1. Research should be conducted into other signal and image processing fields for ideas that may yield previously unthought of techniques that may be applied to the image registration problem. At AFIT, this might be well suited to a multidisciplinary effort in conjunction with the image processing group.
2. Research should be conducted into artificial intelligence or fuzzy reasoning applications to the image registration problem, especially in the areas of landmark identification and selection. Effort spent in this area will almost certainly result in automated

systems no longer requiring a user to interactively determine how to register various medical datasets.

My final recommendation expands upon the previous proposal to have users (possibly at the Wright-Patterson Medical Center or the Wright State University School of Medicine) evaluate the image registration system. Efforts should be taken to increase their exposure to our research, with the objective of gaining their support and participation in future efforts. One example of such support might include obtaining data for continuing image registration and other medical image processing research at AFIT. A possible long term goal might be for a trial installation of the image registration system in the user's environment, providing real world clinical feedback on the system's design and capabilities.

Appendix A. *Image Registration System Context Analysis*

A.1 Overview

In context analysis, the objective is to determine "... why the system is to be created and why certain technical, operational, and economic feasibilities are the criteria which form boundary conditions for the system" (30:6). This is frequently the starting point of the entire development effort, when there may appear to be innumerable possibilities as to how the problem should be addressed, yet no clear indicator as to which direction to pursue first. By placing the problem to be solved in context with all other elements of its environment, the major interfaces with that environment are made apparent. Thus, the problem's scope is determined, the system's overall purpose is better understood, and the development can be focused on meeting those external requirements.

The context analysis's end result, the "needs product," documents the relationship between the system and its environment. It also specifies, at the topmost level, the system requirements to be satisfied. The needs product is therefore composed of:

1. a problem statement, addressing *why* the system is to be built;
2. a context diagram, describing the external interfaces between the system and its environment;
3. an event list, listing the stimuli from the environment to which the system must respond; and
4. a narrative constraint list, identifying those limitations to be imposed upon the system and its development.

A.2 Original Context Diagram

The context diagram shown at Figure A.1 provides some perspective as to how the system evolved over the course of this development's analysis and design phases. Compare this original context diagram with that of the final needs product, presented in the next section.

A.3 Final Context Analysis

A.3.1 Problem Statement. The AFIT Medical Image Processing System (AMIPS) image registration system is responsible for aligning three-dimensional volume datasets

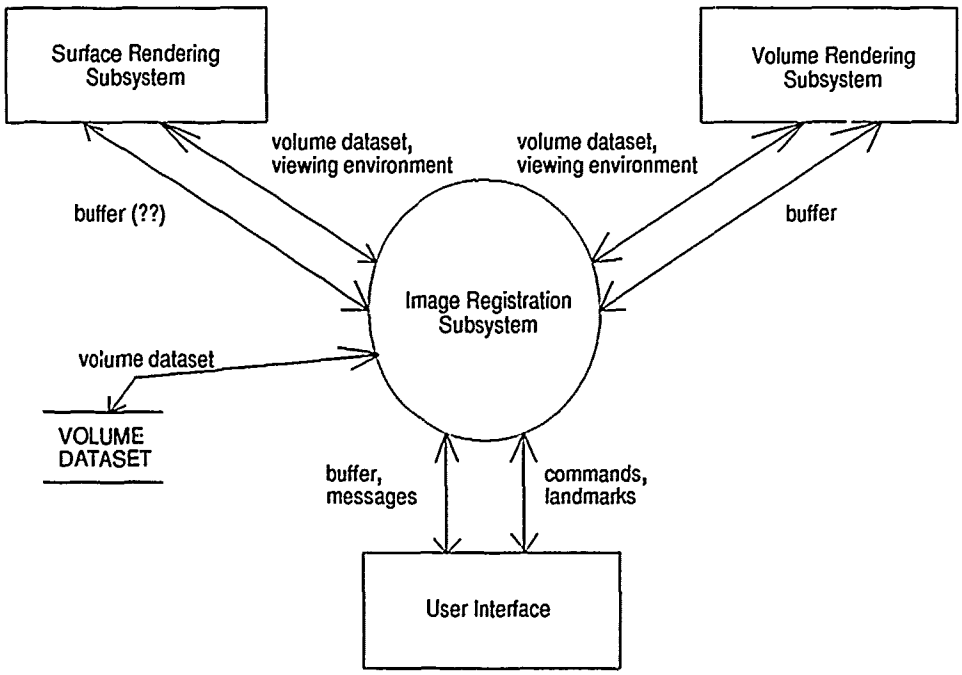

Figure A.1. Original Image Registration System Context Diagram

acquired from different medical imaging modalities. The registration process will be based upon the selection of certain anatomical landmarks (and possibly of external body surface markers placed prior to the imaging session) that are readily discernable by the particular imaging modality. The registration process seeks to correlate corresponding landmarks and markers between the volume datasets being registered. The datasets must be aligned with sufficient accuracy to allow certain analyses of the registered 3-D volumes. A composite analysis should display an image composed of information obtained from registered volumes obtained using more than one imaging modality. A comparative analysis should illustrate the changes having taken place between two or more medical images obtained using the same modality.

A.3.2 Event List . Based upon external stimuli. A 'T' following an event denotes a temporal event, an 'F' denotes a flow-oriented event, and a 'C' denotes a control event.

- The system user, via the AMIPS, requests certain previously acquired volume datasets be registered. (F)
- The system user, via the AMIPS, requests the desired form of analysis to be performed on the registered datasets. (F)
- The system user identifies anatomical landmarks or external markers, displayed on intermediate images from each dataset, to be correlated between those datasets during the registration process. (F)

A.3.3 Narrative Constraint List.

1. Technical
    - As much as possible, the image registration portion of the AMIPS should be executable on both the SGI and the Sun systems located in the Graphics Lab.
    - An object-oriented analysis (OOA) and object-oriented design (OOD) will be performed to develop this system, which is to be implemented using the C++ programming language.
2. Logistical
    - Raw MRI (and possibly CT) data, appropriately sanitized to assure patient anonymity, may be obtained from the Wright-Patterson Medical Center Radiology Department.

- Medical imaging data will also be obtained from public sources (University of North Carolina – Chapel Hill).
- Intermediate images displayed to the user for anatomic landmark and external marker identification will be slice data extracted directly from the medical image datasets. The final images of the registered volumes will be generated by surface rendering software (marching/dividing cube, kriging) or by volume rendering software (distributed ray caster) being developed as separate efforts for the AMIPS.

A.3.4 Context Diagram. The context diagram for the image registration system is shown in Figure A.2.

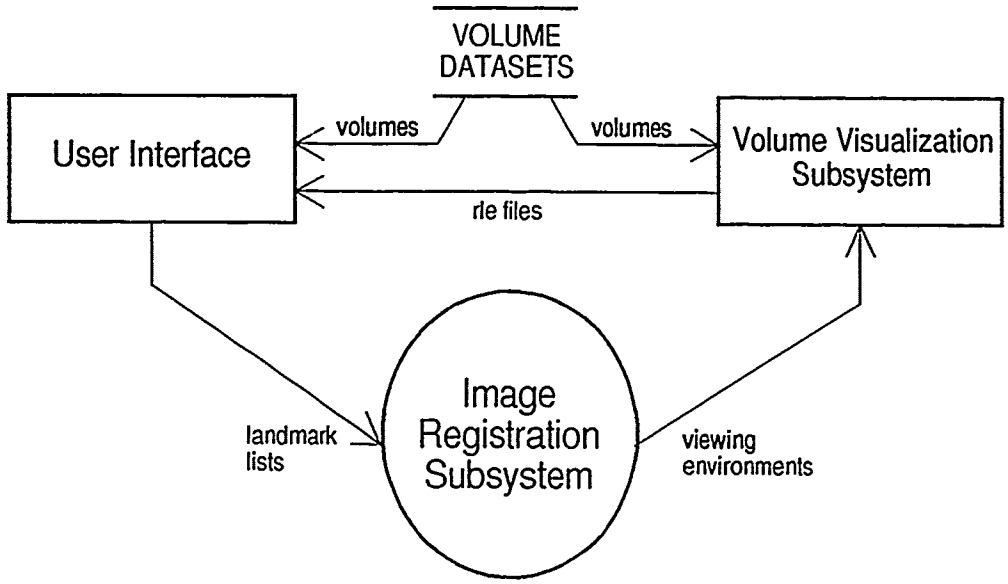

Figure A.2. Image Registration System Context Diagram

A.3.5 Textual System Description. Supplemental textual information regarding the proposed system's operations is provided.

1. For each of the two volume's to be registered, three (or maybe four) 2-D views, each orthogonal to one of the three major axes (and possibly one more showing the opposite sagittal view), would be opened within the user interface's window. Within each 2-D viewing area display actual slices of the medical image dataset as intermediate views of the volumes to be registered.

2. Within any 2-D view a cursor may be moved around using a mouse, which also controls crosshairs associated with the cursor. As the cursor is moved to a point in one of the images corresponding to a voxel in the original volume, all three crosshairs overlay the same voxel in each of their respective views. When a mouse button is clicked, the crosshairs should be frozen over the voxel, thus allowing the user to select desired anatomical landmarks or external markers (crosshairs may be unfrozen or deleted by clicking another mouse button). This process would be repeated until the user has identified the same anatomical landmarks or external markers in both volumes.

3. The user would then request (via the Command Bar) that the two volumes be registered (one volume reoriented so that its voxels identified in the previous step are aligned with the corresponding voxels from the other volume). The image registration subsystem should calculate the transformation matrix (combination of rotation, translation and scale) required to align the corresponding landmarks or markers.

4. Once the two volumes had been registered, the registration transformation matrix would be used to modify the viewing environment associated with one of the two volumes.

5. Each volume, along with its associated viewing environment, should then be passed to a surface rendering or volume rendering tool. The rendering tool will return an image of the volume according to the requested viewing parameters.

6. Each image will then be displayed side-by-side as 3-D views within the user interface. If the user is satisfied that the two volumes are displayed with the proper orientation and appear registered, then the user may declare, via the command bar, whether to next perform a composite or comparative analysis on the volumes.

7. Depending on whether the user requested a composite or a comparative analysis of the two volumes, the two volumes' images will be appropriately manipulated, yielding a single image containing the results of the desired analysis.

8. The resultant image will then be displayed as a single 3-D view within the user interface.

Appendix B. *Class Specifications*

This appendix lists the C++ classs specifications for each of the major components making up the image registration system. These components represent the entities identified by the context and problem analyses that preceded this design.

B.1 User Interface Classes

The user interface is composed of four two-dimensional views and one three-dimensional view associated with each volume being registered. A command bar allows the user to interact with the overall image registration system.

B.1.1 User Interface.

```
#ifndef _USER_INTERFACE_H
#define _USER_INTERFACE_H

/*****************************************************************************
* CLASS NAME:  User_Interface
* DESCRIPTION:  The User Interface class is responsible for building and
* managing all the components necessary to view and interact with
* the volumes used during Image Registration.  This particular
* version of the User Interface is tailored for use on the SGI 4D
* workstations, employing functions from the SGI Graphics Library (gl.h)
* to construct the window, generate various views of the data, and
* control the display cursor in response to mouse commands.

* SUPER CLASS:  Object
* PUBLIC METHODS:
* identify ();
* build2D_Views ();
* build3D_View ();
* buildCommandBar ();
* manageUserInterface ();
* destroyUserInterface ();
* OPERATORS:  None.
* DATE WRITTEN:  27 July 91 by Pat Rizzuto
* VERSION:  1.0
* LAST MODIFIED:
* HISTORY:
*
*****************************************************************************/

#include "amips.h"
#include "Volume.h"
#include "View_2D.h"
```

```
#include "View_3D.h"
#include "Command_Bar.h"
#include "Landmark_List.h"
#include "Image_Registration_Subsystem.h"
#include "Volume_Visualization_Subsystem.h"

#define COR1    0       // Window 0      ************************************
#define SAG1    1       // Window 1      **                                **
#define AXI1    2       // Window 2      **  Maintain same order as used   **
#define ANT1    3       // Window 3      **  in User_Interface.cc for      **
#define COR2    4       // Window 4  **  Window_Coordinates and     **
#define SAG2    5       // Window 5      **  Window_Titles     **
#define AXI2    6       // Window 6      **                                **
#define ANT2    7       // Window 7      ************************************
#define VOL1    8       // Window 8
#define VOL2    9       // Window 9
#define CBAR    10      // Window 10

#define MAXWIN  11      // Maximum number of windows to be displayed

class User_Interface
{
Volume *V1, *V2;
View_2D *V1_Coronal, *V1_Sagittal, *V1_Axial, *V1_AntiSagittal,
*V2_Coronal, *V2_Sagittal, *V2_Axial, *V2_AntiSagittal;
View_3D *V1_3D, *V2_3D;
Command_Bar *CB;
Landmark_ListPtr V1_LandmarkList, V2_LandmarkList;
Volume_Visualization_Subsystem *Volume_Visualizer;
Image_Registration_Subsystem *Image_Registerer;
Matrix Registration_Transformation;
boolean V1_2D_Views_Displayed, V2_2D_Views_Displayed,
V1_3D_View_Displayed,  V2_3D_View_Displayed,
CommandBar_Displayed,
Registration_Enabled;
        long Window_Coordinates [MAXWIN][2];
        String  Window_Titles [MAXWIN];

boolean Editing_Should_Be_Enabled ()
{
   if ( V1_2D_Views_Displayed AND NOT V2_2D_Views_Displayed AND
V1_3D_View_Displayed  AND NOT V2_3D_View_Displayed  AND
CommandBar_Displayed )

      return  TRUE;
   else
      return  FALSE;
};

boolean Registration_Is_Enabled ()
{
```

```
   if ( V1_2D_Views_Displayed AND V2_2D_Views_Displayed AND
V1_3D_View_Displayed  AND V2_3D_View_Displayed  AND
CommandBar_Displayed )
      Registration_Enabled = TRUE;
   else
      Registration_Enabled = FALSE;

   return Registration_Enabled;
};

void Initialize_Transform ()
{
   for (register int row = 0; row < 4; row++)
      for (register int col = 0; col < 4; col++)
         if ( row EQUALS col )
            Registration_Transformation [row][col] = 1.0;
         else
            Registration_Transformation [row][col] = 0.0;
};

void buildColorMap (Volume *thisVolume);
void displayMessage ();
void Register_Volumes ();
void Edit_Volume ();

public:

        User_Interface (Image_Registration_Subsystem *img_reg,
Volume_Visualization_Subsystem *vol_viz_sys);
        ~User_Interface ()
        {
   if ( V1_Coronal NOT_EQUALS NULL )
      delete V1_Coronal;
   if ( V1_Sagittal NOT_EQUALS NULL )
      delete V1_Sagittal;
   if ( V1_Axial NOT_EQUALS NULL )
      delete V1_Axial;
   if ( V1_AntiSagittal NOT_EQUALS NULL )
      delete V1_AntiSagittal;

   if ( V2_Coronal NOT_EQUALS NULL )
      delete V2_Coronal;
   if ( V2_Sagittal NOT_EQUALS NULL )
      delete V2_Sagittal;
   if ( V2_Axial NOT_EQUALS NULL )
      delete V2_Axial;
   if ( V2_AntiSagittal NOT_EQUALS NULL )
      delete V2_AntiSagittal;

   if ( V1_3D NOT_EQUALS NULL )
      delete V1_3D;
```

```
   if ( V2_3D NOT_EQUALS NULL )
      delete V2_3D;

   if ( CB NOT_EQUALS NULL )
      delete CB;

   if ( V1_LandmarkList NOT_EQUALS NULL )
      delete V1_LandmarkList;
   if ( V2_LandmarkList NOT_EQUALS NULL )
      delete V2_LandmarkList;
        };

void identify ();
void build2D_Views (Volume *thisVolume);
void build3D_View (Volume *thisVolume);
void buildCommandBar ();
void manageUserInterface ();
void destroyUserInterface ();
};

#endif
```

B.1.2 Two-Dimensional View.

```
#ifndef _VIEW2D_H_
#define _VIEW2D_H_

/*****************************************************************************
* CLASS NAME:  View_2D
* DESCRIPTION:  A 2-D View class,
* SUPER CLASS:  Window
* PUBLIC INSTANCE VARIABLES:  None.
* PUBLIC METHODS:
* identify ();
* setSlice_Num ();
* getSlice_Num ();
* NextSlice ();
* LastSlice ();
* Open_Window ();
* Close_Window ();
* Redraw_Window ();
* OPERATORS:  None.
* DATE WRITTEN:  5 August 91 by Pat Rizzuto
* VERSION:  1.0
* LAST MODIFIED:
* HISTORY:
*
*****************************************************************************/

#include "amips.h"
#include "Slice.h"
#include "Volume.h"
#include "Menu.h"
#include "Crosshair.h"
#include "Landmark_List.h"
#include "Edit_Tool.h"
#include "UI_window.h"

// Defines to set-up the Text_Window

#define USIZE_2D 256
#define VSIZE_2D 256

#define BACKGROUND_COLOR_2D BLACK
#define DISPLAY_COLOR_2D WHITE
#define VOXEL_DENSITY_COLOR_2D CYAN
#define VOXEL_ADDR_COLOR_2D GREEN

// Defines to set-up the Voxel Density display area

#define DISPLAY_DENSITY_UCOORD 10
#define DISPLAY_DENSITY_VCOORD VSIZE_2D - 10
```

```
#define VOXEL_DENSITY_UCOORD DISPLAY_DENSITY_UCOORD + 90
#define VOXEL_DENSITY_VCOORD DISPLAY_DENSITY_VCOORD

// Defines to set-up the Voxel Coordinates display area

#define DISPLAY_MESSAGE_UCOORD 10
#define DISPLAY_MESSAGE_VCOORD 20

#define DISPLAY_TEMPLaTE_UCOORD 10
#define DISPLAY_TEMPLATE_VCOORD 5

#define VOXEL_XVALUE_UCOORD DISPLAY_TEMPLATE_UCOORD + 30
#define VOXEL_XVALUE_VCOORD DISPLAY_TEMPLATE_VCOORD
#define VOXEL_YVALUE_UCOORD DISPLAY_TEMPLATE_UCOORD + 105
#define VOXEL_YVALUE_VCOORD DISPLAY_TEMPLATE_VCOORD
#define VOXEL_ZVALUE_UCOORD DISPLAY_TEMPLATE_UCOORD + 173
#define VOXEL_ZVALUE_VCOORD DISPLAY_TEMPLATE_VCOORD

// Defines to set-up the Landmark marker/silouette masks

/* LANDMARK_PIXEL is a place-holder in the mask.  When the Landmark is
    actually drawn to the View, the appropriate Landmark color (EXTERNAL_,
    ANATOMIC_, HIGHLIT_, etc.) will be substituted at each LANDMARK_PIXEL
    location.  Also at the time the Landmark is drawn, those pixel locations
    listed as a TRANSPARENT_PIXEL will simply not be updated, and those
    listed as SILOUETTE_PIXELs will be drawn with the SILOUETTE_COLOR. */

#define EXTERNAL_LANDMARK_COLOR GREEN
#define ANATOMIC_LANDMARK_COLOR YELLOW
#define REFERENCE_LANDMARK_COLOR RED
#define HIGHLIT_LANDMARK_COLOR MAGENTA
#define SILOUETTE_COLOR BLACK
#define TRANSPARENT_PIXEL 'T' // = transparent
#define LANDMARK_PIXEL '+' // = colored
#define SILOUETTE_PIXEL ' ' // = BLACK silouette

#define MASK_X_DIM 11 // Must be odd
#define MASK_Y_DIM 11 // Must be odd

// Defines to set-up the Menu associated with this Text_Window

#define NUM_VIEW2D_MENU_ITEMS 4

#define INTERACTWITHVOXEL 1 // See also corresponding menu text
#define GETSLICE 2 //  in View_2D::Build_Menu ().
#define EDITSLICE 3
#define SAVE2DVIEW 4

// Defines to display only certain Voxels

// #define THRESHOLD_VALUE 57
```

```
#define THRESHOLD_VALUE 7 // Redraw_Window will light only the
#define LOWER_VALUE 130 //  pixels corresponding to Voxels
#define UPPER_VALUE 150 //  with a Density value greater
//  than the THRESHOLD_VALUE, or
//  falling between the LOWER_VALUE
//  and the UPPER_VALUE (inclusive).

class View_2D : public UI_Window
{
Volume *View_Volume;
Landmark_ListPtr View_Landmark_List;
Orientationtype View_Orientation;
Crosshair *View_Crosshair;
Density Voxel_Density;
SliceIndex Voxel_Address [3], U_dim, V_dim, W_dim,
Voxel_U, Voxel_V,
Slice_Num, // Not necessarily an Axial slice.
StartRow,    EndRow, // Beginning and ending Raster
StartColumn, EndColumn; //  indices along other 2 axes.
long Relative_Pixel_U, Relative_Pixel_V;
int Uoffset,     Voffset; // Centers slice in Window
Slice *ThisSlice, *LastSlice, *NextSlice;
Queued_Input *Forward_One_Slice, *Back_One_Slice,
*Mouse_X, *Mouse_Y;
Text_Item *Voxel_Text [3], *Density_Text;
        String          Marker_Mask [MASK_Y_DIM];
Edit_Tool *Slice_Editor;
boolean Editing_Enabled;

/* This method ensures only valid values are assigned to Slice_Num, thus
    providing a 'safe' class attribute.  I therefore no longer have to
    always check whether Slice_Num is in the proper range. */

boolean Set_Slice_Num (int anInteger)
{
   if ( (anInteger > 0) AND (anInteger <= W_dim) )
   {
      Slice_Num = (SliceIndex) anInteger;
      return TRUE;
   }
   else
   {
      cerr << "View_2D::Set_Slice_Num => Attempt to set Slice_Num \n"
   << "  outside acceptable range, Slice_Num unchanged.\n";
      cout.flush ();
      return FALSE;
   }
};


/* This method 'safely  increments Slice_Num, ensuring it doesn't violate
```

```
    its range constraints for this View. */

void Increment_Slice_Num ()
{
   if ( Slice_Num < W_dim )
      Slice_Num++;
   else
   {
      cerr << "View_2D::Increment_Slice_Num => Attempt to increase \n"
   << "  Slice_Num above acceptable range, "
   << "Slice_Num unchanged.\n";
      cout.flush ();
   }
};


/* This method 'safely' decrements Slice_Num, ensuring it doesn't violate
    its range constraints for this View. */

void Decrement_Slice_Num ()
{
   if ( Slice_Num > 1 )
      Slice_Num--;
   else
   {
      cerr << "View_2D::Increment_Slice_Num => Attempt to decrease \n"
   << "  Slice_Num below acceptable range, "
   << "Slice_Num unchanged.\n";
      cout.flush ();
   }
};


void Next_Slice ();
void Last_Slice ();
void Get_Slice ();
void Update_Voxel_Address ();
void Update_Text_Display ();
        void Display_All_Landmarks ();
        void Display_Landmark_Here (short pixel_u, short pixel_v,
    Colorindex landmark_color);
        void Display_Landmark_Identifier_Around_Here (short pixel_u,
      short pixel_v);
        void Highlight_This_Landmark ();
virtual void Build_Menu ();

public:

View_2D (Volume* = NULL,              // theVolume
 Landmark_ListPtr = NULL,     // landmark_list
 Orientationtype = AXIAL,     // orientation
```

```
                long = -1, long = -1,        // def_loc_x, def_loc_y
 String = "");       // win_title
~View_2D ();

void identify ();

/* This method should only be called by the User_Interface to alert this
    View that it may allow editing of slices.  This should only happen
    for Volume 1's AXIAL View; however, under the Object-Oriented paradigm,
    this View is unaware of the User_Interface, so there will be no
    that these conditions are met on this end. */

void Allow_Slice_Editing ()
{
   Editing_Enabled = TRUE;
};

virtual void Open_Window ();
virtual void Redraw_Window ();
virtual void InteractWithMouse ();
};

#endif
```

B.1.3 Three-Dimensional View.

```
#ifndef _VIEW3D_H_
#define _VIEW3D_H_

/*****************************************************************************
* CLASS NAME:  View_3D
* DESCRIPTION:  A 3-D View class,
* SUPER CLASS:  Window
* PUBLIC INSTANCE VARIABLES:  None.
* PUBLIC METHODS:
* identify ();
* Open_Window ();
* Close_Window ();
* Redraw_Window ();
* InteractWithMouse ();
* OPERATORS:  None.
* DATE WRITTEN:  12 August 91 by Pat Rizzuto
* VERSION:  1.0
* LAST MODIFIED:
* HISTORY:
*
*****************************************************************************/

#include "amips.h"
#include "Volume.h"
#include "UI_window.h"

// #define USIZE_3D 512 // for Windows with NO borders
// #define VSIZE_3D 512 // for Windows with NO borders
#define USIZE_3D 527 // for Windows with borders
#define VSIZE_3D 396 // for Windows with borders

#define BACKGROUND_COLOR_3D BLACK

class View_3D : public UI_Window
{
Volume *View_Volume;

public:

View_3D (Volume* = NULL,              // theVolume
                long = -1, long = -1,        // def_loc_x, def_loc_y
 String = "");       // win_title
~View_3D ()
{
};

void identify ();

// virtual void Open_Window ();
```

```
// virtual void Close_Window ();
virtual void Redraw_Window ();
virtual void InteractWithMouse ();
};

#endif
```

B.1.4 Command Bar.

```
#ifndef _COMMAND_BAR_H_
#define _COMMAND_BAR_H_

/*****************************************************************************
* CLASS NAME:  Command_Bar
* DESCRIPTION:  A Command Bar class,
* SUPER CLASS:  Window
* PUBLIC INSTANCE VARIABLES:  None.
* PUBLIC METHODS:
* identify ();
* SetupFirstLandmarkListDisplay ();
* SetupSecondLandmarkListDisplay ();
* Open_Window ();
* Close_Window ();
* Redraw_Window ();
* OPERATORS:  None.
* DATE WRITTEN:  12 August 91 by Pat Rizzuto
* VERSION:  1.0
* LAST MODIFIED:
* HISTORY:
*
*****************************************************************************/

#include "amips.h"
#include "Landmark_List.h"
#include "UI_window.h"

//      Defines to set-up the Text_Window

// #define USIZE_CB 239 // for Windows with NO borders
// #define VSIZE_CB        1007 // for Windows with NO borders
#define USIZE_CB 181 // for Windows with borders
#define VSIZE_CB 986 // for Windows with borders

#define BACKGROUND_COLOR_CB BLUE
#define DISPLAY_COLOR_CB        WHITE
#define LANDMARK_ADDR_COLOR_CB CYAN
#define TEXT_COLOR_CB YELLOW
#define TITLE_COLOR_CB MAGENTA
#define LINES_COLOR_CB MAGENTA

//      Defines to set-up the Main Title display area

#define TITLE_FONT_TYPE_CB 44 // Times - Bold
#define TITLE_LINE1_FONT_SIZE_CB 30
#define TITLE_FONT_SIZE_CB 15

#define TITLE_LINE_1_UCOORD 35
#define TITLE_LINE_1_VCOORD    950
```

```
#define TITLE_LINE_2_UCOORD 27
#define TITLE_LINE_2_VCOORD    933
#define TITLE_LINE_3_UCOORD 15
#define TITLE_LINE_3_VCOORD    916

//      Defines to set-up the Landmark display area

#define NUM_LANDMARKS_DISPLAYED  5

#define LANDMARK_DISPLAY_FONT_SIZE 5

#define LANDMARK_TITLE_UCOORD 10
#define LANDMARK_TITLE_VCOORD  360

#define V1_LIST_START_UCOORD  5
#define V2_LIST_START_UCOORD USIZE_CB / 2 + V1_LIST_START_UCOORD
#define LIST_START_VCOORD  5
#define DELTA_LANDMARK_V 20

#define OFFSET_ID_U USIZE_CB / 2 - 20
#define DELTA_ID_V 15
#define OFFSET_NAME_U  5
#define DELTA_NAME_V 12
#define OFFSET_ADDRESS_U 30
#define DELTA_ADDRESS_V 12

#define SOLID_LINE_STYLE 0 // Default predefined style

#define LANDMARK_V_LINE_X0 USIZE_CB / 2
#define LANDMARK_V_LINE_Y0 0
#define LANDMARK_V_LINE_X1 USIZE_CB / 2
#define LANDMARK_V_LINE_Y1 LANDMARK_TITLE_VCOORD + 15
#define LANDMARK_H_LINE_1_X0 0
#define LANDMARK_H_LINE_1_Y0 LANDMARK_TITLE_VCOORD - 5
#define LANDMARK_H_LINE_1_X1 USIZE_CB
#define LANDMARK_H_LINE_1_Y1 LANDMARK_TITLE_VCOORD - 5
#define LANDMARK_H_LINE_2_X0 0
#define LANDMARK_H_LINE_2_Y0 LANDMARK_TITLE_VCOORD + 12
#define LANDMARK_H_LINE_2_X1 USIZE_CB
#define LANDMARK_H_LINE_2_Y1 LANDMARK_TITLE_VCOORD + 12
#define LANDMARK_H_LINE_3_X0 0
#define LANDMARK_H_LINE_3_Y0 LANDMARK_TITLE_VCOORD + 15
#define LANDMARK_H_LINE_3_X1 USIZE_CB
#define LANDMARK_H_LINE_3_Y1 LANDMARK_TITLE_VCOORD + 15

//      Defines to set-up the Menu associated with this Text_Window

#define NUM_CBAR_MENU_ITEMS     4

#define REGISTER                1       // See also corresponding menu text
#define MONITOR                 2       //  in Command_Bar::Build_Menu ().
```

```
#define SAVECBAR                3
#define QUITCBAR                4


class Command_Bar : public UI_Window
{
        Landmark_ListPtr V1_LandmarkList, V2_LandmarkList;
boolean V1_LandmarkList_Available,
V2_LandmarkList_Available,
Registration_Requested;
Font *Main_Title_Font;
Text_Item *V1_LandmarkList_IDs [NUM_LANDMARKS_DISPLAYED],
*V2_LandmarkList_IDs [NUM_LANDMARKS_DISPLAYED],
*V1_LandmarkList_Names [NUM_LANDMARKS_DISPLAYED],
*V2_LandmarkList_Names [NUM_LANDMARKS_DISPLAYED],
*V1_LandmarkList_Addr [NUM_LANDMARKS_DISPLAYED][3],
*V2_LandmarkList_Addr [NUM_LANDMARKS_DISPLAYED][3];

        virtual void Build_Menu ();

public:

Command_Bar (long = -1, long = -1,      // def_loc_x, def_loc_y
     String = ""); // win_title
~Command_Bar ();

void identify ();
boolean Ready_To_Register () { return Registration_Requested; }

void TrackFirstLandmarkList (Landmark_ListPtr first_list)
{
   V1_LandmarkList = first_list;
   V1_LandmarkList_Available = TRUE;
};

void TrackSecondLandmarkList (Landmark_ListPtr second_list)
{
   V2_LandmarkList = second_list;
   V2_LandmarkList_Available = TRUE;
};

// virtual void Open_Window ();
// virtual void Close_Window ();
virtual void Redraw_Window ();
        virtual void InteractWithMouse ();
};

#endif
```

B.2 Data Structure Classes

Three separate data structures are used in the image registration system. A voxel is used to represent an element of volume from patient space, and stores a single density value. A slice is composed of a two dimensional array of voxels, and a volume is composed of a three-dimensional array.

B.2.1 Voxel.

```
#ifndef _VOXEL_H_
#define _VOXEL_H_

#include "amips.h"

/*****************************************************************************
* CLASS NAME:  Voxel
* DESCRIPTION:  A single voxel class, represented by a gray scale value from
* 0 to 255, referred to as the voxel's Density, regardless of the
*       particular imaging modality (MR, CT, etc.) from which it was obtained.
* SUPER CLASS:  Object
* PUBLIC INSTANCE VARIABLES:  None.
* PUBLIC METHODS:
* identify ();
*       getDensity (Density);
*       setDensity (Density);
* OPERATORS:  None.
* DATE WRITTEN:  23 July 91 by Patty Brightbill
* VERSION:  1.0
* LAST MODIFIED:
* HISTORY:
*
*****************************************************************************/

class Voxel
{
Density grayLvl;
float   color;
float   opacity;

public:

void identify ();

void getDensity (Density *value);
void setDensity (Density value);
void getColor (float *c);
void setColor (float c);
void getOpacity (float *o);
void setOpacity (float o);
```

```
};

#endif
```

B.2.2 Slice.

```
#ifndef _SLICE_H
#define _SLICE_H

/*****************************************************************************
* CLASS NAME:  Slice
* DESCRIPTION:  The Slice class simply serves as an intermediate storage
* data structure for certain orthogonal views of a Volume along the
* major axes.
* SUPER CLASS:  Volume
* PUBLIC INSTANCE VARIABLES:  None.
* PUBLIC METHODS:
* identify ();
* getVoxel (unsigned int, unsigned int, Density);
* setVoxel (unsigned int, unsigned int, Density);
* OPERATORS:  None.
* DATE WRITTEN:  26 July 91 by Pat Rizzuto
* VERSION:  1.0
* LAST MODIFIED:
* HISTORY:
*
*****************************************************************************/

#include <stream.h>
#include <time.h>
#include "amips.h"
#include "Voxel.h"

#define DEFAULT_USIZE 256
#define DEFAULT_VSIZE 256

class Slice
{
        Voxel *slice;
        unsigned int slicesizeU, slicesizeV;
Orientationtype Orientation;

        unsigned int voxelAddress (unsigned int u, unsigned int v)
        {
           if ( (u < slicesizeU) AND (v < slicesizeV) )
              return ( (slicesizeU * v) + u );
           else
           {
              cerr << "Slice::voxelAddress:  Illegal Slice coordinates:\n";
              cerr << form ("  u = %d and v = %d not in legal range.\n", u, v);
              return (0);
           }
        };

public:
```

```
Slice (unsigned int u, unsigned int v)
{
           slicesizeU = u;
           slicesizeV = v;
           slice = new Voxel [slicesizeU * slicesizeV];
};

        ~Slice ()
        {
           if (slice NOT_EQUALS NULL)
              delete slice;
              // delete [slicesizeU * slicesizeV] slice;
        };

        void getSize (unsigned int &u, unsigned int &v)
{
   u = slicesizeU;
   v = slicesizeV;
};

void identify ();

        void getVoxel (unsigned int u, unsigned int v, Density *value);
        void setVoxel (unsigned int u, unsigned int v, Density value);

};

#endif
```

B.2.3 Volume.

```
#ifndef _VOLUME_H
#define _VOLUME_H

/*****************************************************************************
* CLASS NAME:  Volume
* DESCRIPTION:
* SUPER CLASS:
* PUBLIC INSTANCE VARIABLES:  None.
* PUBLIC METHODS:
* identify ();
* getSize (unsigned int, unsigned int, unsigned int);
* getVoxelDimensions (float, float, float);
* getInterSlice (float);
* getVoxel (unsigned int, unsigned int, unsigned int, Density);
* getSlice (Orie-tationtype, int);
* setVoxelDimensions (float, float, float);
* setInterSlice (float);
* setVoxel (unsigned int, unsigned int, unsigned int, Density);
*       render (Attributes, Environment);
* OPERATORS:  None.
* DATE WRITTEN:  23 July 91 by Patty Brightbill
* VERSION: 1.0
* LAST MODIFIED:
* HISTORY:
*
*****************************************************************************/

#include <stream.h>
#include <math.h>
#include <unistd.h>
#include <string.h>
#include <time.h>
#include "gpr.h"
#include "amips.h"
#include "Voxel.h"
#include "Slice.h"
#include "readFile.h" // solely for GETLINEs
#include "Attributes.h"
// #include "Environment.h"
// #include "ZShadeBuffer.h"

#define DEFAULT_VOLXSIZE 256
#define DEFAULT_VOLYSIZE 256
#define DEFAULT_VOLZSIZE 256

#define DEFAULT_WIDTH 1.0
#define DEFAULT_HEIGHT 1.0
#define DEFAULT_DEPTH 1.0
#define DEFAULT_INTERSLICE 5.0 // Typical for CT
```

```
#define DEFAULT_COLORMAPLENGTH 256

#define SQR(number) ((number) * (number))
#define MAGNITUDE(vector) ( sqrt( SQR(vector.x) + SQR(vector.y) + SQR(vector.z)) )

class Volume
{
        Voxel          *volume;
        unsigned int   volsizeX, volsizeY, volsizeZ;
        float          voxWidth, voxHeight, voxDepth, interslice;  // millimeters
ColorMap       volColorMap;
int        volColorMapLength; // Number of RGB triples in map
FILETYPE       ftype;

        unsigned int voxelAddress (unsigned int x, unsigned int y, unsigned int z)
        {
           if ( (x < volsizeX) AND (y < volsizeY) AND (z < volsizeZ) )
              return ( (volsizeX * volsizeY * z) + (volsizeX * y) + x );
           else
           {
              cout << "Volume::voxelAddress:  illegal volume indices.\n";
              cout << form ("x = %f, y = %f, z = %f not in legal range.\n",
                            x, y, z);
              return (0);
           }
        };

void loadSunRasterFiles (char *fpath, int fnum, int xfirst,
 int yfirst);
void loadVoxelLabFiles (char *fpath, int fnum, int xfirst,
int yfirst);
void loadChapelHillBinaryFile (char *fname, char *fpath, int fnum,
       int xfirst, int yfirst);
void loadChapelHillAsciiFile (char *fpath, int fnum, int xfirst,
      int yfirst);
void readControl(char *fname, char *fpath, int *Zstart,
 int *Xstart, int *Ystart);
//        void classifyVolume();

public:

Volume (unsigned int x_dim, unsigned int y_dim, unsigned int z_dim);
Volume (char *fname);
        ~Volume ()
        {
           if (volume NOT_EQUALS NULL)
              delete volume;
              // delete [volsizeX * volsizeY * volsizeZ] volume;
        };
```

```
//        void shadeVolume(Environment *, Attributes *);

        void getSize (unsigned int &x, unsigned int &y, unsigned int &z)
{
   x = volsizeX;
   y = volsizeY;
   z = volsizeZ;
};

void getDimensions (float &width, float &height, float &depth)
        {
           width  = voxWidth;
           height = voxHeight;
           depth  = voxDepth;
        };

void getInterSlice (float &distance)
        {
           distance = interslice;
        };

void setDimensions (float width, float height, float depth)
        {
           voxWidth  = width;
           voxHeight = height;
           voxDepth  = depth;
        };

void setInterSlice (float distance)
        {
           interslice = distance;
        };

void identify ();
void getColorMap (int &length, ColorMap &a_colormap);
Slice *getSlice (Orientationtype, unsigned int);
        void getVoxel (unsigned int x, unsigned int y,
       unsigned int z, Density *val);

        void putVoxel (unsigned int x, unsigned int y,
       unsigned int z, Density val);

void addCornerMarkers ();

// void render (Attributes *attr, Environment *env, ZShadeBuffer *zsbuf);

};

#endif
```

B.3 Landmark Classes

A crosshair is employed by the user to interact with a specific voxel and possibly select one as a landmark. If selected, the landmark is placed on the list of landmarks associated with the volume that contains the voxel, and the user interface is updated to reflect the selection. Various routines use pop-up menus to obtain the required levels of user interaction.

B.3.1 Crosshair.

```
#ifndef _CROSSHAIR_H
#define _CROSSHAIR_H

/*********************************************************************
* CLASS NAME:  Crosshair
* DESCRIPTION:  The Crosshair class is used by the User_Interface and the
* Landmark classes to identify anatomical landmarks, external markers,
* and reference axis systems which may be embedded within the Volume
* data.  It also controls the highlighting of Landmarks and Voxels in
* the various Views being displayed.
* SUPER CLASS:  Object
* PUBLIC METHODS:
* identify ();
* Interact_With_Voxel ();
* Refresh_Crosshair ();
* OPERATORS:  None.
* DATE WRITTEN:  15 September 91 by Pat Rizzuto
* VERSION:  1.0
* LAST MODIFIED:
* HISTORY:
*
*********************************************************************/

#include <gl.h>
#include "amips.h"
#include "Landmark_List.h"
#include "Menu.h"

//      Defines to set-up the Menu associated with this Crosshair

#define NUM_CROSSHAIR_MENU_ITEMS 9

#define SETEXTERNALMARKER 1 // See also corresponding
#define SETANATOMICLANDMARK 2 //  Menu text in the
#define CLEARLANDMARK 3 //  Crosshair constructor
#define CLEARALLLANDMARKS 4
#define IDXAXIS 5
#define IDYAXIS 6
#define IDZAXIS 7
```

```
#define CLEARAXISID 8
#define CHANGECROSSHAIR 9

// Defines to set-up the various Crosshair glyphs available for use

#define DEFAULT_CURSOR 0
#define SMALL_CROSSHAIR 1
#define BIG_CROSSHAIR 2
#define SCREEN_CROSSHAIR 3
#define FLAG 4

//      Defines to set-up the colormaps for each of the Crosshair types

#define DONT_CARE 0
#define min_INTENSITY 0
#define MAX_INTENSITY 255

class Crosshair
{
SliceIndex Crosshair_Voxel [3];
        Landmark_ListPtr    Crosshair_Landmark_List;
Menu *Crosshair_Menu;
MenuIdentifiertype Crosshair_Menu_ID;
unsigned short Crosshair_ColorMap [3][3],
Small_Crosshair_Glyph [32],
Big_Crosshair_Glyph [128],
Flag_Glyph [128],
Crosshair_Index;

void Activate_Crosshair ();
void Deactivate_Crosshair ();
void Build_Menu ();

void Set_External_Marker ();
void Set_Anatomic_Landmark ();
void ID_X_Axis ();
void ID_Y_Axis ();
void ID_Z_Axis ();
void Clear_Landmark ();
void Clear_All_Landmarks ();
void Clear_Axis_ID ();
void Change_Crosshair ();

public:

Crosshair (Landmark_ListPtr view2D_landmark_list);

        ~Crosshair ()
        {
   delete Crosshair_Menu;
//    Deactivate_Crosshair (); // Unnecessary, Window closed.
```

```
};

void identify ();
void Interact_With_Voxel (SliceIndex x_addr, SliceIndex y_addr,
  SliceIndex z_addr);
void Refresh_Crosshair ();
};

#endif
```

B.3.2 Landmark.

```
#ifndef _LANDMARK_H
#define _LANDMARK_H

/*****************************************************************************
* CLASS NAME:  Landmark
* DESCRIPTION:  The Landmark class is responsible for building individual
* anatomical landmarks and external markers, located in the Volume
* and set by the user, to be later used to perform Image Registration.
* SUPER CLASS:  ListNode
* PUBLIC METHODS:
* identify ();
*       GetIDnumber ();
// * GetDensity ();
* GetName ();
* GetType ();
*       GetVoxelAddress ();
* OPERATORS:  None.
* DATE WRITTEN:  15 September 91 by Pat Rizzuto
* VERSION:  1.0
* LAST MODIFIED:
* HISTORY:
*
*****************************************************************************/

#include <string.h>
#include "amips.h"
#include "dllist.h"

enum Landmarktype {EXTERNAL, ANATOMIC, REFERENCE};

typedef int LandmarkIDtype;

class Landmark : public ListNode
{
LandmarkIDtype Landmark_Identifier; // Mandatory
// Density Landmark_Density; // Mandatory
String Landmark_Name; // Optional
Landmarktype Landmark_Type; // Mandatory
SliceIndex Landmark_Address [3]; // Mandatory

public:

Landmark (LandmarkIDtype landmark_ID,
  String landmark_name, Landmarktype landmark_type,
  SliceIndex x_addr, SliceIndex y_addr, SliceIndex z_addr);
       ~Landmark ()
       {
       };
```

```
void identify ();
LandmarkIDtype GetID ()
{
   return (Landmark_Identifier);
};

/*
Density GetDensity ()
{
   return (Landmark_Density);
};
*/

String GetName ()
{
   return (strdup (Landmark_Name));
};

Landmarktype GetType ()
{
   return (Landmark_Type);
};

void GetVoxelAddress (SliceIndex &x_addr, SliceIndex &y_addr,
      SliceIndex &z_addr)
{
   x_addr = Landmark_Address [XVAL];
   y_addr = Landmark_Address [YVAL];
   z_addr = Landmark_Address [ZVAL];
};
};

typedef Landmark *LandmarkPtr;

#endif
```

B.3.3 Landmark List.

```
#ifndef _LANDMARK_LIST_H
#define _LANDMARK_LIST_H

/*****************************************************************************
* CLASS NAME:  Landmark_List
* DESCRIPTION:  The Landmark_List class is responsible for tracking and
* manipulating a doubly linked list of anatomical landmarks and
* external markers.
* ACKNOWLEDGEMENT:  I am grateful to John Brunderman for the
* use of his Doubly Linked List class, initially prepared for his
* CSCE 682 AAMRL project, and then used again for his thesis efforts.
* Many of the following methods resemble those from several of John's
* applications declared for those projects, and use the same conventions
* (e.g., the 'OffList' CP Scroll flag).
*
* SUPER CLASS:  List
* PUBLIC METHODS:
* identify ();
*       AddLandmark ();
*       RemoveLandmark ();
* ResetLandmarkList ();
* GetNextLandmark ();
*       GetLandmark ();
*       EndOfList ();
* Changed ();
* Changed_Then_Reset ();
* OPERATORS:  None.
* DATE WRITTEN:  15 September 91 by Pat Rizzuto
* VERSION:  1.0
* LAST MODIFIED:
* HISTORY:
*
*****************************************************************************/

#include "amips.h"
#include "dllist.h"
#include "Landmark.h"

#define SINGLE_STEP 1

class Landmark_List : public List
{
        LandmarkIDtype Next_Landmark_ID;
boolean Landmark_List_Changed,
OffList; // Used to control scrolling
//  along the List in either
//  direction.

public:
```

```
Landmark_List ()
{
          Next_Landmark_ID = 1;
   Landmark_List_Changed = FALSE;
   OffList = FALSE;
};

        ~Landmark_List ()
        {
   Clear ();
        };

void identify ();
void AddLandmark (String landmark_name,
  Landmarktype landmark_type,
  SliceIndex x, SliceIndex y, SliceIndex z);
void RemoveLandmark (LandmarkIDtype landmark_ID);
void RemoveLandmark (LandmarkPtr this_landmark);

void ResetLandmarkList ();
LandmarkPtr GetNextLandmark ();
LandmarkPtr GetLandmark (LandmarkIDtype landmark_ID);
LandmarkPtr GetLandmark (SliceIndex x, SliceIndex y, SliceIndex z);

boolean EndOfList ()
{
   return (OffList OR (GetCP () EQUALS 0));
};

boolean Changed ()
{
   return Landmark_List_Changed;
};

void Reset_Landmark_List_Changed_Flag ()
{
   Landmark_List_Changed = FALSE;
};
};

typedef Landmark_List *Landmark_ListPtr;

#endif
```

B.3.4 Menu.

```
#ifndef _MENU_H
#define _MENU_H

/*****************************************************************************
* CLASS NAME:  Menu
* DESCRIPTION:  The Menu class is responsible for building SGI pop-up menus,
* displaying those menus when requested and providing the caller
* with the desired menu option, and obtaining supplemental information
* using Keyboard input as required.
* SUPER CLASS:  ListNode
* PUBLIC METHODS:
* identify ();
*       Display_Menu ();
* Get_Keyboard_Input ();
* OPERATORS:  None.
* DATE WRITTEN:  15 September 91 by Pat Rizzuto
* VERSION:  1.0
* LAST MODIFIED:
* HISTORY:
*
*****************************************************************************/

#include "amips.h"
#include "textwindow.h"

#define MAX_NUM_MENU_ITEMS     10       // Completely arbitrary
#define NO_ACTION_ARGUMENT 0

#define INPUT_WINDOW_BACKGROUND_COLOR CYAN
#define INPUT_WINDOW_QUESTION_COLOR RED
#define INPUT_WINDOW_RESPONSE_COLOR MAGENTA

#define INPUT_WINDOW_SIZE_X 400
#define INPUT_WINDOW_SIZE_Y 200
#define INPUT_WINDOW_LOC_X (XMAXSCREEN / 2) - (INPUT_WINDOW_SIZE_X / 2)
#define INPUT_WINDOW_LOC_Y (YMAXSCREEN / 2) - (INPUT_WINDOW_SIZE_Y / 2)

#define QUESTION_UCOORD 20 // U & V
#define QUESTION_VCOORD INPUT_WINDOW_SIZE_Y - 50 // Coordinates
#define RESPONSE_UCOORD 20 // relative to
#define RESPONSE_VCOORD INPUT_WINDOW_SIZE_Y - 100 // Text Window

#define FONT_TYPE 23 // Helvetica - Bold
#define FONT_SIZE 12.0

#define NULLCHAR '\0' // ASCII nul
#define BACKSPACE 8 // ASCII bs
#define ENTER 13 // ASCII cr
#define SPACE 32 // ASCII ' '
```

```
#define TILDE 126 // ASCII '~'

#define MAX_RESPONSE_LENGTH 128

typedef long MenuIdentifiertype, MenuItemValuetype;

class Menu
{

public:

Menu (String title, String labels [], MenuItemValuetype values [],
              int number_of_items, MenuIdentifiertype &menu_identifier);

        ~Menu ()
        {
        };

void identify ();
MenuItemValuetype Display_Menu (MenuIdentifiertype menu_identifier);
void Get_Keyboard_Input (String question, String &response);
};

#endif
```

B.4 Image Registration Classes

The image registration subsystem encapsulates all registration activities, so a number of different techniques and algorithms may be installed while preserving the same outward appearance to the rest of the system.

B.4.1 Image Registration Subsystem.

```
#ifndef _IMAGE_REGISTRATION_SUBSYSTEM_H
#define _IMAGE_REGISTRATION_SUBSYSTEM_H

/*****************************************************************************
* CLASS NAME:  Image_Registration_Subsystem
* DESCRIPTION:  The Image_Registration_Subsystem class encapsulates
* the available Image Registration techniques into a single object
* with a common external interface.  Typically accessed by the
* User_Interface, the overall objective of the algorithms employed
* by this method is to compute the Image Registration transformation
* matrix needed to align one Volume with a second.
*
* SUPER CLASS:  Object
* PUBLIC METHODS:
* identify ();
* Select_LUD_Method ();
* Select_QSH_Method ();
* Build_Transformation ();
* OPERATORS:  None.
* DATE WRITTEN:  6 November 91 by Pat Rizzuto
* VERSION:  1.0
* LAST MODIFIED:
* HISTORY:
*
*****************************************************************************/

#include <gl.h>
#include "amips.h"
#include "Landmark_List.h"
#include "LU_Decomposition_Registration.h"

enum Registrationtype {LUD, NYIT_QSH};

class Image_Registration_Subsystem
{
LU_Decompose_RegPtr LU_Decomposition_Method;
Registrationtype Registration_Technique;

public:

Image_Registration_Subsystem ();
```

```
        ~Image_Registration_Subsystem ()
{
};

void identify ();

void Select_LUD_Method () { Registration_Technique = LUD; }
void Select_QSH_Method () { Registration_Technique = NYIT_QSH; }

void Build_Transformation (Landmark_ListPtr landmark_list_1,
   Landmark_ListPtr landmark_list_2,
   Matrix     IR);
};

#endif
```

B.4.2 Landmark Mapping.

```
#ifndef _LU_DECOMPOSITION_REGISTRATION_H
#define _LU_DECOMPOSITION_REGISTRATION_H

/*****************************************************************************
* CLASS NAME:  LU_Decomposition_Registration
* DESCRIPTION:  The LU_Decomposition_Registration class is one form of Image
* Registration which may be performed by the Image_Registration_
* Subsystem.  The objective is to determine the elements of the
* 4x4 matrix IR:
*
* --- ---
* |   |
* | a   e   i   0 |
* | b   f   j   0 |
*  IR  =  | c   g   k   0 |
* | d   h   1   1 |
* |   |
* --- ---
*
* This matrix is then used to modify the viewing parameters associated
* with one of the Volumes, so that both Volumes will be rendered with
* the same orientation, and corresponding Voxels may be accessed for
* further analyses.
* This process uses LU decomposition (AKA factorization) and
* backsubstitution to solve the general linear system of equations
* Ax = B.  However, following a Paul Heckbert's lead (using the same
* procedure to determine a static mapping for various points from
* screen space to texture space), the procedure accepts two sets
* of 3-D points (the two Landmark_Lists), and uses them to solve for
* the above matrix using the formula:
*
*    Landmark_List_1 [x y z w] = Landmark_List_2 [x y z w] * IR
*
* Since no global scaling or perspective operations will be performed
* using the IR matrix, the last column is cleared to its identity
* values, and we only have to solve for twelve unknowns a - l, not
* 16.  We therefore need 12 equations, which we obtain from four
* points in the Landmark_Lists as follows:
*
* ---       ---   --- ---     --- ---
* |         |   |     |     |     |
* |x0  y0  z0   1   0   0   0   1   0   0   0   1|   | a |     | x0' |
* | 0   0   0   1  x0  y0  z0   1   0   0   0   1|   | b |     | y0' |
* | 0   0   0   1   0   0   0   1  x0  y0  z0   1|   | c |     | z0' |
* |x1  y1  z1   1   0   0   0   1   0   0   0   1|   | d |     | x1' |
* | 0   0   0   1  x1  y1  z1   1   0   0   0   1|   | e |     | y1' |
* | 0   0   0   1   0   0   0   1  x1  y1  z1   1|   | f |     | z1' |
* |x2  y2  z2   1   0   0   0   1   0   0   0   1| * | g | =   | x2' |
* | 0   0   0   1  x2  y2  z2   1   0   0   0   1|   | h |     | y2' |
```

```
* | 0   0   0   1   0   0   0   1  x2  y2  z2   1|   | i |     | z2' |
* |x3  y3  z3   1   0   0   0   1   0   0   0   1|   | j |     | x3' |
* | 0   0   0   1  x3  y3  z3   1   0   0   0   1|   | k |     | y3' |
* | 0   0   0   1   0   0   0   1  x3  y3  z3   1|   | l |     | z3' |
* |       |   |     |     |     |
* ---     ---   --- ---     --- ---
*
* where the points {(x0,y0,z0) - (x3,y3,z3)} are from Landmark_List_1
* and {(x0',y0',z0') - (x3',y3',z3')} are from Landmark_List_2, and
* the '1' following each set of point elements (e.g., 'x0 y0 z0 1')
* is 'w' from the standard point representation [x y z w].  Since, as
* stated above, there will be no global scaling or perspective using
* these points, w = 1 in all cases.
* The LU decomposition is a modified version of standard
* Gaussian elimination, which transforms the above matrix into a
* Lower Triangular or Upper Triangular matrix (LU).  This is then
* easily solved using backsubstitution.
*
* Credit for this code goes to a number of sources.  As
* mentioned earlier, Paul Heckbert, in a 1983 New York Institute of
* Technology (NYIT) technical memo and his Nov 1986 IEEE CG&A survey
* of texture mapping techniques, talked about using such a process to
* create the mapping between screen space (in x,y coordinates) and a
* texture space (indexed from 0 to 1).  Phil Amburn (AFIT Lt Col)
* incorporated that technique into the AFIT General Purpose Renderer
* (GPR) for TextureMappedPolygons.  He used code provided by Maj
* Dave Robinson to actually perform the LU decomposition, which was
* based, in part, on routines found in 'Numerical Recipes in C.'
*
* SUPER CLASS:  Object
* ATTRIBUTES:
* Landmark_Matrix - the 12x12 matrix (A) composed of the points from
*   Landmark_List_1, premultiplying the 12x1 vector (x)
*   [a - 1] on the LHS of the equation.
* List_2_Landmarks - the 12x1 vector (B) composed of the points from
*   Landmark_List_2, on the RHS of the equation.
* Number_of_Row_Interchanges - Flag used by the backsubstitution
*   routine, set by the decomposition routine, and
*   stores +1 if an even number of rows were inter-
*   changed during the decomposition, -1 if odd.
* Index - a 1x12 integer vector storing the column number containing
*   the pivot value for each row in the Landmark_
*   Matrix, used for Gaussian elimination.
* PUBLIC METHODS:
* identify ();
* Determine_IR_Transform ();
* OPERATORS:  None.
* DATE WRITTEN:  2 November 91 by Pat Rizzuto
* VERSION:  1.0
* LAST MODIFIED:
* HISTORY:
```

```
*
*****************************************************************************/

#include <gl.h>
#include <math.h> // Necessary for fabs ()
#include <malloc.h> // Necessary for malloc () and free ()
#include "amips.h"
#include "Landmark_List.h"
// #include "Menu.h"

//      Defines to set-up the Menu associated with this LU_Decomposition_
//   Registration class.

// #define NUM_DECOMP_REG_MENU_ITEMS        4

// #define xxxxx 1       // See also corresponding
// #define xxxxx 2       //  Menu text in the
// #define xxxxx 3       //  LU_Decomposition_Registration
// #define xxxxx 4 //  constructor

// Defines to establish allowable dimensions for matrices and vectors

#define NUM_ROWS_LOWER 1
#define NUM_ROWS_UPPER 12
#define NUM_COLUMNS_LOWER 1
#define NUM_COLUMNS_UPPER 12

#define TINY 1.0e-20 // Limit of precision for this problem

class LU_Decomposition_Registration
{
float **Landmark_Matrix;
        float *List_2_Landmarks;
float Number_of_Row_Interchanges;
int *Index;
//        Menu                    *LU_Decompose_Reg_Menu;
//        MenuIdentifiertype      LU_Decompose_Reg_Menu_ID;

float **nrmatrix ();
float *nrvector ();
int *nrivector ();
void nrfree_vector (float *this_vector);
void LU_Decompose ();
void LU_Backsubstitute ();
//        void Build_Menu ();

public:

LU_Decomposition_Registration ();

        ~LU_Decomposition_Registration ()
```

```
        {
};

void identify ();

void Determine_IR_Transform (Landmark_ListPtr landmark_list_1,
     Landmark_ListPtr landmark_list_2,
     Matrix       IR);
};

typedef LU_Decomposition_Registration *LU_Decompose_RegPtr;

#endif
```

B.5 Volume Visualization Classes

As with the image registration system, the volume visualization system abstracts all the details of the various surface and volume rendering tools that may be used to display three-dimensional images of volumes.

B.5.1 Volume Visualization Subsystem.

```
#ifndef _VOLUME_VISUALIZATION_SUBSYSTEM_H
#define _VOLUME_VISUALIZATION_SUBSYSTEM_H

/*****************************************************************************
* CLASS NAME:  Volume_Visualization_Subsystem
* DESCRIPTION:  The Volume_Visualization_Subsystem class encapsulates
* the available Surface and Volume Rendering techniques into a single
* object with a common external interface.  Typically accessed by the
* User_Interface, the overall objective of the algorithms employed
* by this method is to render a 3-D Volume composed of Voxels.
*
* SUPER CLASS:  Object
* PUBLIC METHODS:
* identify ();
* Select_Vanilla_Marching_Cubes ()
* Select_Distribute_Ray_Caster ()
* OPERATORS:  None.
* DATE WRITTEN:  6 November 91 by Pat Rizzuto
* VERSION:  1.0
* LAST MODIFIED:
* HISTORY:
*
*****************************************************************************/

#include <gl.h>
#include "amips.h"

enum Vol_Visualizationtype {DIST_RAY_CAST, VAN_MARCH_CUBE};

class Volume_Visualization_Subsystem
{
// Brightbill_Renderer *Distributed_Ray_Caster;
// Parrott_Renderer *Vanilla_Marching_Cubes;
Vol_Visualizationtype Rendering_Technique;


public:

Volume_Visualization_Subsystem ();

        ~Volume_Visualization_Subsystem ()
{
```

```
};

void identify ();

void Select_Distributed_Ray_Caster ()
{
   Rendering_Technique = DIST_RAY_CAST;
}

void Select_Vanilla_Marching_Cubes ()
{
   Rendering_Technique = VAN_MARCH_CUBE;
}
};

#endif
```

Bibliography

1. Bajcsy, R. *et al.* "A Computerized System for the Elastic Matching of Deformed Radiographic Images to Idealized Atlas Images," *J. Computer Assisted Tomography*, 7(4):618–625 (August 1983).
2. Brightbill, Capt Patricia L. *Distributed Ray Casting for High-Speed Volume Rendering*. MS thesis, AFIT/GCS/ENG/91D. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, planned for publication in March 1992.
3. Brooks, Frederick P. Jr. *The Mythical Man-Month*. Reading MA: Addison-Wesley Publishing Co., 1975.
4. Davis, Alan M. *Software Requirements Analysis and Specification*. Englewood Cliffs NJ: Prentice-Hall, Inc., 1990.
5. Drebin, Robert A. *et al.* "Volume Rendering," *ACM Computer Graphics*, *22*(4):65–74 (August 1988).
6. Evans, A. C. *et al.* "Anatomical-Functional Correlation Using an Advanced MRI-Based Region of Interest Atlas with Positron Emission Tomography," *J. Cerebral Blood Flow and Metabolism*, *8*(8):513–530 (August 1988).
7. Farrell, Edward J. *et al.* "Color 3-D Imaging of Normal and Pathologic Intracranial Structures," *IEEE Computer Graphics & Applications*, *4*(5):5–17 (September 1984).
8. Fellingham, L. L. *et al.* "Three-dimensional imaging: Applications to pre-operative and radiation therapy planning." *Proceedings of SPIE, Volume 602, Biostereometrics '85*, edited by A. M. Coblentz and Robin E. Herron. 320–325. Bellingham WA: Society of Photo-Optical Instrumentation Engineers, 1985.
9. Foley, James D. *et al. Computer Graphics: Principles and Practice* (Second Edition). Reading MA: Addison-Wesley Publishing Co., 1990.
10. Fox, Peter T. *et al.* "A Stereotactic Method of Anatomical Localization for Positron Emission Tomography," *J. Computer Assisted Tomography*, *9*(1):141–153 (January/February 1985).
11. Frieder, Gideon *et al.* "Back-to-Front Display of Voxel-Based Objects," *IEEE Computer Graphics & Applications*, *5*(1):52–59 (January 1985).
12. Gamboa-Aldeco, Arturo *et al.* "Correlation of 3D Surfaces from Multiple Modalities in Medical Imaging." *Proceedings of SPIE, Volume 626 (Part 2), Medicine XIV/PACS IV*, edited by Roger H. Schneider and Samuel J. Dwyer III. 467–473. Bellingham WA: Society of Photo-Optical Instrumentation Engineers, 1986.
13. Heckbert, Paul S. "Survey of Texture Mapping," *IEEE Computer Graphics & Applications*, *6*(11):56–67 (August 1986).
14. Hu, X. *et al.* "Volumetric Rendering of Multimodality, Multivariable Medical Imaging Data." *Proceedings of the Chapel Hill Workshop on Volume Visualization*. 45–49.

Chapel Hill NC: Computer Science Department of the University of North Carolina, Chapel Hill, 1989.

15. Humphrey, Watts S. *Managing the Software Process*. Reading MA: Addison-Wesley Publishing Co., 1989.

16. Levin, David N. *et al.* "Retrospective Geometric Correlation of MR, CT, and PET Images," *Radiology*, *169*(3):817–823 (December 1988).

17. Levoy, Marc. "Direct Visualization of Surfaces from Computed Tomographic Data." *Proceedings of SPIE, Volume 914 (Part B), Medical Imaging II: Image Data Management and Display*, edited by Roger H. Schneider and Samuel J. Dwyer III. 828–841. Bellingham WA: Society of Photo-Optical Instrumentation Engineers, 1988.

18. Levoy, Marc. "Display of Surfaces from Volume Data," *IEEE Computer Graphics & Applications*, *8*(3):29–37 (May 1988).

19. Maguire, Gerald Q. *et al.* "Graphics Applied to Medical Image Registration," *IEEE Computer Graphics & Applications*, *11*(2):20–28 (March 1991).

20. Marcus, Aaron and Andries van Dam. "User-Interface Developments for the Nineties," *IEEE Computer*, *24*(9):49–57 (September 1991).

21. McShan, Daniel L. and Benedick A. Frass. "Integration of Multi-Modality Imaging for Use in Radiation Therapy Treatment Planning." *Proceedings of the 2nd International Symposium, Computer Assisted Radiology (CAR) '87*, edited by H. U. *et al* Lemke. 300–304. Berlin: Springer-Verlag, 1987.

22. Neiw, H. M. *et al.* "Automated Three-Dimensional Registration of Medical Images." *Proceedings of SPIE, Volume 1445, Image Processing*, edited by Murray H. Loew. 259–264. Bellingham WA: Society of Photo-Optical Instrumentation Engineers, 1991.

23. Ney, Derek R. *et al.* "Volumetric Rendering of Computer Tomography Data: Principles and Techniques," *IEEE Computer Graphics & Applications*, *10*(3):24–32 (March 1990).

24. Noz, Marilyn E. *et al.* "Graphical Aids for Tomographic Image Correlation." *Proceedings of the 1st International Symposium, Computer Assisted Radiology (CAR) '85*, edited by H. U. Lemke. 608–611. Berlin: Springer-Verlag, 1985.

25. Parrott, Capt Rob. *Evaluation of Scalar Value Estimation Techniques for 3-D Medical Imaging*. MS thesis, AFIT/GCS/ENG/91D–17. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1991.

26. Pelizzari, C. A. *et al.* "Accurate Three-Dimensional Registration of CT, PET and/or MR Images of the Brain," *J. Computer Assisted Tomography*, *13*(1):20–26 (January/February 1989).

27. Ratib, Osman *et al.* "A New Technique for Elastic Registration of Tomographic Images." *Proceedings of SPIE, Volume 914 (Part A), Medical Imaging II: Image Formation, Detection, Processing, and Interpretation*, edited by Roger H. Schneider and Samuel J. Dwyer III. 452-455. Bellingham WA: Society of Photo-Optical Instrumentation Engineers, 1988.

28. Rhodes, Michael L. *et al.* "Computer Graphics and an Interactive Stereotactic System for CT-Aided Neurosurgery," *IEEE Computer Graphics & Applications*, *3*(5):31–38 (August 1983).

29. Robb, R. A. and C. Barillot. "Interactive Display and Analysis of 3-D Medical Images," *IEEE Transactions on Medical Imaging*, *8*(3):217–226 (September 1989).

30. Ross, D. T. and K. E. Schoman Jr. "Structured Analysis for Requirements Definition," *IEEE Transactions on Software Engineering*, *3*(1):6–15 (January 1977).

31. Schad, Lothar R. *et al.* "Three Dimensional Image Correlation of CT, MR, and PET Studies in Radiotherapy Treatment Planning of Brain Tumors," *J. Computer Assisted Tomography*, *11*(6):948–954 (November/December 1987).

32. Schiers, C. *et al.* "Interactive 3D Registration of Image Volumes from Different Sources." *Proceedings of the 3rd International Symposium, Computer Assisted Radiology (CAR) '89*, edited by H. U. Lemke and M. L. Rhodes. 666–670. Berlin: Springer-Verlag, 1989.

33. Simpson, Capt Dennis. *An Application of the Object-Oriented Paradigm to a Flight Simulator*. MS thesis, AFIT/GCS/ENG/91D-22. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1991.

34. Sommerville, Ian. *Software Engineering* (Third Edition). Reading MA: Addison-Wesley Publishing Co., 1989.

35. Stytz, M. R. and O. Frieder. "Three-Dimensional Medical Imaging Modalities: An Overview," *Critical Reviews in Biomedical Engineering*, *18*(1):27–54 (July 1990).

36. Stytz, Martin R. and Ophir Frieder. "Computer Systems for Three-Dimensional Diagnostic Imaging: An Examination of the State of the Art," *Critical Reviews in Biomedical Engineering*, *19*(1):1–45 (July 1991).

37. Talton, D. A. *et al.* "Volume Rendering Algorithms for the Presentation of 3D Medical Data." *Proceedings, Volume III, NCGA's Computer Graphics '87 Eighth Annual Conference and Exposition*. 119–128. Fairfax VA: National Computer Graphics Association, 1987.

38. Tiede, Ulf *et al.* "Investigation of Medical 3D-Rendering Algorithms," *IEEE Computer Graphics & Applications*, *10*(3):41–53 (March 1990).

39. Toennies, Klaus D. *et al.* "Surface Registration for the Segmentation of Implanted Bone Grafts." *Proceedings of the 3rd International Symposium, Computer Assisted Radiology (CAR) '89*, edited by H. U. Lemke and M. L. Rhodes. 381–385. Berlin: Springer-Verlag, 1989.

40. Toennies, Klaus D. *et al.* "Registration of 3D Objects and Surfaces," *IEEE Computer Graphics & Applications*, *10*(5):52–62 (May 1990).

41. Valentino, D. J. *et al.* "Mapping Brain Function to Brain Anatomy." *Proceedings of SPIE, Volume 914 (Part A), Medical Imaging II: Image Formation, Detection, Processing, and Interpretation*, edited by Roger H. Schneider and Samuel J. Dwyer III. 445–451. Bellingham WA: Society of Photo-Optical Instrumentation Engineers, 1988.

42. Vandermeulen, D. *et al.* "A Prototype Medical Workstation for Computer Assisted Stereotactic Neurosurgery." *Proceedings of SPIE, Volume 1234, Medical Imaging IV: PACS System Design and Evaluation*, edited by Samuel J. Dwyer III and R. Gilbert Jost. 887–893. Bellingham WA: Society of Photo-Optical Instrumentation Engineers, 1990.

43. Watt, Alan. *Fundamentals of Three-Dimensional Computer Graphics.* Reading MA: Addison-Wesley Publishing Co., 1989.