NORTHWESTERN UNIVERSITY

Registration Graphs: A Language for Modeling and Analyzing Registration in Image-Guided Surgery

A DISSERTATION

SUBMITTED TO THE GRADUATE SCHOOL

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

for the degree

DOCTOR OF PHILOSOPHY

Field of Mechanical Engineering

By

Jon Thomas Lea

EVANSTON, ILLINOIS

December 1998

ABSTRACT

Registration Graphs: A Language for Modeling and Analyzing Registration in Image-Guided Surgery

Jon Thomas Lea

Computer-assisted surgical systems must bring diagnostic images, surgical plans, patient anatomy, surgical tools, robots, vision systems, and other components into accurate alignment with one another. Multi-step registration procedures have been devised, which are difficult to analyze, or even to describe concisely.

A method for diagramming registration strategies and procedures makes descriptions straightforward and simplifies analysis. A notation that uses a graph theoretic framework consisting of two primary elements: *features*, representing objects (or parts of objects); and *links*, representing measurement actions is introduced. Connectivity properties of the resulting *registration graph* are readily determined by inspection.

To provide *quantitative* as well as *qualitative* analysis of a registration strategy, a model for propagating measurement uncertainties through the graph is presented. The model involves the first and second statistical moments, mean and covariance. Methods for combining the means and covariances of serial as well as parallel measurements are shown.

The rules and algorithms governing the registration graph and uncertainty analysis are implemented as a C++ library and Windows program. A case study of a registration strategy verifies that the library and program are effective, and shows that registration is more than just a "final alignment between tool and patient."

Acknowledgments

First and foremost, I would like to thank my advisor, Prof. Michael Peshkin, for his trust and support through my career as a graduate student and member of the Laboratory for Intelligent Mechanical Systems. It was he who saw the potential of the registration graph when I showed him some sketchy "musings" I thought helped explain registration. I hope that a just a smidgen of his amazing insight has rubbed off on me.

I also would like to thank Prof. Ed Colgate, co-director of LIMS and member of my dissertation committee, whose knowledge and rigorous approach to problems have been ever so instructive. Thanks also to my third dissertation committee member, Prof. Scott Delp, who always brought a medically minded perspective to the table.

Over the course of my graduate work, I have had the pleasure of meeting and discussing work in the field of computer-assisted surgery with many people, including Russ Taylor, Dave Simon, Pete Loan. Thank you for keeping my interest strong in this field.

My fellow lab-mates also deserve mention as they were always willing to discuss the technical, political, and philosophical that contribute to the graduate student experience. In no particular order, thanks to Mike Brokowski, Ken Grace, Pat Jensen, Keith Anderson, Paul Millman, Ambarish Goswami, Witaya Wannasuphoprasit, Julio Santos-Munné, Carl

Moore, Bernie Reger, and Mike Brown. Many others remain unnamed here but know who they are - thanks to you also.

A special thanks to Prof. Phil Guichelaar of Western Michigan University, whose teaching style and personal friendship inspired me to choose the path that led to the Ph.D.

I would like to extend a long-deserved thank you to my parents, Tom and Louise. Their values and lessons taught me to enjoy both work *and* play, the two of which I often cannot distinguish from one another. I hope I have made them proud with this achievement; it is the least I could do for all they have done for me. Thanks also to my in-laws, Murray and Hazel, for their encouragement and understanding throughout my studies.

Finally, my most heart-felt thanks go to my wife and soulmate, Sarah. Without her undying love, patience, and support, this would never have been possible.

This work was supported by a Walter P. Murphy Fellowship from the McCormick School of Engineering, and grants from the National Science Foundation and the Whitaker Foundation.

List of Abbreviations

2D	two dimensional
3D	three dimensional
CAD	computer-aided drafting
CMM	coordinate measuring machine
CRT	cathode ray tube
CT	computed tomography
EKG	electrocardiogram
II	image intensifier
LCD	liquid crystal display
LED	light emitting diode
MRI	magnetic resonance imaging
RPY	roll, pitch, yaw (axes for rotation)
SCARA	Selective Compliance Assembly Robot Arm
STL	Standard Template Library
THR	total hip replacement
TKR	total knee replacement

To Sarah Kathryn.

Contents

List of Tables

List of Figures

1 Introduction

During work toward my master's degree I was involved in the development of a computer-assisted surgery system for total knee replacement [1][2]. The total knee replacement system utilized preoperatively obtained CT data to construct a 3D model of the patient's femur on a computer. Other 3D models of knee implants - titanium components used to replace the articulating surfaces of the knee joint - could be tried out on the bone model to ascertain their fit. Once a particular implant was chosen, the positions of the necessary cuts and holes to accommodate the implant as well as three widely-spaced fiducial points (defined by small implanted pins) on the bone were digitally stored along with the 3D model of the femur.

In the operating room, a robot was used to measure the three fiducial points on the patient's immobilized femur. These points were then matched to those in the 3D bone model, thereby *registering* the preoperatively determined resection positions with the intraoperative position of the femur. The robot then displayed where resections were to be made by aligning a cutting guide against the femur, and the surgeon made the resections with a conventional surgical saw.

It is obvious that unless registration was performed successfully, the subsequent display of resection positions by the robot would be impossible. A more dangerous prospect lay in registration being performed inaccurately, resulting in erroneous display of the resection positions. Sources of inaccuracy are not always apparent, and can be found not only in the intraoperative measurement step, which completes registration, but also throughout all measurements that make up the architecture of the computer-assisted surgery system. This issue is not unique to total knee system described above - it is a common trait of all computer-assisted surgery systems.

Another aspect of registration causing quite a bit of confusion in the computer-assisted surgery community is that registration strategies vary widely. While the predominant theme of registration in the context of computer-assisted surgery is to align image data, be it CT or MRI scans, fluoroscopic x-rays, or ultrasound images, with devices such as robots or computer-tracked hand-held tools, there are many minor aspects in any one registration strategy which set it apart from others. At conferences, an individual presenting work done with a system for a particular surgery often will find that members of the audience tend to make false assumptions about the registration strategy based on experiences with their own systems, often developed for completely different surgeries and using different hardware. In addition, the computer-assisted surgery field is inherently multidisciplinary and the engineers and doctors in attendance speak in different technical vernaculars. This leads to confusion within the community as to what actually composes a registration strategy for a given computer-assisted surgery system.

It is my intent in this thesis to address these two problems - a) understanding a registration strategy's architecture and b) understanding effects that errors have on the system's overall accuracy - through the use of an illustrative construct called the *registration graph*.

1.1 Computer-Assisted Surgery

To the uninitiated it would seem that nearly all instruments found in an operating room fall under the banner of computer-assisted surgery. For instance, an EKG machine contains microprocessors as well as a CRT or LCD screen to display electrical signals corresponding to contractions of a patient's heart. Devices as simple as a modern thermometer employ thermoelectric sensors and an LED readout to measure a patient's body temperature. While these instruments could be considered "computerized" and may be used in surgery, they are not elements of a computer-assisted surgery system.

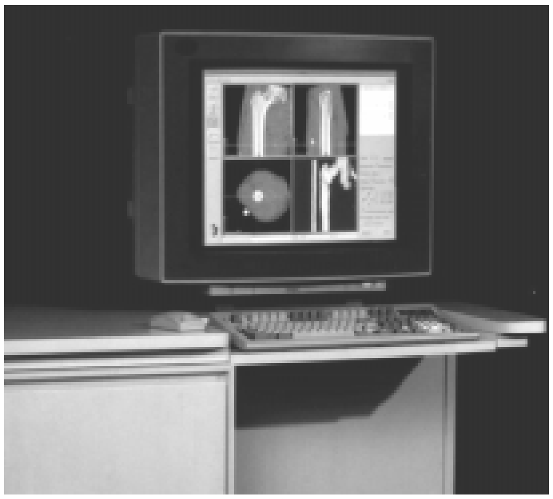
For the purposes of this thesis and in accordance with the rapidly emerging field, the scope of computer-assisted surgery will include those systems of instruments used to help the surgeon move tools and/or implants into desired positions with respect to the patient's anatomy. To define what comprises such systems, the following working definition is introduced:

> Computer-assisted surgery systems involve some components that image, measure, or control position. These include robots, articulated pointers, optical tracking devices, fluoroscopes, and CT or MRI scanners. Other components cannot be so described, but are equally important. These include bones, tissues, fiducials, cutting tools, and many others. Such components are collectively used to help the surgeon position a tool on the patient's anatomy in a manner that is easier, more accurate, and often less invasive than conventional techniques allow.

It is impossible to list each and every component that might be included in a computer-assisted surgery system. Often it is the novel integration of components by a designer that yields a new computer-assisted surgery system. New technologies are always being introduced that make new designs possible. A look at two commercially available systems will show us the diversity of computer-assisted surgery systems and will complement our working definition of the computer-assisted surgery system.

The Integrated Surgical Systems RoboDoc™ system [3][4] (see Figure 1.1) is used in total hip replacement and was developed out of a collaboration between Howard Paul at the University of California at Davis and Russ Taylor at IBM (now at Johns Hopkins University) [5]. The system is used to mill a cavity into the proximal end of the femur. The cavity will accept the long stem of an implant used to replace the femoral head. It is hoped that a precisely shaped cavity will reduce gaps between bone and implant, thereby promoting bone growth into porous surfaces on the implant and eliminating the need for bone cement.

A CT scan is performed preoperatively with small titanium landmark screws placed on the femur. The conventional planning technique is replaced with a graphic computer workstation, in which the CT data are used to display a 3D representation of the hip joint. Using this workstation, a model of the implant is placed in a desired position on the representation of the femur, and the CT coordinates of the landmark screws and the implant can be saved. The surgeon can choose different implants and manipulate them onscreen until completely satisfied with the preoperative plan.


(a)

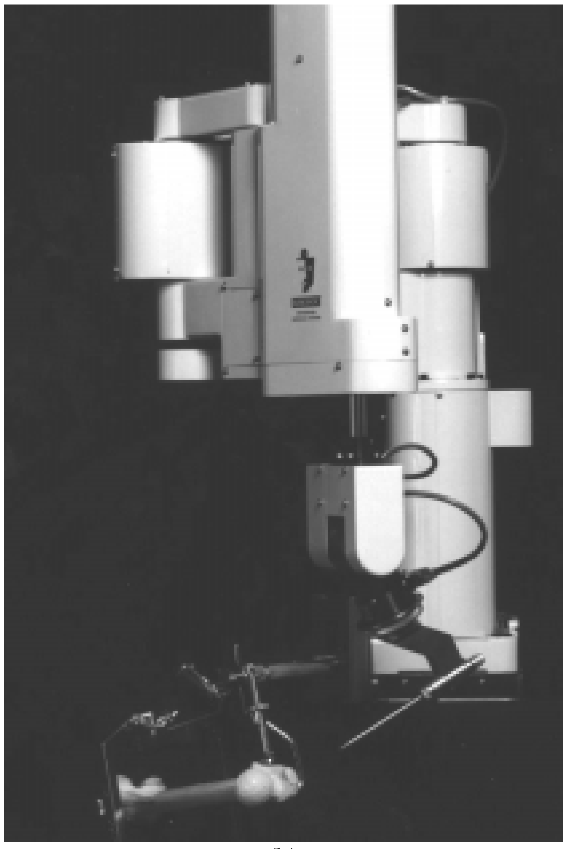
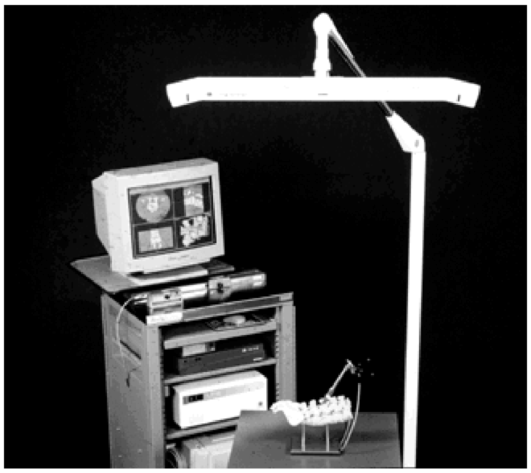

(b)

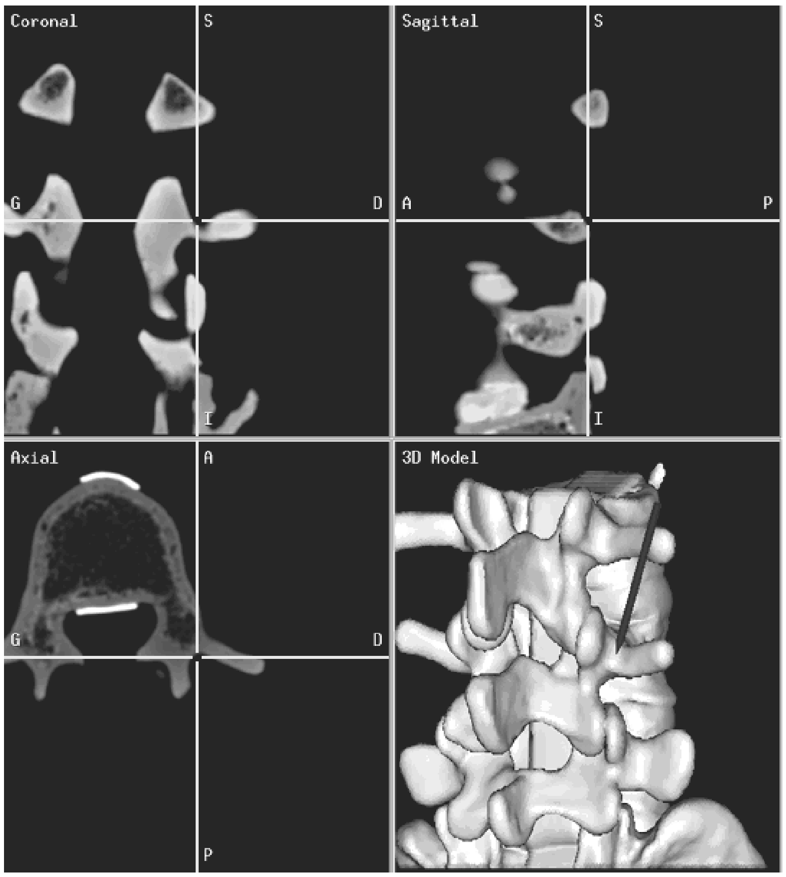
Figure 1.1: (a) The RoboDoc™ system employs a computer workstation to preoperatively prescribe the position of the femoral component in total hip replacement. (b) In the operating room, a modified SCARA robot is used to mill out the cavity for the femoral component.

Once in the operating room, the opening incisions are made and the femur is immobilized using a custom bone clamp attached to the system's base unit. The system uses a modified SCARA robot outfitted with an additional pitch-axis joint at the robot's wrist. Other modifications include a slower, weaker drive system (large forces at the tool tip are not required) and redundant encoders to ensure safety. A probe mounted to the robot's wrist is guided to the landmark screws and the position of each screw is determined in the robot's workspace. These positions are then matched to those of the screws in the CT data, registering the preoperative femoral implant position with the intraoperative position of the femur. A high-speed burring tool replaces the probe on the robot, and the cavity is milled out. A force sensor, affixed to the wrist of the robot between the last joint and the high speed burr, ensures that the operation proceeds at the proper cutting speed through both the soft trabecular bone and the harder cortical bone [6]. Dimensional accuracy of the cavity has reportedly increased by an order of magnitude with resultant tolerances under 0.05 mm. The system is widely used in Europe, and will be introduced domestically pending FDA approval.

The StealthStation™ (see Figure 1.2) by Surgical Navigation Technologies (a subsidiary of the Sofamor Danek Group) is used in surgeries of the head and spine, and evolved out of work done by Richard Bucholz at the St. Louis University School of Medicine [7]. Stereotactic methods have long been used by neurosurgeons to pinpoint the position of structures inside the skull that have been identified preoperatively via a CT or MRI scans. The StealthStation provides an intuitive and user-friendly alternative to conventional stereotactic tools.



(a)



(b)

Figure 1.2: (a) The StealthStation™ by Surgical Navigation Technologies uses an optical localizer and computer to track the position of a probe fitted with infrared LEDs. (b) The position of the tip of the probe is shown on the computer screen as both a crosshair in the appropriate CT or MRI image and as pointer on a 3D model of the bone (figure b).

One application that the StealthStation is well suited to is spinal fixation of the lower back. This procedure requires screws to be implanted into slender columns of bone - the pedicles - on each vertebra involved. Conventional techniques for implanting the screws rely on visually orienting the screws with respect to local anatomical landmarks on the surface of the vertebra. The StealthStation allows the surgeon to directly visualize the final position of the screw in the pedicle by displaying the screw's trajectory on images derived from a CT or MRI scan.

In use, a CT or MRI scan is obtained and the surgeon studies the images preoperatively in the conventional manner. Intraoperatively, the CT/MRI data is loaded into a graphic computer workstation and displayed in four different views onscreen. The patient is prepared as usual by making an incision along the spine and retracting the surrounding tissues.

An optical localizer is used to track the positions of hand-held tools and other objects. The optical localizer achieves this by triangulating the position of infrared LEDs affixed to each tool. Tools outfitted with LEDs include pointing probes, drills, and drill guides, as well as a special bone 'flag' that is clamped to a relatively large finlike structure, the spinous process, on the vertebra. Using a pointing probe, the surgeon digitizes certain anatomical landmarks on the vertebra. The coordinates of these landmarks are matched to like coordinates in the CT/MRI data, and once done provide the needed registration between the intraoperative position of the vertebra in the optical localizer's reference frame and the preoperative position of the vertebra in the CT/MRI reference frame. The vertebral 'flag' provides continuous feedback to the system as to the vertebra's position, thus eliminating the need to immobilize the vertebra. Once registration is complete, the surgeon can point to features on the vertebra using the pointing probe and the corresponding position of the probe's tip is shown superimposed on the views of the CT/MRI data on the workstation screen. The surgeon can align a drill or drill guide in a similar manner and perform the necessary operations. The screws are affixed in the conventional manner.

These two systems are quite representative of both the hardware used and approaches taken in the computer-assisted surgery field. An excellent reference guide and overview of recent works can be found in [8]. An earlier survey of robots in medicine provides some historical insight and can be found in [9].

1.2 Registration

Simply put, registration is a series of measurements and alignments of the elements (e.g., 2D x-rays, 3D CT scans, robotic manipulators) of a computer-assisted surgery system. Many of these measurements must be performed each time the system is used, i.e., for every surgery. This metric aspect of registration is readily apparent. Other measurements include machined dimensions and calibration procedures performed when the system is being built or set up in a hospital. A more subtle aspect of registration is the temporal arrangement of these measurements and alignments, and the dependence that certain measurements have on previously obtained measurements.

As an example, consider the registration strategy employed in the RoboDoc system. When using the system in the operating room, one typically takes for granted that the kinematic model of the robot includes not only the parameters used to drive each joint into desired positions, but also the parameters that describe where the tool (e.g., the probe) is mounted on the robot's wrist. Without a model of the tool as mounted on the robot, the intraoperative measurement of landmark screw positions would be impossible. The model of the tool must be known, and must be known *before* the landmark screw positions are measured.

It is not necessary to completely illustrate registration at this time. The nature and scope of registration will be naturally revealed in Chapter 2 as the registration graph is developed.

1.3 Overview of the Thesis

The intended audience of this thesis is the multidisciplinary constituency of the computer-assisted surgery field, that is, engineers *and* surgeons. As mentioned earlier, registration strategies are often misunderstood and it is hoped that the registration graph will provide a foundation for a common vocabulary among all disciplines involved. I assume that readers of this thesis are somewhat familiar with measuring devices used in computer-assisted surgery, such as robots and CT scanners, as well as surgical procedures like total knee replacement or biopsy of a tumor within the brain.

In Chapter 2, the registration graph is introduced and the rules governing registration are presented as the registration graph is developed through the first half of the chapter. The chapter also contains a brief survey of registration strategies in which key aspects of particular system's registration method are noted by inspection of its corresponding registration graph. This chapter should be of interest to all who would like a better understanding of what registration is.

Chapter 3 presents the theory behind a metric model for ascertaining a registration strategy's viability. Measurements associated with a registration step are modeled not only by their means, but also their covariances. Methods for combining serial measurements (e.g., as in the kinematic model of a robot) as well as parallel measurements (e.g., measuring the position of an object using two different robots) are presented.

The registration graph and metric uncertainty model are combined in Chapter 4. Here, the registration graph and necessary logic and algorithms are embodied as a class library written in C++. An interactive program developed for the Microsoft Windows® 95/98 platform is presented. The program uses the class library and allows a user to interactively build and explore registration graphs.

A case study that highlights the usefulness of the registration graph is presented in Chapter 5. Finally, concluding remarks and future work are discussed in Chapter 6. To assist those that may wish to use the registration grapher program, a user's guide is found in the appendix.

2 Qualitative Modeling with Registration Graphs

The Northwestern total knee replacement system will be used to introduce the registration graph. The system was briefly introduced in Chapter 1 and contains many similarities to the RoboDoc system discussed therein. An overview that will enable the introduction of the registration graph follows, while a more detailed description can be found in [1][2].

Preoperatively, titanium landmark screws are inserted into the patient's femur and tibia, and a CT scan of the knee joint is obtained and is loaded into a graphic computer workstation. The desired position of the knee implants is prescribed by manipulating 3D models of the implants on a CT-derived model of the knee joint. Positions of the landmark screws and the resections required for the implants are saved.

Intraoperatively, the knee joint is immobilized using specialized bone clamps. After measuring the intraoperative locations of the landmark screws thereby registering the position of the preoperative plan to the intraoperative positions of the bones, the robot moves a cutting guide into place. The required resections are made with a conventional reciprocating saw and the implants are fitted on the bones.

Figure 2.1 shows the registration graph of the Northwestern TKR system.

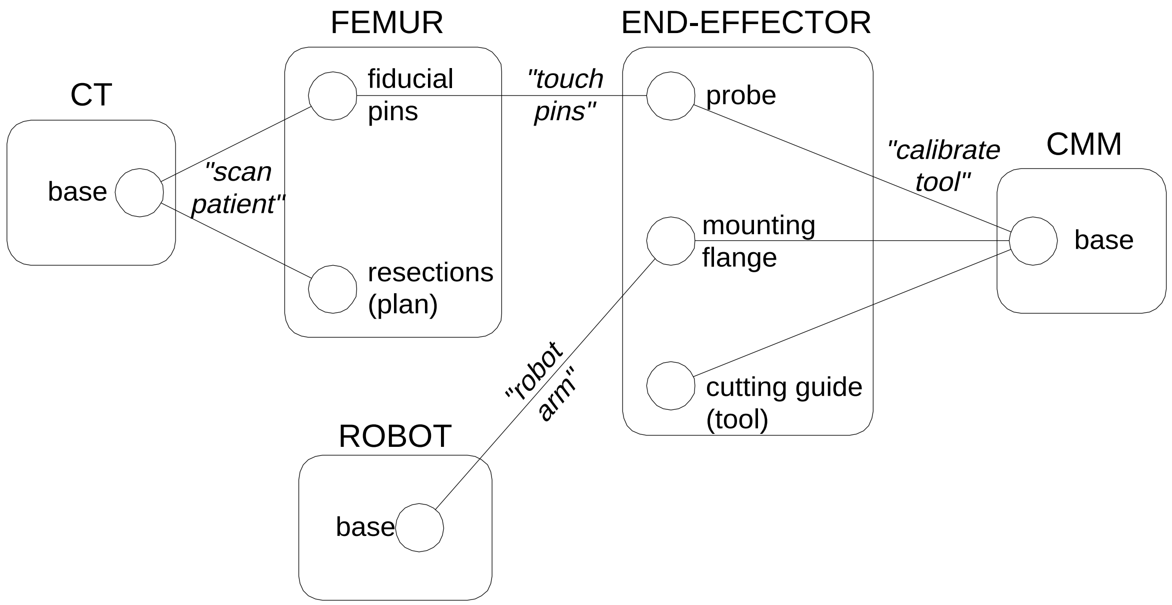


Figure 2.1: Features and links of the registration graph for the Northwestern TKR system.

2.1 Features and Links

A registration graph consists of features interconnected by links. The terms vertices and edges are more standard in graph theory [10], but the geometric connotations of these terms make them confusing in this context.

Features, shown as small circles, represent objects. Examples in Figure 2.1 include the robot’s base, fiducial pins, the CT scanner base, the probe on the robot's end effector (the “hand” of the robot), the coordinate measuring machine (CMM), and the cutting guide. The surgical plan of intended resection positions is of course not a physical object, but it turns out to connect to other objects in a manner similar to that of a physical object, so it is also shown as a feature.

An unusual and key characteristic of registration graphs is that measurement devices, and the objects whose positions they measure, enter into the graph in identical fashion; all are simply features. (Optionally, a feature may be thought of as specifically representing as a coordinate frame associated with an object, as in Paul [11]. This is especially useful when interpreting features such as the [ROBOT : base] in Figure 2.1, and such thinking will serve us well Chapter 3.)

A *rigid body* group, indicated by a rounded rectangle, visually collects features that will never move relative to one another. The set of features can be thought of as parts of a single rigid body. For instance, the probe and cutting guide of the end effector are actually two different parts, but can be thought of as a single rigid body because the probe is tightly affixed to the body of the cutting guide. The registration graph rules and logic developed below do not require the use of rigid body groups, and thus rigid body groups are not a necessity. However, clear explanation of a registration strategy through it's graph is one of our primary goals, and rigid body groups serve the purpose of hierarchically arranging the features so the graph can be easily interpreted.

Links, shown as lines, connect two features. They represent an *act of measurement*, which establishes a known spatial relationship between the objects represented by the two features. While it might seem appropriate to show links as arrows representing in what direction the measurement is being made, a registration "digraph" unnecessarily complicates the picture and wrongly infers that the only way to move from feature to feature is in the direction of an arrow.

One usually thinks of a measuring device dispassionately measuring the spatial relationship between two other objects, thus establishing a connection between them. A notation consistent with this philosophy would show a measuring device solely as a link, connecting the objects it measures. Here, however, a measuring device is represented as a feature, and as a consequence it is more intimately involved: it reaches out to measure (through links) the relationship of its own coordinate frame to that of another object.

Other graph conventions might seem more natural, and could equally well describe the Northwestern TKR registration strategy. Some candidates explored while looking for the appropriate representation for registration include graph-theoretic techniques of data flow diagrams in structured analysis [12] and Petri nets [13]. The one described here is the only convention I have found that extends successfully to arbitrarily complex registration strategies.

The registration graph in Figure 2.1 shows all of the measurement operations, and all of the objects that are involved in registration in the Northwestern TKR system:

- A coordinate measuring machine (CMM) measures its own relation to three parts of the end-effector: the probe, the mounting flange, and the cutting guide.
- A CT scanner measures its own relation to the fiducial pins. A "plan" feature is shown as well, with a link to the CT like that of the fiducial pins.

- The robot measures its own relation to the end effector flange. It can also command that relation, but that capability is not relevant to us yet.
- All of the links listed above connect one feature that is capable of measuring, to one that is not. One link was not listed: the one that connects probe to fiducial pins. Neither a probe nor a fiducial pin is normally thought of as a measuring device. Yet they do have a very limited measuring capability; they can measure *coincidence*. Bringing two objects into physical contact can be thought of as a measurement operation.

One can now observe the series of links that connects plan to tool. It is just such a connected subgraph that validates registration. However, the connected subgraph in Figure 2.1 does not reflect several of the important aspects of registration. First, the links are valid at different times. For instance, the validity of the [END-EFFECTOR : probe → FEMUR : fiducial pins] link is broken by moving the end effector to orient the cutting guide, so registration would seem to be lost at that point. Second, the robot appears to be uninvolved. In practice the robot is the central device, and a primary error source. These problems are resolved by the introduction of events and induced links.

2.2 Events Tags and Induced Links

When objects (features) move relative to one another, measurement operations (links) that were previously performed become invalid. To group links that are simultaneously valid we introduce *events*. A link's validity during one or more events is indicated by one or more *event tags* on the link. The event tag can be interpreted as a function on the link setting the link's "resistance" to zero when the link is valid and to infinity when not. As a matter of convention, we will use (1) as the earliest possible event.

Many links are permanently valid once established, notably ones established between two objects on a single rigid body. I will present a construct that accounts for the length of validity momentarily. For now, I will simply mark these links with all appropriate event tags. For instance, the first event (chronologically) includes measurements of the relation between the CMM and the three components of the end-effector. These links are marked with event tag (1) in Figure 2.2. However, the [ROBOT : base → END-EFFECTOR : mounting flange] link is always known once the robot is powered up for use in the operating room, and is thus marked with tags (3,4).

The links to the CMM become invalid when the end-effector is removed from the test bay of the CMM. However, the spatial relationship(s) of the components to each other, which can be derived from the links to the CMM, persists. A new construct, the *induced link*, captures this persistence. The links we have shown up to now will be known as *direct* links.

An induced link, shown as a dashed line connecting two features, is valid if there is a *simultaneous connected subgraph* that connects the two features. A simultaneous connected subgraph is simply a subset of the features and links, such that all of the features are connected to one another, directly or indirectly, via links that belong to a common event. To prevent excessive proliferation of valid subgraphs for an induced link, other induced links with the same event are not used to determine validity.

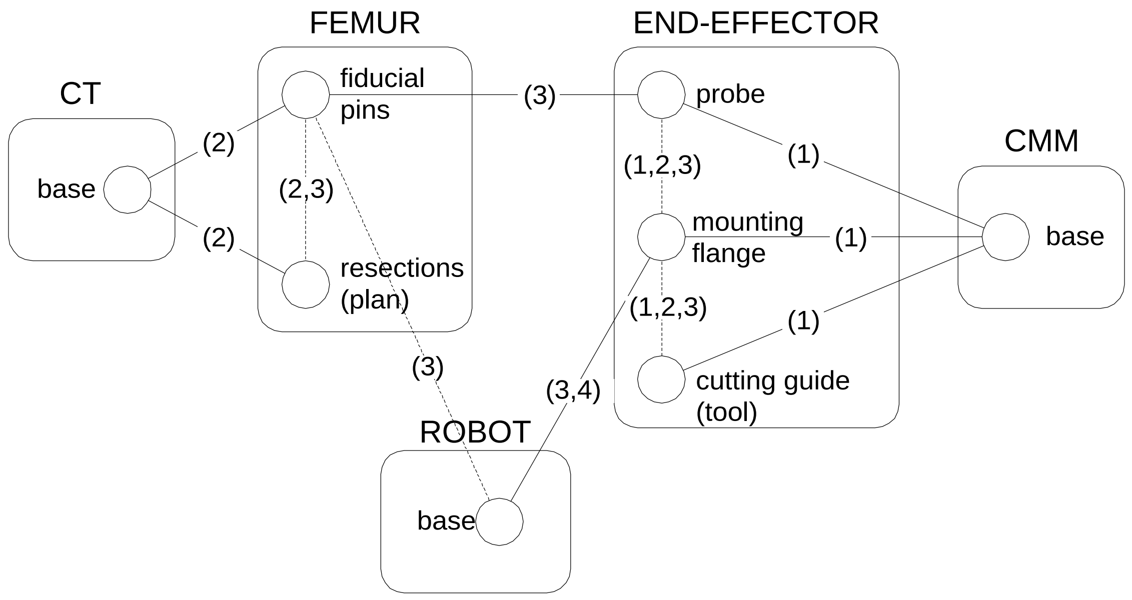


Figure 2.2: Registration graph for the Northwestern TKR system, showing induced links and event tags.

Figure 2.2 shows the Northwestern TKR system again, now including the induced links. Induced links indicate indirect knowledge of the spatial relationship between two objects, derived via the simultaneous connected subgraph that connects them. The validity of the indirect knowledge may greatly outlast that of the subgraph, which induced it. Most of the induced links connect pairs of objects that belong to a single rigid body. No relative motion ever occurs between such pairs, so their validity is permanent.

However, consider the induced link [ROBOT : base → FEMUR : fiducial pins]. The simultaneous connected subgraph that induced it is [ROBOT : base → END-EFFECTOR : mounting flange → END-EFFECTOR : probe → FEMUR : fiducial pins], all of which are members of event (3). The induced link [ROBOT : base → FEMUR : fiducial pins] is valid so long as the femur does not move relative to the robot. This is reflected in the hardware of the Northwestern TKR system; the femur is rigidly fixtured with respect to the robot, from the time the fiducial pins are touched until the computer-assisted phase of surgery is complete.

2.3 Connectivity Properties

We can now make some observations concerning the connectivity of a registration graph. First, the graphs provide a basic condition for registration:

> *Basic connectivity*: Registration requires that there exist a simultaneous connected subgraph containing the plan and the tool. In Figure 2.2, the subgraph is [FEMUR : resections (plan) → FEMUR : fiducial pins → ROBOT : base → END-EFFECTOR : mounting flange → END-EFFECTOR : cutting guide (tool)].

Not all computer-assisted surgery systems contain an active positioning device. For those that do (in the present example, a robot) a further connectivity property is needed, in order to be able to compute the actuator velocities needed to move the tool. This capability might be called control.

> *Control connectivity*: Control requires that the robot (or other active positioning device) must also be a part of the simultaneous connected subgraph.

If a graph has basic connectivity but not control connectivity, it is still possible to determine the actual tool position with respect to the plan. However it is not possible to predict what effect a particular commanded motion of the positioning device will have on the tool's location with respect to the plan.

2.4 Maintaining registration

It is not enough to establish registration at a single moment; we must continuously maintain registration while the robot moves the tool. It may also be possible to maintain registration while the patient moves during surgery. If so, the patient does not need to be rigidly immobilized, an attractive feature for a surgical system. Both issues can be addressed through the connectivity properties of registration graphs.

We make a distinction between two kinds of events: transient and sustained.

A *transient* event describing a link's validity, shown by appending the numeral of the event tag with a "t", indicates a one-shot measurement operation. Examples: a bone contour is identified in a CT image; a probe touches a fiducial marker or bone contour; a single fluoroscopic image of an alignment artifact is acquired.

A *sustained* event describing a link's validity, shown by appending the numeral of the event tag with an "s", indicates an ongoing measurement operation performed by one of the two features. Also, when two objects are brought into coincidence and locked, the link is considered to be sustained. Examples: an optical localizer monitors the location of an LED "flag" on a moving bone; a tool is held by a robot that possesses joint encoders; clips on a stereotactic headframe are brought into contact with mounting brackets and fastened.

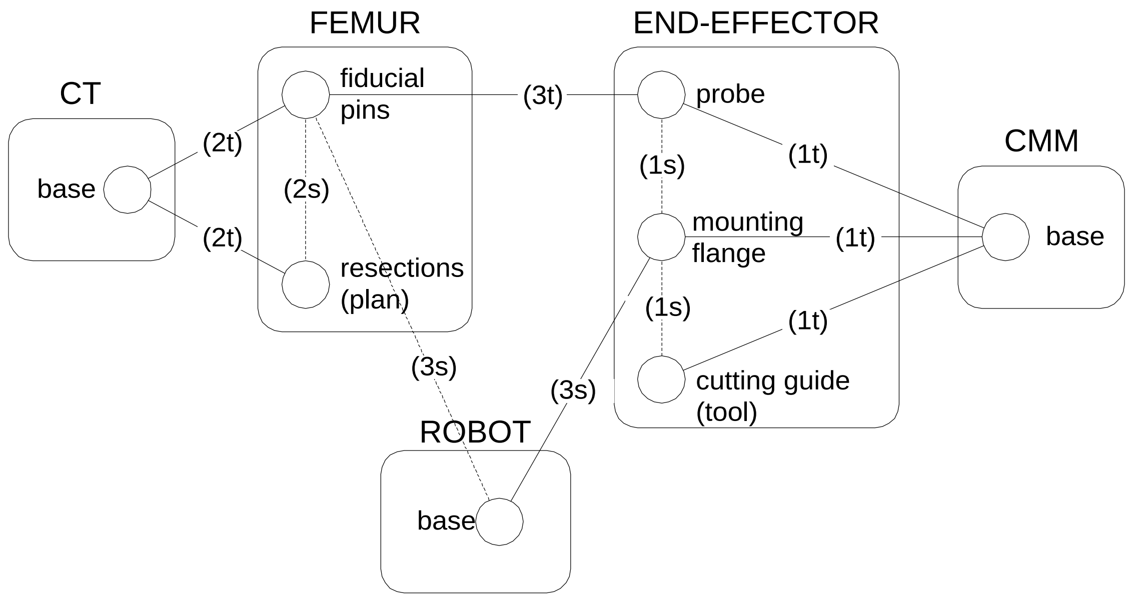


Figure 2.3: Registration graph for the Northwestern TKR system, with transient and sustained event tags marked with "t" and "s" respectively.

Figure 2.3 shows the registration graph for the Northwestern TKR system with the appropriate event tags to reflect their links' transient or sustained character.

I noted above that the induced link [ROBOT : base → FEMUR : fiducial pins] in the Northwestern TKR system required a femur fixator to maintain it. More generally, we can make the following observation:

> *Fixation*: Induced links require immobilization to remain valid once the subgraph(s) that enabled the link is expired.

Sometimes the required immobilization is trivially present because the induced link connects objects, which are both parts of a single rigid body, e.g. the end-effector. Other induced links require the explicit use of fixation devices. For instance, the [ROBOT : base → FEMUR : fiducial pins] induced link requires a femur fixator in the Northwestern TKR system.

Where an explicit fixation device is needed, it may couple the features directly, or it may couple any features of the rigid bodies to which they belong. Recall that rigid body groups are merely an organizational tool; they do not affect the connectivity properties of the graphs. However, they make the need for immobilization devices especially visible by emphasizing induced links that are not contained within a rigid body group.

We can now add a third connectivity property:

> *Sustained connectivity*: Continuous registration despite tool motion and patient motion requires that the simultaneous connected subgraph consist entirely of sustained and induced links.

Sustained connectivity does not necessarily imply that a non-fixated patient during surgery is possible. Immobilization requirements created by induced links between the anatomy rigid body group and other rigid body groups still pertain.

2.5 Graph Reduction and Expansion

One of the benefits of the registration graph is that it makes explicit the entire chain of measurements that comprise (and contribute error to) registration. However explicitness sometimes comes at the expense of clarity and conciseness. We may prefer to represent an end-effector as a single feature, as in Figure 2.4, rather than as a cutting guide, probe, and

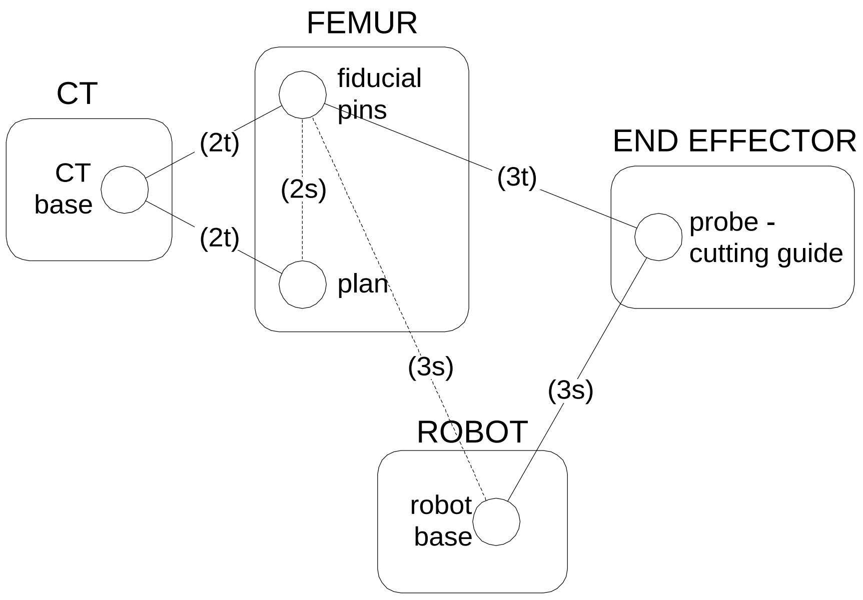


Figure 2.4: The features of the end effector rigid body group have been reduced to a single feature using the reduction rule.

mounting flange whose relationship has been established by a CMM, as in Figure 2.3. A reduction rule makes clear the conditions under which such collapse can be performed without affecting any connectivity properties:

> *Reduction rule*: Two features connected by a link with a sustained event tag may be collapsed into one feature.

The converse is true, so we also may think of the reduction rule as an expansion rule. Although we have represented the robot as a single link between the robot base and mounting flange, we could instead show the robot as seven serially connected features representing each linkage of the robot.

The reduction rule should be used sparingly to remove uninteresting clutter, or to help determine how new components should be represented. A good rule of thumb is not to collapse features unless the combined feature can be given an enlightening name. For instance, seven robot features can be collapsed to a single feature named "robot", and the several components of an end-effector can be collapsed to a single feature named "end-effector". It is usually not a good idea to collapse a robot and its end-effector to a single feature. Also, collapsing a robot and a fluoroscope can only result in a "fluorobot", and is a poor idea.

It should be noted that the registration graph should be extendible to the problem of non-rigid registration by means of scale, that is, a graph can be expanded to a finite element level of detail. The error analysis of Chapter 3 should also extend well to such detail. This is not attempted herein, as it is beyond the scope of this thesis.

2.6 A Short Survey of Registration Strategies

In the following sections, I graph other computer-assisted surgery systems, selected to illustrate a variety of interesting aspects of registration strategies. The graphs are detailed to the extent I can determine each system's registration procedures from the literature or other communication. In cases where the device by which a measurement is made is unknown, I have simply presumed that the measurement is established by CMM.

2.6.1 Other systems like the Northwestern TKR system

Figure 2.3 showed the registration graph of the Northwestern TKR system. Several other computer-assisted surgery systems have virtually identical registration strategies to that of the

Northwestern TKR system. These include the RoboDoc system described in Chapter 1 [3][4], the most recent system for total knee replacement by Fadda et al. [14] at Rizzoli Institutes, and a system for femoral osteotomies by Moctezuma, et al. [15] at the Institute for Machine Tools and Industrial Management in Münich.

2.6.2 StealthStation System

A notable aspect of this system, described in Chapter 1, is that registration is maintained despite patient motion during surgery, and therefore no fixation of the involved bone is required. Construction of a registration graph makes this apparent. Referring to Figure 2.5,

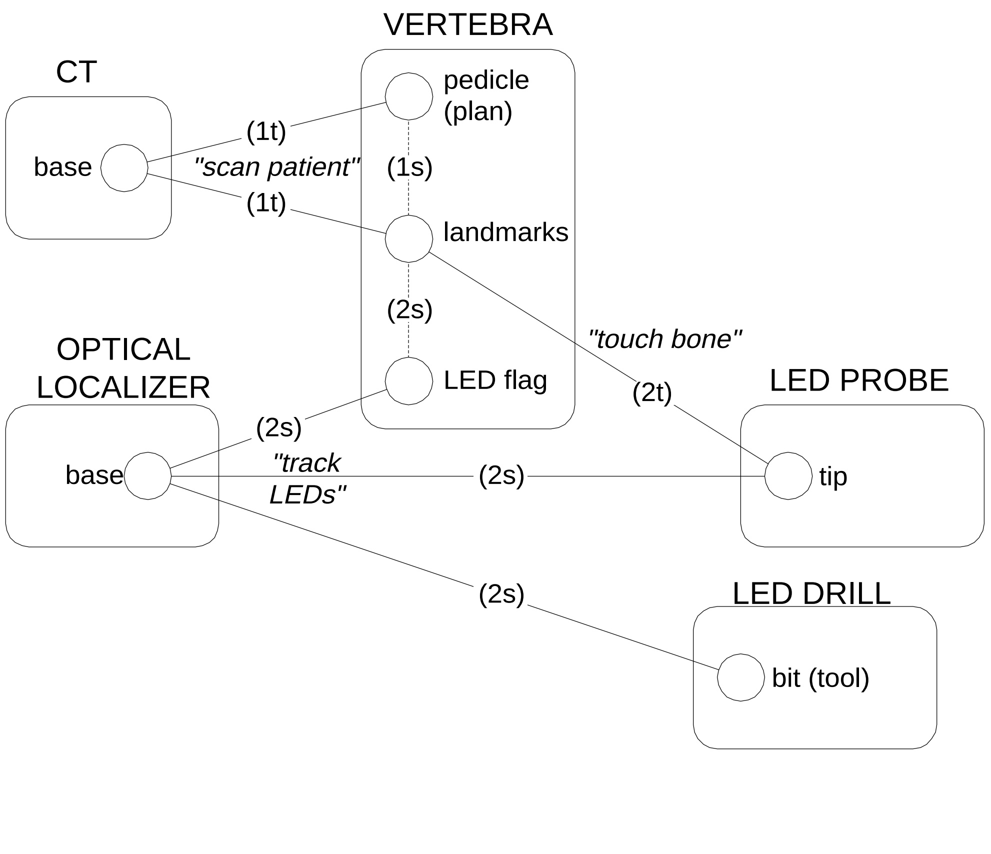


Figure 2.5: Registration graph of the StealthStation as used for pedicle screw placement.

in which the StealthStation is used to drill a pilot hole down the shaft of the pedicle of a vertebra, we find that

- A CT scan is obtained preoperatively. Intraoperatively, a computer displays the CT data as well as a 3D reconstruction of the vertebra derived from the CT data. The desired drill trajectory through the pedicle (the plan) can be implicitly viewed in the CT data. An induced link is thus formed between the plan and other parts of the vertebra noted as landmarks.
- Intraoperatively, a 'flag', instrumented with infrared LEDs, is attached to the spinous process of the vertebra. An optical localizer tracks the motion of this flag.
- A pointing probe, outfitted with LEDs, is used to digitize various landmark points from the surface of the vertebra. While the probe contacts the surface of the vertebra, a [LED PROBE : tip → VERTEBRA : landmark] transient link exists. This creates a [VERTEBRA : flag → VERTEBRA : landmark] induced link. (It is worth noting that a 3D-3D matching algorithm is typically used to correlate the digitized points from the optical localizer to the CT image of the vertebra. The difficulty of this matching operation is not acknowledged by the registration graph.)
- The optical localizer tracks a hand-held surgical drill, outfitted with LEDs, while the surgeon orients the drill bit to the desired trajectory. The trajectory that the drill bit makes through the pedicle is displayed on the computer screen showing the CT images. (note: the LED drill is not a part of the commercial StealthStation. However, a drill with LEDs can be and often is built and used by the operators of the system.)

The subgraph [VERTEBRA : pedicle (plan) → VERTEBRA : landmarks → VERTEBRA : LED flag → OPTICAL LOCALIZER : base → LED DRILL : bit (tool)] which establishes sustained connectivity does not include any induced links between rigid body groups (which would signal a need for fixturing.) Thus the graph shows that the vertebra need not be fixtured.

The graph does not have control connectivity, because there is no active positioning device. It would be possible to include the "computer-controlled surgeon" used here into the registration graph, but I have not done so here.

Several systems bear similar registration graphs to that of the StealthStation. Lavallée et al. [16] at Grenoble University Hospital in La Tronche, France have developed a computer-assisted surgery system for implanting pedicle screws that uses an explicit preoperative plan to prescribe screw placement. Nolte, et al., [17] at the University of Bern in Switzerland have developed a system for pedicle screw placement with an identical registration graph to that of Lavallée et al.

2.6.3 Long Beach Stereotactic Neurosurgery System

The robotic stereotactic neurosurgery system used by Kwoh, et al. [18] at the Memorial Medical Center of Long Beach was one of the first computer-assisted surgery systems developed. The registration graph of this system is typical of systems that use stereotactic frames, such as Glauser et al. [19], Drake et al., [20]. Referring to Figure 2.6,

- A CT image is obtained of the patient's head, showing also the fiducial markers on a removable headframe that the patient wears. A plan (a syringe trajectory that reaches the biopsy site in the brain safely) is created in the data from the CT, and thus an induced link is shown from fiducial markers to plan. Recall that induced links require immobilization, which is provided here by fixating the headframe to the skull.

- Later, mounting clips on the headframe are locked to brackets attached to the robot base. This constitutes a sustained "coincidence" link.
- The induced links [HEADFRAME : fiducial markers → HEADFRAME : mounting clips] and [ROBOT : base → ROBOT : mounting bracket], were apparently established earlier by direct measurement (here assumed to be performed by CMM) , although one could imagine doing it differently by touching the a probe mounted on the end effector to the features (which would result in a different registration graph, of course.)

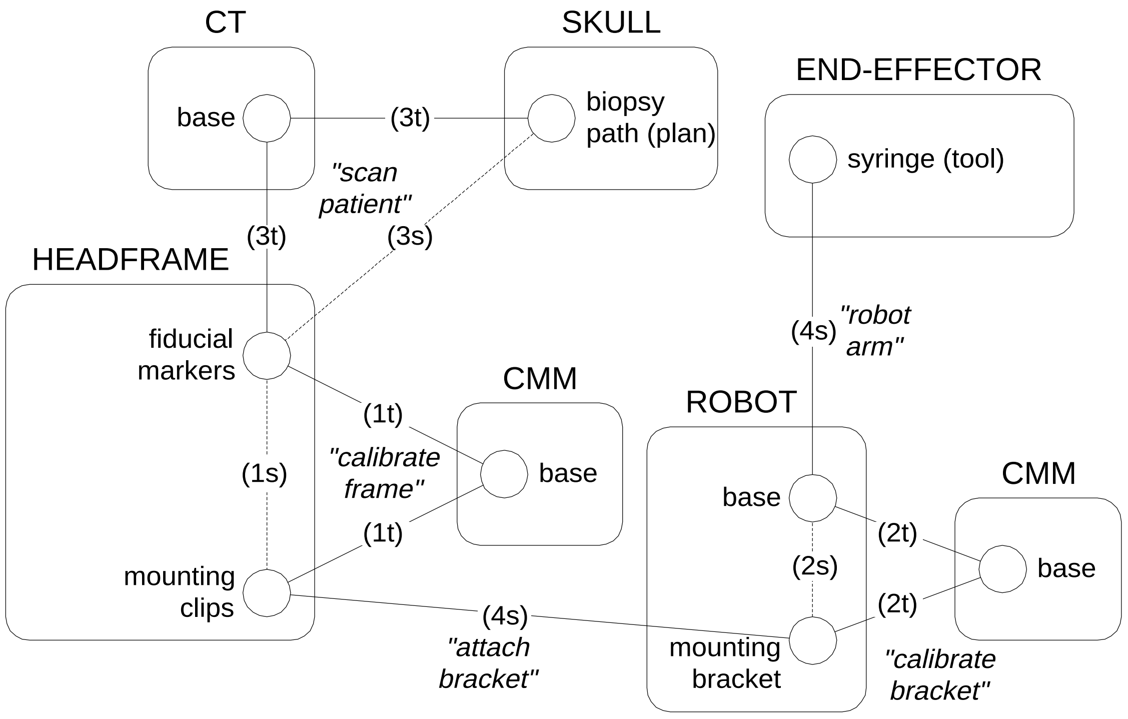


Figure 2.6: Registration graph of the Long Beach stereotactic neurosurgery system.

The subgraph [SKULL : biopsy path (plan) → HEADFRAME : fiducial markers → HEADFRAME : mounting clips → ROBOT : mounting bracket → ROBOT : base → END-EFFECTOR : syringe (tool)] establishes sustained connectivity, but involves two fixtures: one to maintain the induced link from headframe to patient's skull, and another corresponding to the sustained "coincidence" link between the mounting clips and the mounting bracket.

2.6.4 Northwestern Pedicle Screw Placement System

Santos-Munné [21][22] at Northwestern University describes a system for pedicle screw placement. The design philosophy is to stay as close to conventional practice as possible,

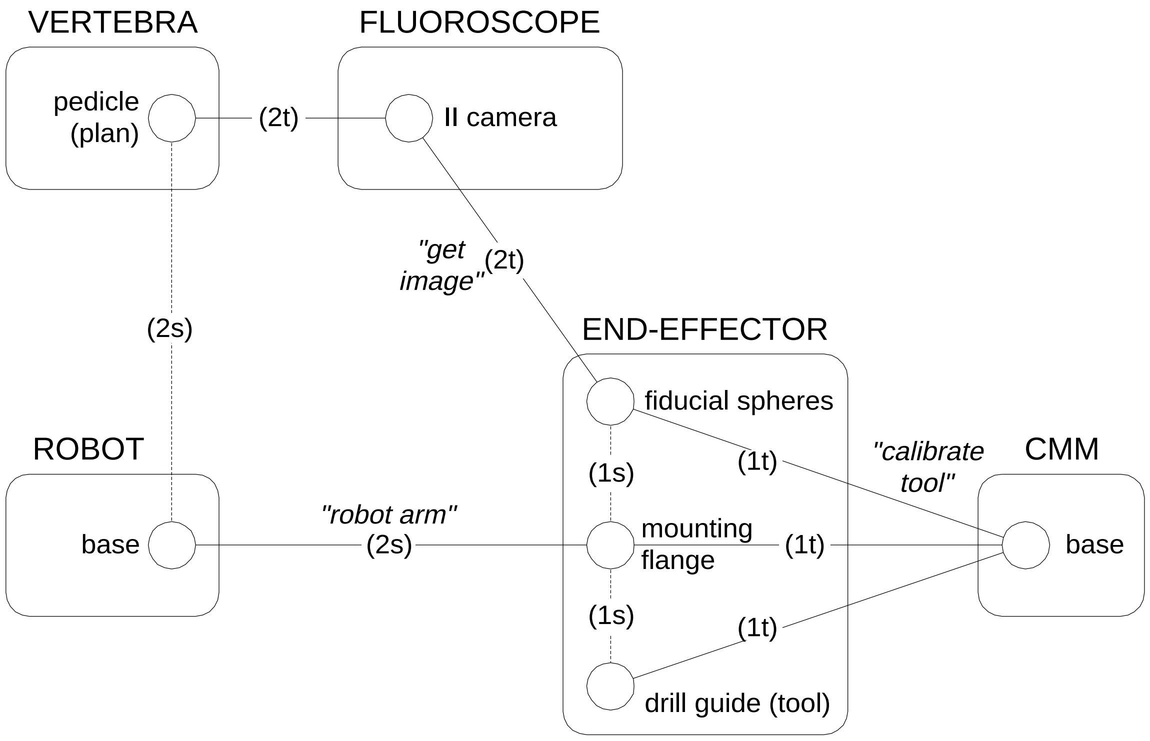


Figure 2.7: Registration graph of the Northwestern pedicle screw placement system.

while achieving greater accuracy through the use of a robot for drill guidance. As in conventional practice, planning is done in two parts, corresponding to two orthographic projections. Referring to Figure 2.7,

- The robot's end effector consists of a drill guide, an array of fiducial spheres, and a mounting flange. Induced links between the components were created by CMM.
- Intraoperatively an anterior-posterior (AP) fluoroscopic image of the vertebra and the array of fiducial spheres on the end effector, which is held close to the patient, is obtained. The image is used to create the AP projection of the plan. A sagittal fluoroscopic image is used similarly.

An induced link is established between the vertebra and the robot, via the connected subgraph [VERTEBRA : pedicle (plan) → FLUOROSCOPE : II camera → END-EFFECTOR : fiducial spheres → END-EFFECTOR : mounting flange → ROBOT : base].

The subgraph [VERTEBRA : pedicle (plan) → ROBOT : base → END-EFFECTOR : mounting flange → END-EFFECTOR : drill guide (tool)] establishes sustained connectivity. However, the [VERTEBRA : pedicle (plan) → ROBOT : base] induced link requires immobilization.

2.6.5 Grenoble Pedicle Screw Placement System

An earlier system for pedicle screw placement developed at Grenoble than that of [16] is described by Troccaz et al. [23], and uses much of the same instrumentation as the Northwestern pedicle screw placement system. However, the Northwestern and Grenoble systems do not share the same registration strategy, and therefore the registration graphs are different. Referring to Figure 2.8,

- Drill trajectory planning is done on a 3D reconstruction of the vertebra obtained from CT. Therefore, an induced link exists between plan and the vertebral bone surface.
- A reference grid held by the robot is imaged by the fluoroscope, and the subgraph [FLUOROSCOPE : II camera → REFERENCE GRID : fiducial holes → REFERENCE GRID :

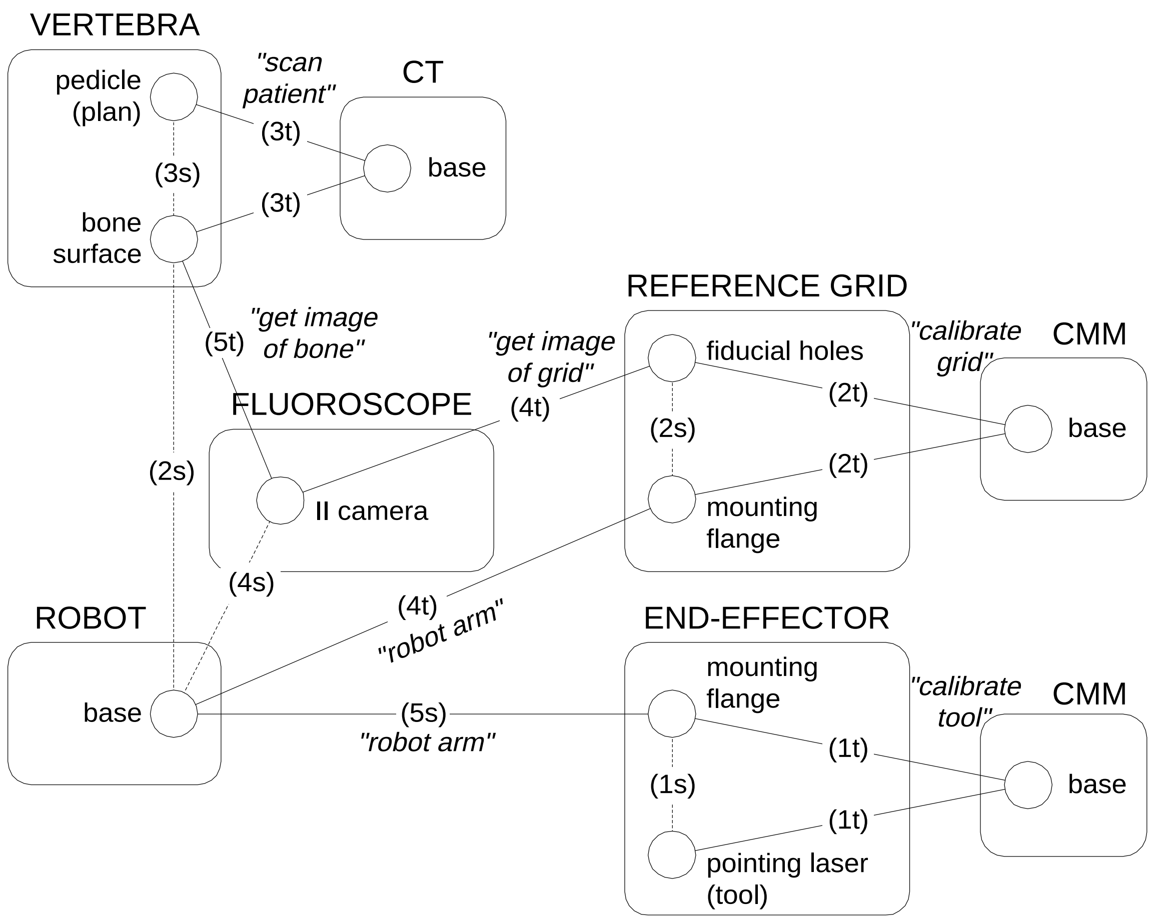


Figure 2.8: Registration graph of the Grenoble pedicle screw placement system that employs a fluoroscope intraoperatively.

mounting flange → ROBOT : base] establishes an induced link [FLUOROSCOPE : II camera → ROBOT : base]. This induced link must be maintained by fixturing robot to the fluoroscope.

- A fluoroscopic image of the vertebra is acquired. Together with the [FLUOROSCOPE : II camera → ROBOT : base] induced link, a connected subgraph is created which induces a [VERTEBRA : bone surface → ROBOT : base] link.

The reference grid is removed from the robot and replaced with a pointing laser. The surgeon uses the laser beam as a guide for drilling.

The subgraph [VERTEBRA : pedicle (plan) → VERTEBRA : bone surface → ROBOT : base → END-EFFECTOR : mounting flange → END-EFFECTOR : laser (tool)] establishes sustained connectivity. The [VERTEBRA : bone surface → ROBOT : base] induced link requires fixation during surgery. The [FLUOROSCOPE : II camera → ROBOT : base] induced link is not involved in the subgraph. It was only used to establish the [VERTEBRA : bone surface → ROBOT : base] link, and may be broken thereafter.

A like registration strategy has been used in other systems developed by the Grenoble group, notably a system for stereotactic neurosurgery by Lavallée et al. [24].

2.6.6 Imperial College Keyhole Surgery

A system for percutaneous biopsies (of, for example, the kidney) has been developed by Potamianos, et al. [25] at Imperial College in London. This system displays a computer-generated image of a biopsy needle superimposed on static fluoroscopic images of the

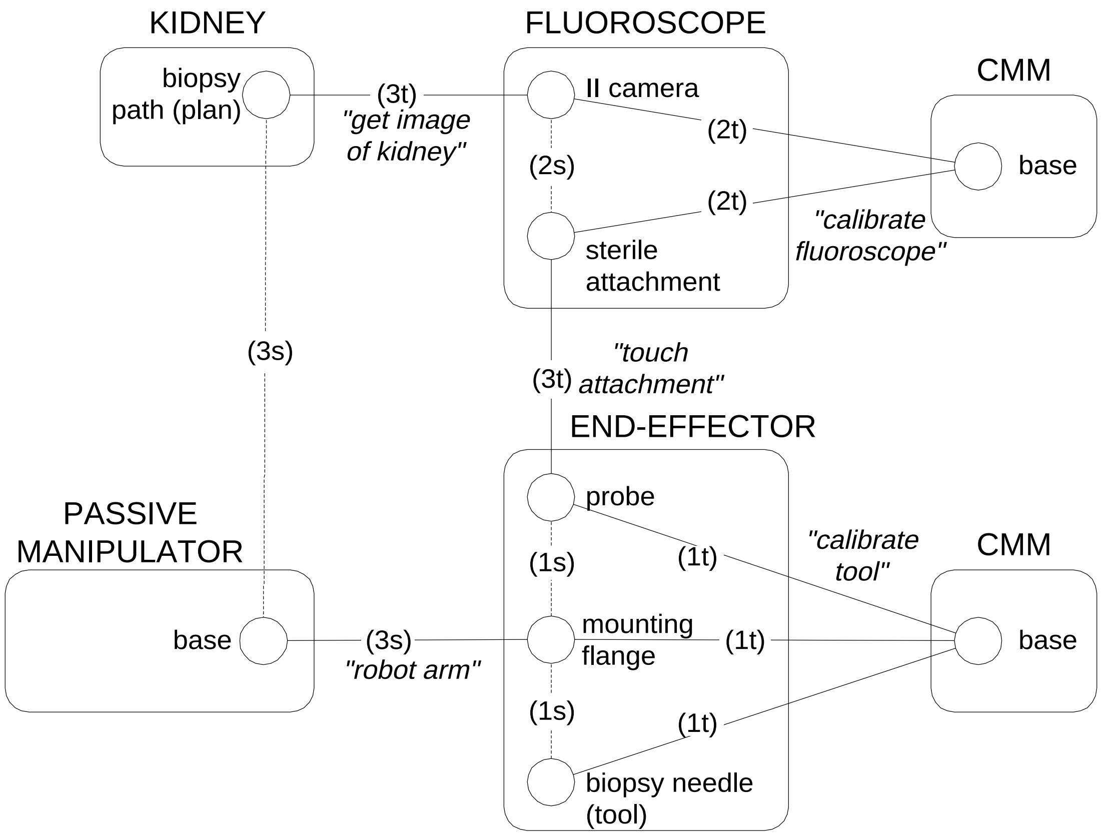


Figure 2.9: Registration graph of the Imperial College keyhole surgery system.

patient as the real needle is manually moved about. The registration graph, shown in Figure 2.9, is notable in that there is no preoperative plan.

- A fluoroscope image of the biopsy site is acquired.
- A probe on the passive manipulator's end effector is placed into contact with a sterile attachment on the fluoroscope. This induces a link between the manipulator and the biopsy site.

The subgraph [KIDNEY : biopsy site → PASSIVE MANIPULATOR : base → END-EFFECTOR : mounting-flange → END-EFFECTOR : needle] establishes sustained connectivity. The [PASSIVE MANIPULATOR : base → KIDNEY : biopsy site] induced link must be maintained by immobilizing the patient.

2.7 Summary

The registration graphs shown in chapter illustrate how connectivity properties of registration graphs are useful in the study of registration pathways and immobilization requirements of computer-assisted surgery systems. In the next chapter, we will derive the necessary theory to add a quantitative measure of analysis to the registration graph.

3 Quantitative Modeling using Uncertainty Analysis

The connectivity rules of the previous chapter provide simple “go” or “no go” feedback as to whether or not registration has been achieved. The next useful step in modeling registration strategies is to implement a quantitative metric as to how well the registration has been performed. That is, given that connectivity of a simultaneously connected subgraph is established, how certain are we that the combination of measurements along the subgraph reasonably represent the true relative position of the first feature in the subgraph to the last feature in the subgraph?

In this chapter, the theory behind a useful method for answering this question is presented. The spatial measurements are modeled by their first and second statistical moments, means and covariances. It is shown how to “add up” the means and covariances of a simultaneously connected subgraph that provides a path for registration. The “addition” takes into account both the common serial compounding of measurements (for combining links connected in a subgraph) as well as parallel compounding (for combining multiple subgraphs connecting two features). Finally, a method for visualizing the hyper-dimensional error ellipsoid that is described by a measurement’s covariances is shown.

3.1 Representing Position

The measurements that provide position information about one feature with respect to another are obviously of primary importance in registration. Through series of such measurements (achieved using measuring devices common to computer-assisted surgery: robots, optical localizers, CT scanners, fluoroscopes, etc.) it is possible to "learn" the relative spatial relation that exists between disparate objects. The registration graph presented in the previous chapter provides us with the link construct to represent the act of this measurement. It is now necessary to define a representation for this measurement.

The 3D pose of a feature with respect to another is described by a transformation that is a function of the six-degree of freedoms (DOF) that exist between two coordinate frames:

$$\mathbf{T} = f(x, y, z, \phi, \theta, \psi) \tag{3.1}$$

The first three components of the parameter vector, *x*, *y*, *z*, specify the position between the origins of the features' coordinate frames, and the second three components represent the rotations necessary to describe the relative orientation between the frames. I will use the roll, pitch, and yaw (RPY) sequence of rotations to describe the orientation, where roll is a rotation ϕ about the *z*-axis, pitch is a rotation θ about the *y*-axis, and yaw is a rotation ψ about the *x*-axis.

The functional representation of the transformation is a 4x4 matrix whose upper left 3x3 sub-matrix represents the rotations necessary to orient the first feature's coordinate frame with the second's frame, and the upper right 3x1 vector represents the translations needed to match the first frame's origin with the second's origin.

$$\mathbf{T} = \begin{bmatrix} & \mathbf{R}_{3x3} & & \mathbf{t}_{3x1} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.2}$$

The bottom row contains a series of 0's and a 1, making this a *homogeneous* transformation (the homogeneous representation provides many benefits such as simple matrix multiplication to "add" transformations and the ability to include scaling and perspective effects. See [11] for more detail.) The vector **t** is simply the three translational components

$$\mathbf{t} = [x \quad y \quad z]^{tr} \tag{3.3}$$

where ^{tr} denotes the vector transpose. The rotation matrix for RPY representation is

$$\mathbf{R} = \begin{bmatrix} \cos\phi\cos\theta & \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi & \cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi \\ sin\phi\cos\theta & \sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi & \sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi \\ -\sin\theta & \cos\theta\sin\psi & \cos\theta\cos\psi \end{bmatrix} \tag{3.4}$$

It is often necessary to extract the parameter vector from the 4x4 homogeneous matrix. The components of the translational vector **t** are simply copied into the first three components of the parameter vector. The rotational parameters can found from

$$\phi = \text{atan2}(R_{21}, R_{11}) \tag{3.5}$$

$$\phi = \phi + 180° \tag{3.6}$$

$$\theta = \text{atan2}(-R_{31}, \cos\phi R_{11} + \sin\phi R_{21}) \tag{3.7}$$

$$\psi = \text{atan2}(\sin\phi R_{13} - \cos\phi R_{23}, -\sin\phi R_{12} + \cos\phi R_{22}) \tag{3.8}$$

3.2 Representing Uncertainty

A second metric that is useful is how *well* the measurements represent the actual relative positions of the features. This is a significant problem in registration, and has been receiving deserved attention recently. In [26][27], Simon presents techniques for recommending points to be digitized for a 3D-3D point-to-surface matching algorithm (e.g. "picking" points on a bone) that use a measure of the "stiffness" of resultant registration based on the uncertainty present in digitizing the points. Ellis [28] compares the errors of registrations between data sets achieved by finding fiducial markers on the femur and tibia using Roentgen Stereogrammetric Analysis, CT scans, and a pointer mounted to a passive metrology arm. Nolte [17] provides some measure of the registration error in a system for spine surgery, and Grimson [29] discuss errors observed in superimposing MRI data onto the video image of a patient in a system for "enhanced reality" visualization. Simon also has done some investigation into overall accuracy in an image-guided system for orthopedics in which he proposes a screw axis representation of implant placement error [30]. Glossop suggests a two-parameter error metric for the placement of pedicle screws [31].

Most of the aforementioned works unfortunately (for us) do not model all errors present in a computer-assisted surgery system, but only that of error in the last defining step of registration. This is due in part to the fact that the authors are interested to see how well certain algorithms (e.g., 3D matching) are behaving. Additionally, the error metrics proposed in [30][31] do not work well with our 6-vector representation of relative positions between features. To implement an uncertainty metric in the registration graph, we must look elsewhere.

An uncertainty measure that is compatible our chosen representation for position would include errors for each of the six variables. There are two dominant methods for describing uncertainty in this manner: stochastic measures, such a covariances, and geometric measures.

3.2.1 Geometric Bounds

Taylor chose an inequality representation for the expected errors on each variable in his doctoral thesis [32] and uses this in subsequent work [33]. The maximum errors are typically derived from historical worst-case evidence of, say, a robot linkages actual position versus its commanded position, or geometric constraints in the workspace. For instance, positions and orientations of a 10 mm cube placed nominally in the center of a 11 mm cubic hole (see Figure 3.1) by a robot would maximally be bounded by

$$-0.5 \le x \le 0.5 \tag{3.9}$$

$$-0.5 \le y \le 0.5 \tag{3.10}$$

$$-6^{\circ} \le \theta \le 6^{\circ} \tag{3.11}$$

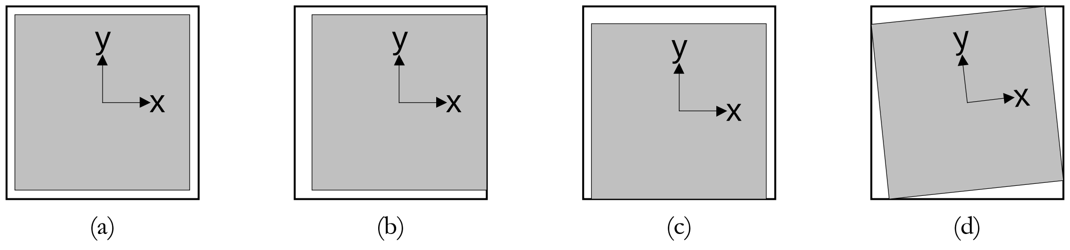


Figure 3.1: Positioning of a 10 mm cube in a 11 mm cubic hole. (a) nominal position (centered in hole), (b) Δx = 0.5 mm, (c) Δy = -0.5 mm, (d) $\Delta\theta$ = 6°.

The errors may be less if the robot's inherent accuracy is greater than that of the geometric constraints created by the hole. The set of inequalities describes a polytope, a geometric volume bounded by intersecting planes. The polytope is the uncertainty volume.

Visualization of the polytope is accomplished by projecting its "shadow" onto a two-dimensional plane (e.g. the *x-y* plane), resulting in a polygon describing the errors in the two variables associated with the plane. The projection algorithm is highly optimized since projection of high-order polytopes, which result from serial compounding of measurements (as in a robot), are extremely complex and brute force algorithms for projection are slow.

Brooks develops a symbolic method for computation of the polytope so that questions other that "What is the resultant accuracy of positioning by this robot?" can be asked [34][35]. One such interesting question is "Given the desired resultant accuracy, how accurate must this robot be?"

Geometric bounds on uncertainty are certainly robust in that they can be used to represent a worst case error, and thus would seem well suited to the task of error analysis in computer-assisted surgery. However, geometric bounds are not commonly specified in product literature for robots, optical localizers, etc., and are not familiar to most of the computer-assisted surgery field. I will choose a more common error bound, covariance, which is described next.

3.2.2 Stochastic Bounds

Perhaps the most familiar metric for the "goodness" of a measurement is *standard deviation*. Standard deviation is a measure of how measurements vary about their means, and is the square root of the measurements' *variance*. Variance of a set of *n* measurements is defined by

$$\sigma_x{}^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2 \tag{3.12}$$

where $\bar{x}$ represents the measurements' mean position obtained by

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{3.13}$$

In multivariate cases, another measure, *covariance*, is necessary to describe joint errors between the variables. Covariance is given by

$$\sigma_{xy}{}^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y}) \tag{3.14}$$

and is related to the variances of the variables by the correlation coefficient

$$\rho_{xy} = \frac{\sigma_{xy}{}^2}{\sigma_x \sigma_y} \tag{3.15}$$

which can vary from -1 to 1. More detail can be found in [36].

In the 6 DOF case, the covariances are listed in a 6x6 matrix, providing us with a convenient representation for mathematical manipulation of all the covariances. The matrix has the form

$$C = \begin{bmatrix} \sigma_x^2 & \rho_{xy}\sigma_x\sigma_y & \Lambda & \Lambda & \Lambda & \rho_{x\psi}\sigma_x\sigma_\psi \\ \rho_{xy}\sigma_x\sigma_y & \sigma_y^2 & & & & \\ \mathrm{M} & & \sigma_z^2 & & & \\ \mathrm{M} & & & \sigma_\phi^2 & & \\ \mathrm{M} & & & & \sigma_\theta^2 & \\ \rho_{x\psi}\sigma_x\sigma_\psi & & & & & \sigma_\psi^2 \end{bmatrix} \tag{3.16}$$

Note that an element of the covariance matrix can be expressed as

$$C_{ij} = \rho_{ij}\sigma_i\sigma_j \tag{3.17}$$

This is consistent with variances since for elements along the diagonal, $i=j$ and $\rho_{ii}=1$.

Durrant-Whyte presents a stochastic model of an object's position as measured by a stereo camera and a robot manipulator [37]. His model uses the inverse of the covariance matrix, the *information* matrix, which is useful in that when only partial information is available (e.g., only an object's x and y position are observable), the infinite variances associated with the unknown information are represented as zeros. The network of transformations relating disparate objects is modeled by a path matrix whose elements are 0 or 1 to indicate a whether or not a transformation belongs to a loop of transformations, and an incidence matrix having elements 1, 0, or -1 that indicate which objects each transformation exists between. These two matrices are used in algorithms that adjust the transformation estimations so the network remains consistent, that is, the loops "sum" to zero.

Smith and Cheeseman present a method for modeling transformations with means and covariances [38]. The problem they were interested in is estimating the final position and uncertainty of a mobile robot after a number of moves with respect to its initial position.

This relation is derived by methodically reducing the intermediate transformations (determined with say, wheel encoders) associated with each move to a single transformation. Additionally, the robot's position could be remotely observed (e.g., with a camera system) providing a direct measure of the relative initial and final positions. Algorithms are presented that combine two serial connected transformations as well as two parallel connected transformations. Repeated use of the combination algorithms results in the single desired estimate of the position and uncertainty. This implementation integrates more naturally into the registration graph than that of Durrant-Whyte since the event tags on registration graph links make only parts of the registration graph valid at a given moment. Representing the registration graph with path and incidence matrices would be difficult at best.

3.3 The Approximate Transformation

Instead of representing the measurements of a feature with respect to another simply by its position vector, we want to include the covariance matrix that represents the uncertainty of the measurement. Smith and Cheeseman call this vector and matrix combo the *approximate transformation*. When using a computer-assisted surgery system, measurements can be repeatedly sampled and means and variances can be explicitly calculated. In cases where this is impossible and when exploring a registration strategy simply "on paper", a priori estimates of covariances can be used with single measurements to provide a measure of how accurate the registration strategy is. In cases where only the variances or standard deviations are known (e.g., a robot's accuracy specification from the manufacturer), the covariances will be assumed to be zero.

Given that the links of the registration graph will now carry mean and covariance information, we need to look the methods for combining the information so that approximate transformation between disparate features, such as a "tool" and a "plan", can be determined.

3.3.1 Serial Compounding

Serial compounding is used to combine the approximate transformations of two links that connect three features. Such compounding is found in the registration graph when two (or more) direct links create an induced link. The approximate transformation of the induced link is derived from the approximate transformations of the direct links. In Figure 3.2, direct links [CMM : base → END-EFFECTOR : probe] and [CMM : base → END-EFFECTOR : mounting flange] induce the link [END-EFFECTOR : mounting flange → END-EFFECTOR : probe]. Also, the links [CMM : base → END-EFFECTOR : cutting guide (tool)] and [CMM : base → END-EFFECTOR : mounting flange] induce the link [END-EFFECTOR : mounting flange → END-EFFECTOR : cutting guide (tool)]. The nominal transformations are computed by

$$ {}^{\substack{\text{END EFFECTOR :}\\ \text{mounting flange}}}\mathbf{T}_{\substack{\text{END EFFECTOR :}\\ \text{probe}}} = {}^{\substack{\text{END EFFECTOR :}\\ \text{mounting flange}}}\mathbf{T}_{\substack{\text{CMM :}\\ \text{base}}} \cdot {}^{\substack{\text{CMM :}\\ \text{base}}}\mathbf{T}_{\substack{\text{END EFFECTOR :}\\ \text{probe}}} \tag{3.18} $$

$$ {}^{\substack{\text{END EFFECTOR :}\\ \text{mounting flange}}}\mathbf{T}_{\substack{\text{END EFFECTOR :}\\ \text{cutting guide (tool)}}} = {}^{\substack{\text{END EFFECTOR :}\\ \text{mounting flange}}}\mathbf{T}_{\substack{\text{CMM :}\\ \text{base}}} \cdot {}^{\substack{\text{CMM :}\\ \text{base}}}\mathbf{T}_{\substack{\text{END EFFECTOR :}\\ \text{cutting guide (tool)}}} \tag{3.19} $$

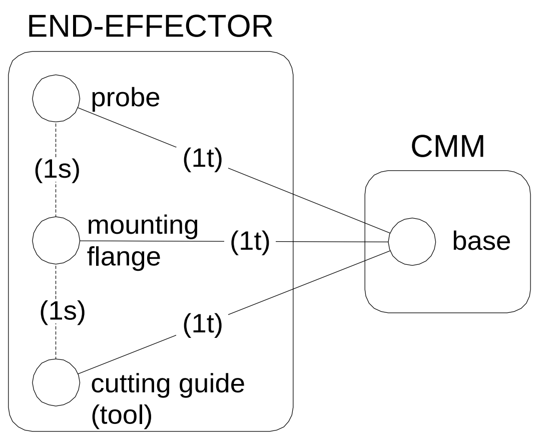


Figure 3.2: The approximate transformations associated with direct links are used to derive the approximate transformations of the induced links.

where **T** is the homogeneous matrix form of the transformation. Note that the transformation ${}^{END\ EFFECTOR\ :\ mounting\ flange}\mathbf{T}_{CMM\ :\ base}$ would seem to imply that the end effector measures its relation with respect to the CMM. In actuality, the reverse measurement is achievable. Therefore, the transformation ${}^{END\ EFFECTOR\ :\ mounting\ flange}\mathbf{T}_{CMM\ :\ base}$ is found by inverting the achievable transformation

$$ {}^{\substack{END\ EFFECTOR: \\ mounting\ flange}}\mathbf{T}_{CMM:base} = {}^{CMM:base}\mathbf{T}^{-1}_{\substack{END\ EFFECTOR: \\ mounting\ flange}} \tag{3.20} $$

When the reverse transformation is needed, we must also reverse the covariance matrix. This is done by first approximating the inverse nominal transformation by a first-order Taylor series expansion to obtain a deviate matrix

$$ [\Delta x' \quad \Delta y' \quad \Delta z' \quad \Delta\phi' \quad \Delta\theta' \quad \Delta\psi']^{tr} = \mathbf{J}[\Delta x \quad \Delta y \quad \Delta z \quad \Delta\phi \quad \Delta\theta \quad \Delta\psi]^{tr} \tag{3.21} $$

where $'$ denotes the reverse nominal transformation and **J** is the 6x6 Jacobian evaluated at the mean values of the variables

$$\mathbf{J} = \begin{bmatrix} \frac{\partial x'}{\partial x} & \frac{\partial x'}{\partial y} & \Lambda & \Lambda & \Lambda & \frac{\partial x'}{\partial \psi} \\ \frac{\partial y'}{\partial x} & \frac{\partial y'}{\partial y} & & & & \\ \mathrm{M} & & \mathrm{O} & & & \\ \mathrm{M} & & & \mathrm{O} & & \\ \mathrm{M} & & & & \mathrm{O} & \\ \frac{\partial \psi'}{\partial x} & & & & & \frac{\partial \psi'}{\partial \psi} \end{bmatrix} \tag{3.22}$$

The estimate of the reverse covariance is achieved through “squaring” both sides of equation by post-multiplying by their respective transposes and taking the expectation of the result

$$E\left(\begin{bmatrix} \Delta x' \\ \Delta y' \\ \Delta z' \\ \Delta \phi' \\ \Delta \theta' \\ \Delta \psi' \end{bmatrix} \begin{bmatrix} \Delta x' & \Delta y' & \Delta z' & \Delta \phi' & \Delta \theta' & \Delta \psi' \end{bmatrix} \right) = E\left(\mathbf{J} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \\ \Delta \phi \\ \Delta \theta \\ \Delta \psi \end{bmatrix} \begin{bmatrix} \Delta x & \Delta y & \Delta z & \Delta \phi & \Delta \theta & \Delta \psi \end{bmatrix} \mathbf{J}^{tr} \right) \tag{3.23}$$

$$\mathbf{C}' \cong \mathbf{J}\mathbf{C}\mathbf{J}^{tr} \tag{3.24}$$

For the reverse transformation of equation 3.20, the reverse covariance is

$$ {}^{\substack{\mathit{END\ EFFECTOR:} \\ \mathit{mounting\ flange}}}\mathbf{C}_{\mathit{CMM:base}} \cong \mathbf{J}\, {}^{\mathit{CMM:base}}\mathbf{C}_{\substack{\mathit{END\ EFFECTOR:} \\ \mathit{mounting\ flange}}}\, \mathbf{J}^{tr} \tag{3.25}$$

Once the nominal transformations have been computed and the covariance matrices reversed so that the subgraph can be traversed in the correct direction, serially compounded covariances can be computed. This is done by approximating the compound nominal transformation by a first-order Taylor series expansion to obtain a deviate matrix, and taking the expectation of the "squared" deviate matrix. For the induced link [END-EFFECTOR : mounting flange → END-EFFECTOR : probe], we obtain the following covariance matrix

$$^{\substack{END\ EFFECTOR: \\ mounting\ flange}}\mathbf{C}_{\substack{END\ EFFECTOR: \\ probe}} \cong \mathbf{J}\begin{bmatrix} ^{\substack{END\ EFFECTOR: \\ mounting\ flange}}\mathbf{C}_{\substack{CMM: \\ base}} & \mathbf{0} \\ \mathbf{0} & ^{\substack{CMM: \\ base}}\mathbf{C}_{\substack{END\ EFFECTOR: \\ probe}} \end{bmatrix}\mathbf{J}^{tr} \tag{3.26}$$

where the $\mathbf{J}$ is the 6x12 Jacobian of the compound transformation evaluated at the means of the variables x, y, z, ϕ, θ, ψ of the transformations $^{END\ EFFECTOR\ :\ mounting\ flange}\mathbf{T}_{CMM\ :\ base}$ and $^{CMM\ :\ base}\mathbf{T}_{END\ EFFECTOR\ :\ probe}$. The 6x12 Jacobian can be written as the concatenation of two 6x6 Jacobians as

$$\mathbf{J} \cong [\mathbf{P}|\mathbf{Q}] \tag{3.27}$$

and equation 3.26 then takes the form

$$^{\substack{END\ EFFECTOR: \\ mounting\ flange}}\mathbf{C}_{\substack{END\ EFFECTOR: \\ probe}} \cong \mathbf{P}\,^{\substack{END\ EFFECTOR: \\ mounting\ flange}}\mathbf{C}_{\substack{CMM: \\ base}}\mathbf{P}^{tr} + \mathbf{Q}\,^{\substack{CMM: \\ base}}\mathbf{C}_{\substack{END\ EFFECTOR: \\ probe}}\mathbf{Q}^{tr} \tag{3.28}$$

3.3.2 Parallel Compounding

Parallel compounding is used to combine the approximate transformations of two links that connect just two features. While this is not directly allowed in the registration graph, it is encountered when calculating the approximate transformation for an induced link where two simultaneous connected subgraphs have been reduced to two links by the serial compounding operation of Section 3.3.1. A hypothetical example is shown in Figure 3.3 where the induced link [ROBOT : base → OBJECT : base] is created by the subgraphs [ROBOT : base → CAMERA1 : base → OBJECT : base] and [ROBOT : base → CAMERA2 : base → OBJECT : base].

The first step in parallel compounding is to compute the Kalman gain matrix. The Kalman filter [39] is used extensively in the field of digital filtering to estimate the current state of a signal based on previous estimates and a current measure of the signal. The Kalman gain matrix is a weighting factor applied to the current measurement and previous

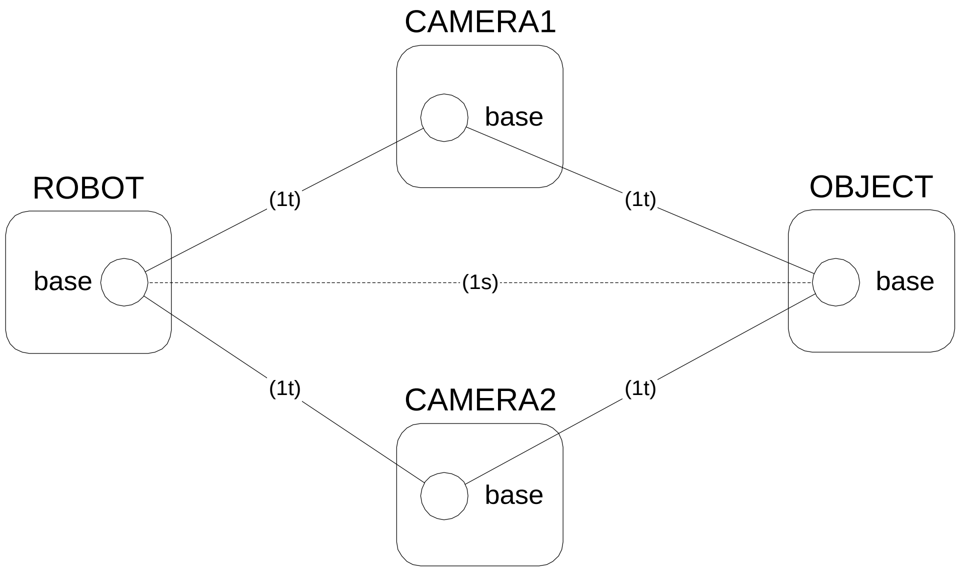


Figure 3.3: An induced link may have several subgraphs that validate it. The link [ROBOT : base → OBJECT : base] has two valid subgraphs: one that passes through [CAMERA1 : base] and another that passes through [CAMERA2 : base].

estimates that minimizes the covariance of the current estimate. A concise overview of the Kalman filter can be found in [40]. In the example of Figure 3.3, the two links that result from serially compounding the subgraphs that induce [ROBOT : base → OBJECT : base] produce the Kalman gain matrix

$$\mathbf{K} = \mathbf{C}_1[\mathbf{C}_1 + \mathbf{C}_2]^{-1} \tag{3.29}$$

where $\mathbf{C}_1$ is the serially compounded covariance from the subgraph that passes through the CAMERA1 rigid body, and $\mathbf{C}_2$ is the serially compounded covariance from the subgraph that passes through the CAMERA2 rigid body. The Kalman gain matrix is used to compute the covariance associated with the induced link by

$$\mathbf{C}_3 = \mathbf{C}_1 - \mathbf{K}\mathbf{C}_1 \tag{3.30}$$

and the parallel compounded estimate of the nominal transformation is the weighted average

$$\begin{bmatrix} x_3 \\ y_3 \\ z_3 \\ \phi_3 \\ \theta_3 \\ \psi_3 \end{bmatrix} =_1 \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ \phi_1 \\ \theta_1 \\ \psi_1 \end{bmatrix} + \mathbf{K}\left(\begin{bmatrix} x_2 \\ y_2 \\ z_2 \\ \phi_2 \\ \theta_2 \\ \psi_2 \end{bmatrix} - \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ \phi_1 \\ \theta_1 \\ \psi_1 \end{bmatrix} \right) \tag{3.31}$$

For parallel compounding, appropriate transformation reversals must be done so all subgraphs are traversed from a “start” feature to a “finish” feature.

3.3.3 Combining Procedure

The procedures given in Section 3.3.1 and Section 3.3.2 for serial and parallel compounding can be used to reduce numerous subgraphs that each contain several links. A recursive algorithm is described by Smith and Cheeseman [38] that accomplishes this:

1. Mark the “start” and “finish” features.
2. Compound any two serial links whose common feature is not the “start” or “finish” feature.
3. Compound any two parallel links.
4. Repeat steps 2 and 3 until the set of subgraphs has been reduced to a single link.

Using this algorithm, the approximate transformation for almost any induced link in the registration graph can be computed. The one caveat is that the approximate transformation for a link whose subgraphs are arranged as a “Wheatstone bridge” cannot be computed. For

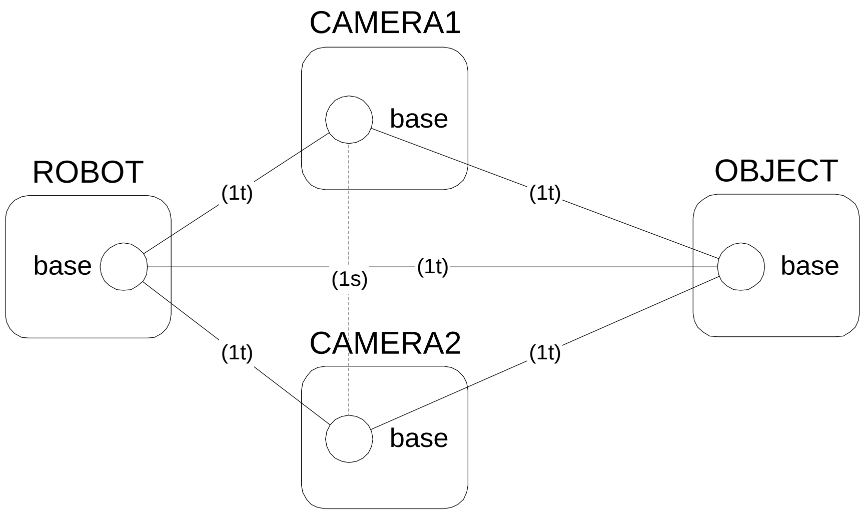


Figure 3.4: The approximate transformations for the induced link [CAMERA1 : base → CAMERA2 : base] is not computable since the subgraphs cannot be reduced to a single link.

example in Figure 3.4, the approximate transformation for the link [CAMERA1 : base $\rightarrow$ CAMERA2 : base] cannot be computed because the subgraphs will not reduce to a single link. This can addressed by using a "Delta-Y" transformation adopted from electrical circuit theory. However, in practice I have not encountered this problem in a registration graph.

3.4 Visualizing the Error Ellipsoid

The nominal transformation ($x, y, z, \phi, \theta, \psi$)associated with a link is readily understood, as it relates the relative position of two features. The variances (or standard deviations) associated with the variables are also easily interpreted as likelihood bounds on the nominal transformation. However, covariances are a bit more difficult to interpret, so we will derive a set of equations that will allow us to plot the error ellipses of the hyper-dimensional error ellipsoid.

Recall that the approximate transformation is represented by a nominal transformation (a 6-parameter vector) and a covariance matrix. Assuming that the probability distribution about the nominal transformation is Gaussian, the probability that some transformation is lies near the nominal transformation is given by

$$P(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^n \det \mathbf{C}}} e^{-\frac{1}{2}[(\mathbf{x}-\bar{\mathbf{x}})^{tr} \mathbf{C}^{-1}(\mathbf{x}-\bar{\mathbf{x}})]} \tag{3.32}$$

where $\mathbf{x}$ is the 6-parameter vector of some transformation, $\bar{\mathbf{x}}$ is 6-parameter vector of the nominal transformation, and n is the number of dimensions. The surfaces of constant probability define the hyper-dimensional error ellipsoid, and are centered about the nominal transformation $\bar{\mathbf{x}}$. The equation of this ellipsoid is given by

$$(\mathbf{x}-\bar{\mathbf{x}})\mathbf{C}^{-1}(\mathbf{x}-\bar{\mathbf{x}})^{tr} = k^2 \tag{3.33}$$

where *k* is a constant chosen for a particular confidence threshold. It is impossible to draw the six dimensional error ellipsoid, so we instead draw the set of error ellipses that is the projections of the six dimensional error ellipsoid onto the 15 two dimensional planes (i.e., *xy*, *xz*, *xφ*, … *θψ*). For each plane, the covariance matrix is reduced by striking out all rows and columns except those that we are interested in. For instance, to project into the *xy* plane, the covariance matrix reduces to

$$\mathbf{C} = \begin{bmatrix} \sigma_x^2 & \rho\sigma_x\sigma_y \\ \rho\sigma_x\sigma_y & \sigma_y^2 \end{bmatrix} \tag{3.34}$$

Substituting this into Equation 3.33 yields,

$$\begin{bmatrix}(x-\bar{x}) & (y-\bar{y})\end{bmatrix}\left(\frac{1}{\sigma_x^2\sigma_y^2(1-\rho^2)}\begin{bmatrix} \sigma_y^2 & -\rho\sigma_x\sigma_y \\ -\rho\sigma_x\sigma_y & \sigma_x^2 \end{bmatrix}\right)\begin{bmatrix}(x-\bar{x}) \\ (y-\bar{y})\end{bmatrix} = k^2 \tag{3.35}$$

which can be further manipulated into a form corresponding to the family of two-dimensional ellipses

$$\frac{1}{\sigma_x^2\sigma_y^2(1-\rho^2)}\left(\sigma_y^2(x-\bar{x})^2 - 2\rho\sigma_x\sigma_y(x-\bar{x})(y-\bar{y}) + \sigma_x^2(y-\bar{y})^2\right) = k^2 \tag{3.36}$$

The general form of two dimensional ellipses whose origin is $(\bar{x},\bar{y})$ is

$$A(x-\bar{x})^2 + 2B(x-\bar{x})(y-\bar{y}) + C(y-\bar{y})^2 = k^2 \tag{3.37}$$

Comparing Equations 3.36 and 3.37, we find

$$A = \frac{1}{\sigma_x^2(1-\rho^2)} \tag{3.38}$$

$$B = -\frac{\rho^2}{\rho\sigma_x\sigma_y(1-\rho^2)} \tag{3.39}$$

$$C = \frac{1}{\sigma_y^2(1-\rho^2)} \tag{3.40}$$

To plot the ellipse, we need to know the angle α that the major axis of the ellipse makes with the *x*-axis, and the lengths of the major and minor axes (i.e., the largest and smallest "radii") *m* and *n*. The angle α is given by

$$\alpha = \frac{1}{2}\arctan\left(\frac{2B}{A-C}\right), \qquad -\frac{\pi}{2} \le \alpha \le \frac{\pi}{2} \tag{3.41}$$

and the lengths of the major and minor axes are

$$m = \sqrt{\frac{2k^2}{A+C-T}} \tag{3.42}$$

$$n = \sqrt{\frac{2k^2}{A+C+T}} \tag{3.43}$$

where

$$T = \sqrt{A^2 + C^2 - 2AC + 4B^2} \tag{3.44}$$

3.5 Summary

We now have a set of tools for both qualitative and quantitative analysis of registration in computer-assisted surgery. However, manually drawing graphs and making longhand calculations is tedious. In the next chapter, I present a C++ implementation of the rules and algorithms of Chapters 2 and 3, as well as an interactive Windows 95/98 program for exploring registration strategies.

4 A C++ Class Library and Graphical Program

As a proof of viability of the registration graph developed in Chapter 2, I wanted to implement the rules in computer code so that registration graphs could be automatically analyzed. The uncertainty analysis of Chapter 3 also was a candidate for implementation. What follows in this chapter are an overview of the C++ class library developed that realizes the registration graph and a description of a graphical CAD-like program that runs under Microsoft Windows 95/98. The program will be used to illustrate a simple example in Chapter 5.

4.1 The C++ Library

In order to implement the registration graph in computer code, the first decision one faces is what language to use. Due to its popularity in the engineering field, the C language [41] was a front-runner candidate. However, the registration graph is inherently object-oriented in nature, and C's sequel - C++ - was chosen. Using the C++ language [42], each element of the registration graph could be implemented as an object class that defines both its inherent properties (e.g., a rigid body contains a list of its features, an event tag is transient or sustained) and rules that govern how the element interacts with other elements of the registration graph (e.g., an induced link is valid if a simultaneously connected subgraph connecting its features exists). Below I discuss the key aspects of each element of the registration graph library. For this chapter, knowledge of the C++ language is not a prerequisite but it is assumed that the reader has a general "feel" for object-oriented design. To those conversant only in C (or similar language) I offer this: think of a C++ object simply as a C `struct` that contains programming variables *and* functions that act on those variables or other objects.

To dynamically manage the memory allocated to each object, I have used the Standard Template Library (STL) [43][44], which is a set of definitions (header files) for data structures like vectors, linked lists, etc. For vector and matrix manipulation I have written a small, uncomplicated set of classes. Therefore, the registration graph library should easily port and recompile to platforms other than the PC on which it was developed.

4.1.1 The Hierarchy

A C++ *class* has been written for each element of the registration graph. Each class is a definition of the set of variables and functions associated with each element. While an instance of each element can be created as a stand-alone object in code (e.g., just as `int i` creates an integer object named "i", `RGFeature femur` creates a feature object named "femur"), they are useless for analysis until assigned to their appropriate "parent". This is accomplished with member functions in each class that manage the addition and deletion of "child" objects. Internally, each "parent" objects maintains a list of its "child" objects. The hierarchy of ownership is shown in Figure 4.1.

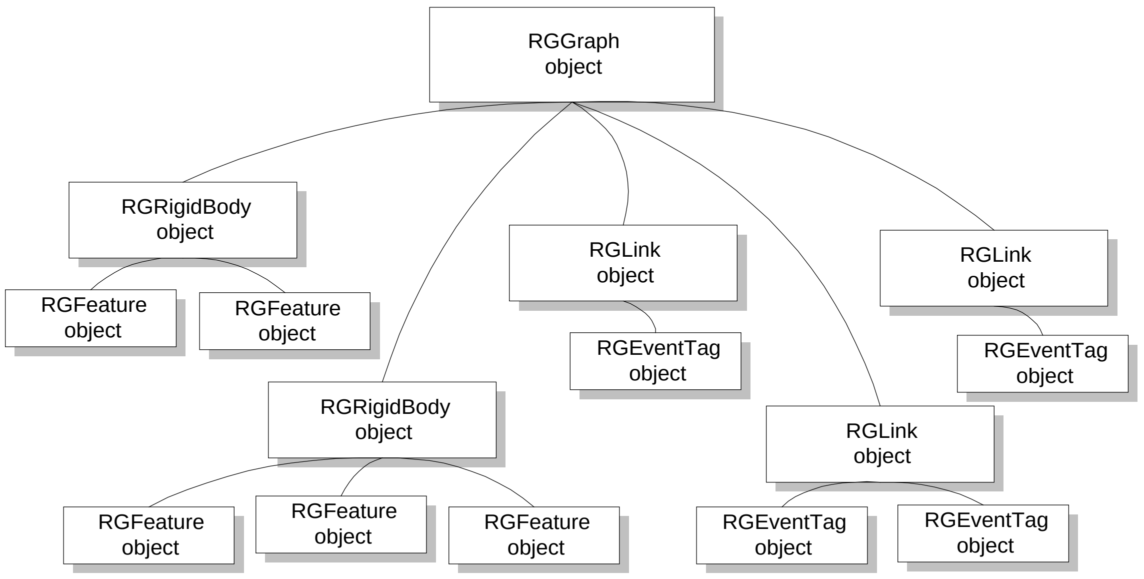


Figure 4.1: Heirarchy of object ownership in the registration graph C++ library.

4.1.2 The Registration Graph Class

The top-level class is the `RGGraph` class. The prefix `RG` denotes that this class belongs to the *R*egistration *G*raph library. Inside an `RGGraph` object, two lists (STL `vector<>` templates) are maintained: one that records the rigid bodies (`RGRigidBody` objects) belonging to the graph and another that records the links (`RGLink` objects). Recall that features and links are the primary elements of the registration graph. This hierarchical arrangement for the class library does not violate the registration graph architecture; the rigid body class could have been left out or made a "sibling" of the feature class. By setting the rigid body class hierarchically above the feature class so the `RGGraph` class does not manage a list of features, we may use the same intuitive name identifier for many features in the graph (e.g. "base"), rather than more confusing names (e.g., "CTbase" or "ROBOTbase").

Management of the lists is provided through a set of public member functions - functions that are accessible by the user of the class library. The public functions are shown in Table 4.1. Mutator functions (`AddRigidBody`, `DeleteRigidBody`, …) are used to add and remove rigid body and link objects from the internal lists. Inspector functions (`GetRigidBody`, `GetLink`) search the appropriate internal list for an object matching the identifier argument; a name string (e.g., "ROBOT") in the case of a rigid body object and two features defining a link in the case of a link object. The `IsUniqueRigidBody` and `IsUniqueLink` analysis functions are called by their respective `Add`… functions to check whether or not a similar rigid body / link has been added to the graph already. The functions check the their object argument against those in the internal lists by checking the objects' identifier(s). These functions are publicly listed since they are also useful to a programmer (a user of the library). Finally, the internal control routines (`RewindRigidBodies`, `AdvanceRigidBodies`, …) are used to traverse the object lists.

```
Public:
   // mutators
   bool AddRigidBody(RGRigidBody *const);
   bool DeleteRigidBody(RGRigidBody *const);
   bool AddLink(RGLink *const);
   bool DeleteLink(RGLink *const);

   // inspectors
   RGRigidBody *const GetRigidBody(const String);
   RGLink *const      GetLink(const RGFeature *const, const RGFeature *const);

   // analysis routines
   bool IsUniqueRigidBody(RGRigidBody *const);
   bool IsUniqueLink(RGLink *const);

   // internal control routines
   RGRigidBody *const RewindRigidBodies(unsigned int);
   RGRigidBody *const AdvanceRigidBodies(unsigned int);
   RGLink *const      RewindLinks(unsigned int);
   RGLink *const      AdvanceLinks(unsigned int);
```

Table 4.1: Public member functions of the `RGGraph` class.

```
private:
   // mutators
   void UpdateLink(RGLink *const);
   void UpdateAllLinks();

   // analysis routines
   int  SearchAndAssignPaths(RGLink *const, const RGEventTag& tag);
   void DFS(RGLink *const, RGFeature *const, RGFeature *const,
            const RGEventTag&); // (called by SearchAndAssignPaths)
   int  ComputeMeanAndCovariance(RGLink *const);
```

Table 4.2: Private member functions of the `RGGraph` class.

Internally, the class uses private functions inaccessible to the programmer. These functions, listed in Table 4.2, are implementations of the registration graph rules in Chapter 2 and the approximate transformation compounding algorithms in Chapter 3. `UpdateLink` calls `SearchAndAssignPaths` to revise the list of valid subgraphs (discussed in Section 4.1.5) of an induced link and `ComputeMeanAndCovariance` to update an induced link's approximate transformation (mean and covariance is *assigned* for a direct link). `UpdateAllLinks` simply calls `UpdateLink` for all induced links in the registration graph. The analysis routine `DFS` is used by `SearchAndAssignPaths` to explore the registration graph for valid subgraphs using a depth-first search strategy.

4.1.3 The Rigid Body Class

The `RGRigidBody` class contains a string variable that identifies the rigid body by an assigned name. Therefore, the registration graph may not have two similarly named rigid body objects (This is checked by the `IsUniqueRigidBody` function of the `RGGraph` class). Also, the class contains a list that records all features that belong a rigid body object. The public functions of the `RGRigidBody` class are listed in Table 4.3.

```
public:
   // mutators
   bool SetName(const String);
   bool AddFeature(RGFeature *const);
   bool DeleteFeature(RGFeature *const);

   // inspectors
   RGGraph *const   GetParentGraph();
   const String     GetName();
   RGFeature *const GetFeature(const String);

   // analysis routines
   bool IsUniqueFeature(RGFeature *const);

   // internal control routines
   RGFeature *const RewindFeatures(unsigned int);
   RGFeature *const AdvanceFeatures(unsigned int);
```

Table 4.3: Public member functions of the `RGRigidBody` class.

The mutator function `SetName` changes the string variable to the function's argument as long as no other rigid body in the registration graph has the same name. `AddFeature` and `DeleteFeature` work in a similar manner to their counterparts in the `RGGraph` class - they add and remove features from the internal list. The inspector routines `GetParentGraph` and `GetName` return a pointer to the object's owner and the object's name respectively. `GetFeature` searches the internal list for a feature matching the name identifier. `IsUniqueFeature` checks whether or not a similarly named feature already exists in the rigid body's internal feature list. The control routines `RewindFeatures` and `AdvanceFeatures` are used to traverse the feature list.

4.1.4 The Feature Class

The `RGFeature` class is fairly simple, since features simply provide endpoints for the links of the registration graph. The `RGFeature` class also contains a string variable that identifies the feature. The public functions of the RGRigidBody class are listed in Table 4.4. The functions work similarly to that of the equivalent functions in the RGRigidBody class.

```
public:
   // mutators
   bool SetName(const String);

   // inspectors
   RGRigidBody *const GetParentRigidBody();
   const String       GetName();
```

Table 4.4: Public member functions of the RGFeature class.

4.1.5 The Link Class

Most of the functionality of the registration graph is embedded in the link element. The RGLink class reflects this, as it is the largest in the library. The internal variables include: a link type variable (direct or induced), references to the features the link connects, a list of event tags, a list of subgraph paths (each subgraph path is a list of the features traversed), and a vector representing the nominal (mean) position and matrix for covariances. The public functions of the class are listed in Table 4.5.

The Set… mutator functions assign their listed argument to the appropriate internal variable. SetCovariance has two incarnations - one whose argument is a vector of the six standard deviations (the standard deviations are squared and inserted on the diagonal of the covariance matrix), and a second that takes a matrix an directly assigns it to the covariance matrix variable. SwapFeatures is used to exchange the "start" and "finish" features references. This is used when a link has been specified in one direction, but the mean and

```
public:
   // mutators
   bool SetType(const RGLinkType);
   bool SetStartFeature(RGFeature *const);
   bool SetFinishFeature(RGFeature *const);
   void SwapFeatures();
   bool SetMean(const RowVector6);
   bool SetCovariance(const RowVector6);
   bool SetCovariance(const Matrix6);

   bool AddEventTag(RGEventTag *const);
   bool DeleteEventTag(RGEventTag *const);

   // inspectors
   RGGraph *const    GetParentGraph();
   const RGLinkType  GetType();
   RGFeature *const  GetStartFeature();
   RGFeature *const  GetFinishFeature();
   RGEventTag *const GetEventTag(const RGTagType, const unsigned int);
   const String      GetStringOfEventTags(int);
   RowVector6        GetMean();
   Matrix6           GetCovariance();

   // analysis routines
   const bool HasDefaultTag() const;
   const bool IsUniqueEventTag(RGEventTag *const);
   const int  IsPassable() const;
   const bool IsAddableEventTag(RGEventTag *const);
   const bool HasSameStartTimeAs(RGLink *const);

   // internal control routines
   RGEventTag *const RewindEventTags(unsigned int);
   RGEventTag *const AdvanceEventTags(unsigned int);
```

Table 4.5: Public member functions of the `RGLink` class.

covariance are known in the opposite direction. `AddEventTag` and `DeleteEventTag` are used to add and remove `RGEventTag` objects from the event tag list.

The `Get`… inspector functions are used to retrieve internal variables. `GetStringOfEventTags` is the only uniquely new inspector function. It is used to create a string listing the event tags, e.g., "1t, 2t", or "3s". By default, all links will have at least one tag: "1t".

The analysis functions are of particular interest. `HasDefaultTag` is used to check whether or not any event tags have been assigned to the link. If not, a tag of "1t" is used. `IsUniqueEventTag` is used to check if any event tags are stored in the list with the same value and type (transient or sustained). `IsPassable` returns the number of valid subgraph paths stored in the list of paths. If zero is returned, there are no valid subgraphs. `IsAddableEventTag` determines whether or not an event tag can be added to the link. A transient tag may be added to the link if there are no other transient tags with the same value (event), or if there are any sustained event tags with a value less than or equal to that of the tag in question. A sustained tag may be added if there are no other tags on the link that cover the same (or more) events. For example, a "1s" tag may be added even if there is a "1t" and/or a "2s" tag on the link. If the tag is to be added to the link (with `AddEventTag`), the lesser tags ("1t" and "2s") will be removed. `HasSameStartTimeAs` compares the earliest event tag value in the link's list to the earliest event tag value of another link. This link is used by the `DFS` private function of the `RGGraph` class to determine whether or not a path is passable, since a link is only traversable if it has the same start time (value) as the induced link's start time.

The internal control routines, `RewindEventTags` and `AdvanceEventTags`, are used to traverse the event tag list.

4.1.6 The Event Tag Class

The last class that makes up the registration graph library is `RGEventTag`. As with the `RGFeature` class, this class is fairly simple. The internal variables are a tag type (transient or sustained) and an integer value.

The `Set`… and `Get`… functions introduce no surprises, and work similarly to their counterparts in the other classes. The operator functions are new, and provide some interesting functionality. The = operator assigns the tag type and integer value of the right hand argument to that of the left-hand argument. The + operator returns the intersection of two tags' event spans. If two sustained tags are "summed", the tag with the larger value is returned. Two transient tags are summable only if their values are the same, otherwise a tag with a value of zero is returned†. In the case where a sustained and a transient tag are summed, the transient tag is simply returned its value is greater than that of the sustained tag,

```
public:
   // mutators
   bool SetType(const RGTagType);
   bool SetValue(const unsigned int);

   // inspectors
   RGLink *const      GetParentLink();
   const RGTagType    GetType();
   const unsigned int GetValue();

   // operators
   RGEventTag&       operator =(const RGEventTag&);
   friend RGEventTag operator +(const RGEventTag&, const RGEventTag&);
   friend bool       operator ==(const RGEventTag&, const RGEventTag&);
```

Table 4.6: Public member functions of the `RGEventTag` class.

† Recall that in Chapter 2 we chose to use 1 as the earliest possible event. Now it is apparent that there was a motivating factor - we want to use zero to represent the empty set.

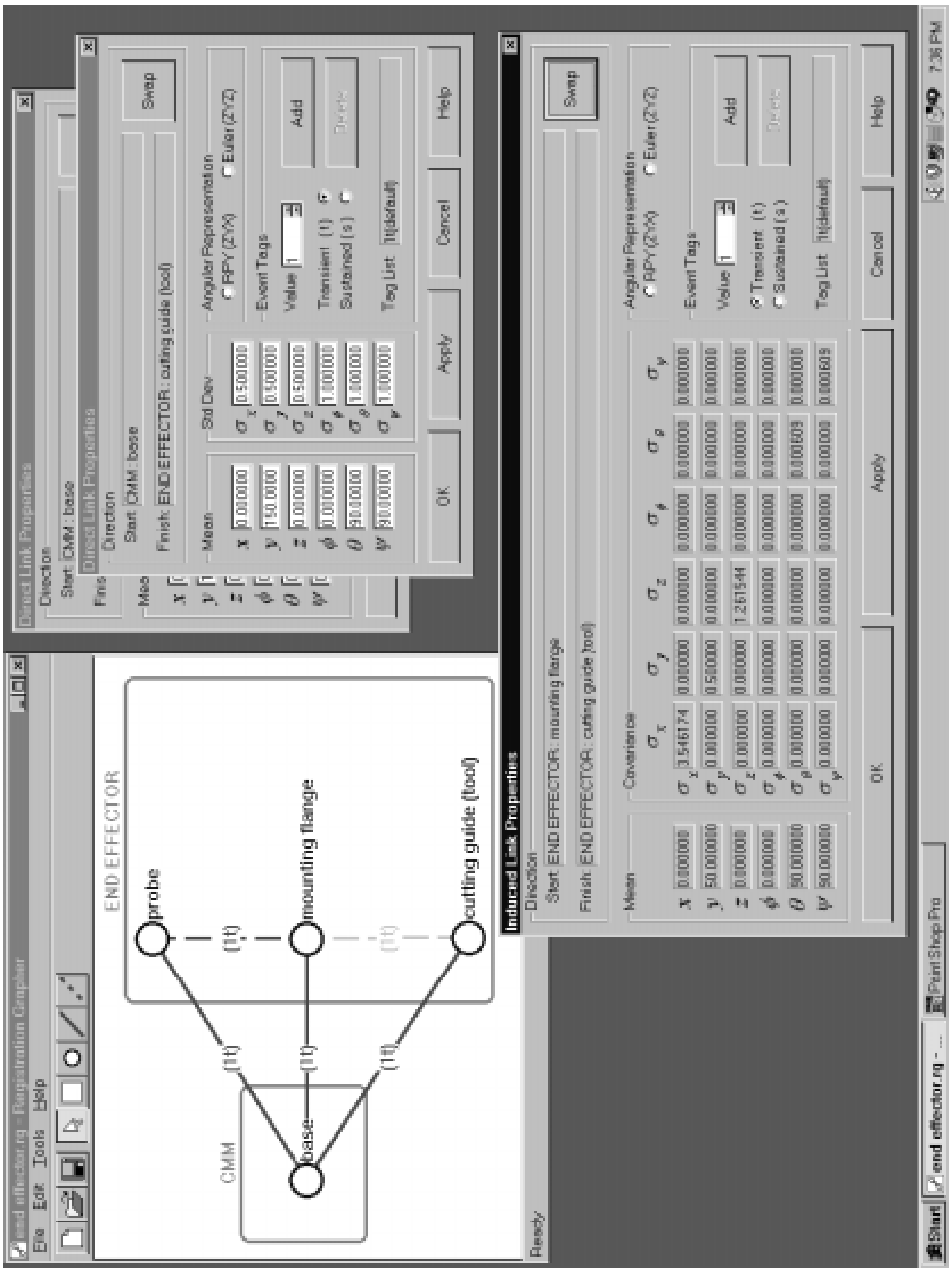
otherwise a tag with a value of zero is returned. The operator functions allow a programmer to write quite readable code. For instance, two tags can be summed and assigned to a third tag with `tag3 = tag1 + tag2`. The == operator is used to compare two event tags, and returns a "true" value if the tag type and value of both event tags are the same. Otherwise a "false" value is returned.

4.2 The *Registration Grapher* Program

During the development of the C++ class library, I wrote many command-line interpreter programs to build and test registration graphs. However, the results were less than satisfying since the registration graph is more naturally visualized as a "drawing". I decided to have a go at writing a Microsoft Windows program that would allow a user to interactively draw and explore the registration graph. I did not know what I was in for. The commercial software market redefines itself on what seems a daily basis, and many tools that I started out using fell by the wayside. For instance, at the outset of my work the Borland compiler with its OWL (Object Windows Library) for creating Windows applications was the height of fashion for programmers. By the time I completed the C++ Library, Borland had completely dropped the product, forcing me to change to the Watcom compiler that uses Microsoft's MFC (Microsoft Foundation Classes) for Windows programming. Nonetheless, I did find the experience gained useful.

What resulted from these endeavors is a program that can be used to interactively draw a registration graph. Figure 4.2 shows the program in use. The main drawing window allows the user to draw, move, and resize the elements of the registration graph. Double-clicking any element brings up a dialog box that displays all of the element's attributes. The dialogs are also interactive, and allow the user to change the names of rigid bodies and features, assign means and standard deviations to direct links, and so on. The induced link's dialog box shows the automatically calculated mean vector and covariance matrix.

This program is used in the next chapter to study the registration graph and uncertainty of the Northwestern TKR system presented in Chapter 2.



Direct Link Properties

Direction
Start: CMM : base
Finish: END EFFECTOR : cutting guide (tool)

Mean		Std Dev	
x	0.000000	σ_x	0.500000
y	150.0000	σ_y	0.500000
z	0.000000	σ_z	0.500000
ϕ	0.000000	σ_ϕ	1.000000
θ	90.00000	σ_θ	1.000000
ψ	90.00000	σ_ψ	1.000000

Angular Representation: RPY (ZYX) Euler (ZYZ)
Event Tags: Value 1; Transient (t) Sustained (s)
Tag List: 1t(default)
Swap Add Delete OK Apply Cancel Help

Induced Link Properties

Direction
Start: END EFFECTOR : mounting flange
Finish: END EFFECTOR : cutting guide (tool)

Mean		Covariance	σ_x	σ_y	σ_z	σ_ϕ	σ_θ	σ_ψ
x	0.000000	σ_x	3.546174	0.000000	0.000000	0.000000	0.000000	0.000000
y	50.00000	σ_y	0.000000	0.500000	0.000000	0.000000	0.000000	0.000000
z	0.000000	σ_z	0.000000	0.000000	1.251544	0.000000	0.000000	0.000000
ϕ	0.000000	σ_ϕ	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
θ	90.00000	σ_θ	0.000000	0.000000	0.000000	0.000000	0.000609	0.000000
ψ	90.00000	σ_ψ	0.000000	0.000000	0.000000	0.000000	0.000000	0.000609

Angular Representation: RPY (ZYX) Euler (ZYZ)
Event Tags: Value 1; [✓ Transient (t)] Sustained (s)
Tag List: 1t(default)
Swap Add Delete OK Apply Cancel Help

Figure 4.2: The Registration Grapher program.

5 A Case Study - the Northwestern TKR system

Now that a graphical method for analyzing registration has been designed, methods for studying uncertainty have been discussed, and a C++ library and Windows program to interactively "play" with registration strategies have been developed, an example of how these tools can be employed is warranted. We turn again to the Northwestern computer-assisted surgery system for total knee replacement (for an overview of the system, see the brief review in the introduction of Chapter 2 or full descriptions in [1][2]). What follows is a brief study of that system using the Registration Grapher program. In Section 5.1, measurements made of end effector features using a coordinate measuring machine (CMM) are studied. Acquisition of the CT scan of the patient's femur and preoperative planning of resection placement are studied in Section 5.2, and in Section 5.3 the final registration steps performed in the operating room are investigated.

5.1 Using the CMM

The resolution of the digital images (CT data in this case), the accuracy of intraoperative measuring devices (the robot), and the exactness of data-matching algorithms (e.g., an iterative closest point (ICP) technique [45]) surface are often regarded as the only sources of error in computer-assisted surgery systems. One of the more subtle sources of error in the Northwestern TKR system is the robot's end effector - the device used to digitize the positions of the pins in the femur and provide guidance for the blade of the saw. Admittedly, the measurements between features of the end effector are usually known more precisely than those of other operations, but the tool is a source of error and warrants modeling. By introducing the end effector into the registration graph, it is possible to investigate interesting questions often not posed. For instance, one might wonder what tolerances must be kept if the tool is mass manufactured and the only "measurements" are dimensions specified in the part drawings.

For the Northwestern system, the end effector was machined and assembled first. Subsequently, the relationships between relevant features were determined with a CMM. A

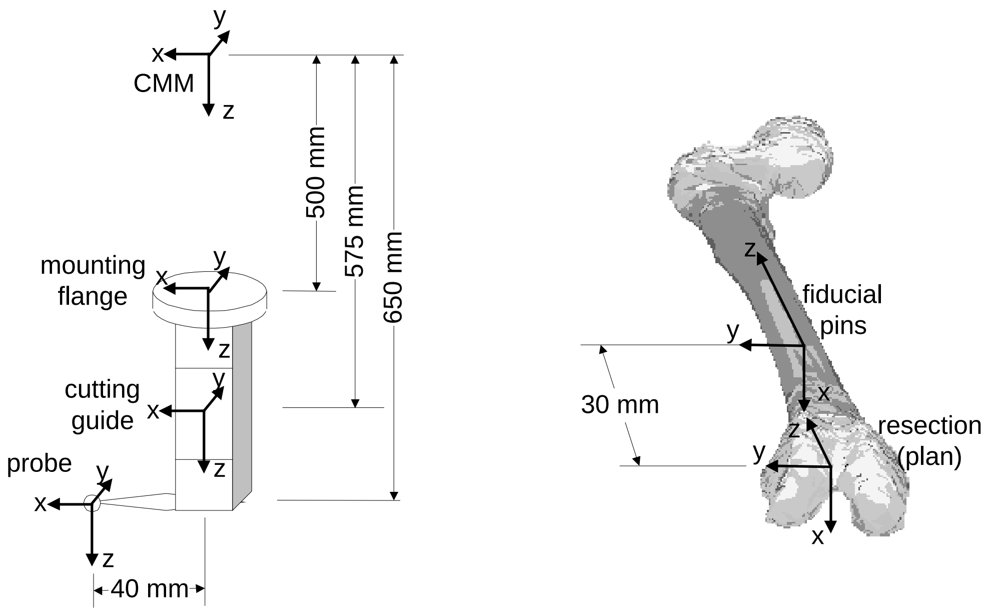


Figure 5.1: (a) The simplified distances from the CMM base reference frame to the features of the end effector.(b) The distance from the fiducial pin.

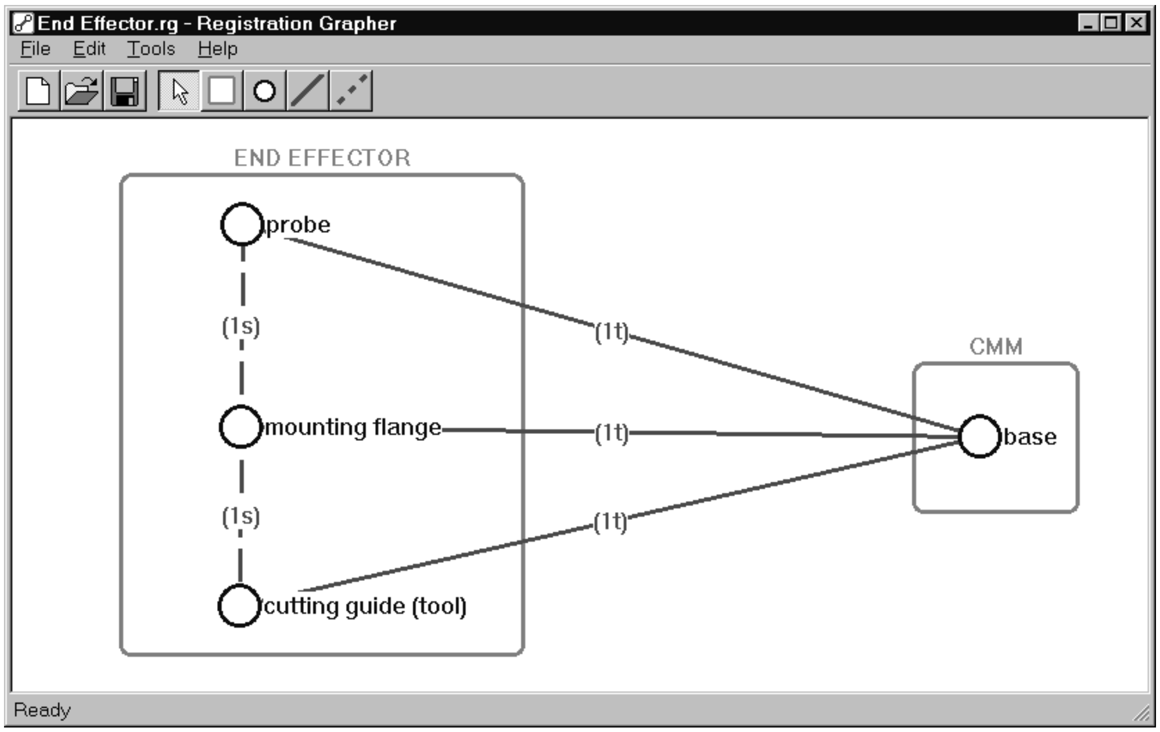


Figure 5.2: Portion of the Northwestern TKR registration graph showing measurement operations on the end effector using the CMM.

simplified version of the setup and measurements made is shown in Figure 5.1. We can study aspects of these measurement operations by looking at that portion of the Northwestern TKR registration graph in the Registration Grapher program, shown in Figure 5.2.

The nominal dimensions are assigned to the means of the direct links, and for our purposes here the accuracy of the CMM is assumed to be $\sigma_{xyz\phi\theta\psi}$ = (0.1mm, 0.1mm, 0.1mm, 0.1°, 0.1°, 0.1°). The means and covariances of the induced links are automatically computed, and can be inspected via the induced link dialog box (brought up by double clicking).

Looking at the [END EFFECTOR : mounting flange → END EFFECTOR : probe] induced link, we see that the covariance matrix has several off diagonal terms (see Table 5.1). This is due to the relative offsets in the *x* and *z* directions of the mounting flange and probe features. The off diagonal terms indicate the dependence of one error on another, and are of particular interest. For instance, since $\sigma_x = 0.296$, $\sigma_\psi = 0.141$, and $\sigma_{x\psi} = 0.026$, we find that $\rho_{x\psi} = 0.622$. This indicates that positive errors in *x* somewhat strongly effect positive errors in ψ (Recall that ρ can vary from -1 to 1). By looking at the set of 15 error ellipses corresponding to the upper triangle of the covariance matrix, (Figure 5.3), we note that those ellipses that are tipped counterclockwise from the x-axis indicate the two uncertainties are positively coupled, and that those are tipped clockwise are negatively coupled. The lengths of the principle axes of the error ellipses are equal to the standard deviations of the variables. In Figure 5.3, the ellipses are relatively small.

0.089	0	-0.018	0	1.60E-18	0.026
0	0.093	0	0.007	0.026	0
-0.018	0	0.025	0	-4.27E-19	-0.007
0	0.007	0	0.020	0	0
1.60E-18	0.026	-4.27E-19	0	0.020	6.12E-19
0.026	0	-0.007	0	6.12E-19	0.020

Table 5.1: Covariance matrix for the [END EFFECTOR : mounting flange → END EFFECTOR : probe] induced link

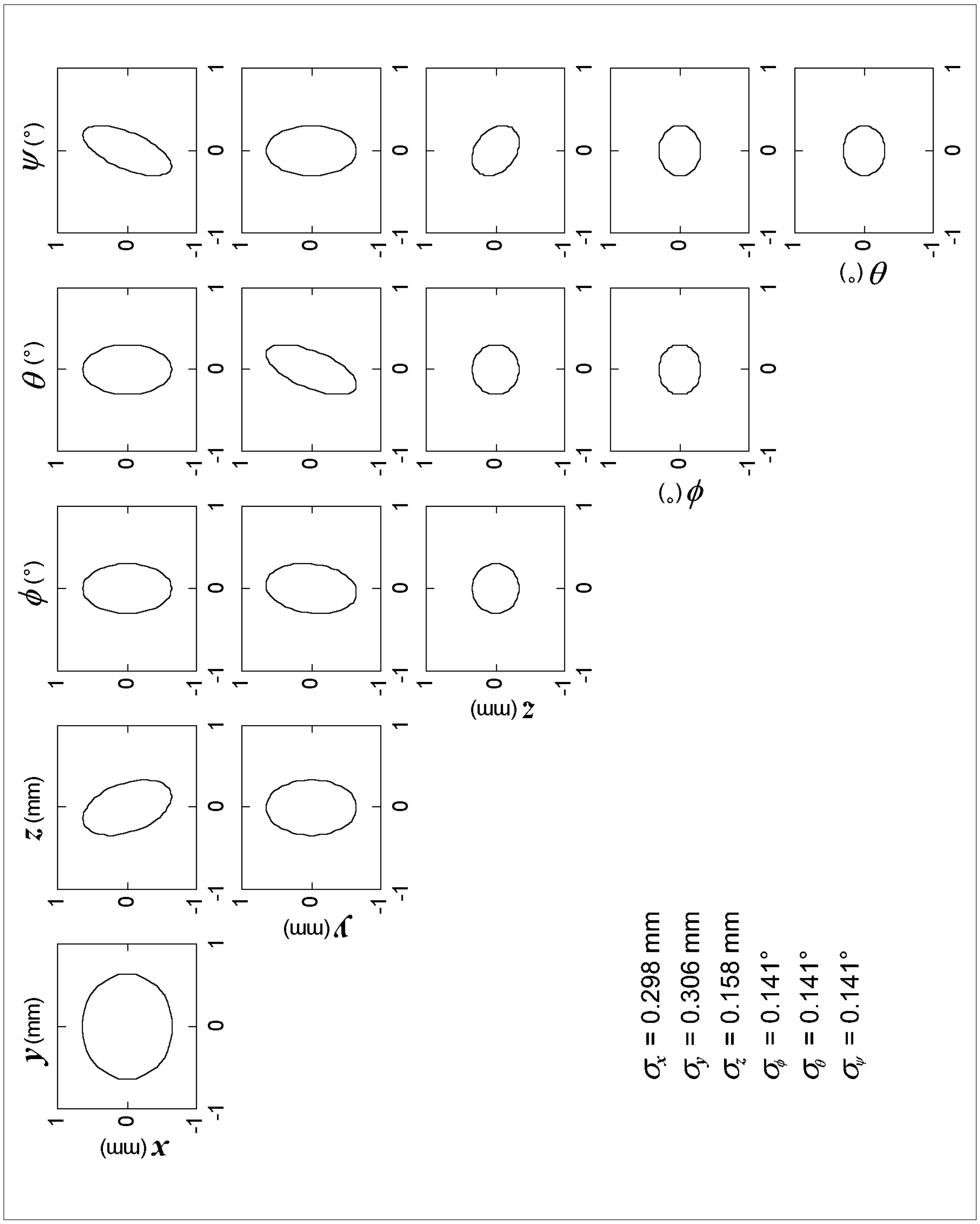


Figure 5.3: Error ellipses for the [END EFFECTOR : mounting flange → END EFFECTOR : probe] induced link. The ellipses correspond to a 90% probability (k^2 = 4.605).

5.2 Scanning the Patient

The position of the CT scanner coordinate frame is irrelevant, so we simply assign it to be coincident with the coordinate frame of the fiducial pins. The scanner is capable of spacing individual scans 1 mm apart. The pixel pitch of each scan is on the order of 0.25 mm. Since it is possible to interpolate along the direction of the scan, we will use 0.25 mm as the size of a voxel and the uncertainty in locating the origin of the fiducial pin coordinate frame (Figure 5.1(b)). The angular deviations are modeled with the estimates $\sigma_\phi = \sigma_\theta = 0.15°$ (arctan of 0.25 mm error located 100 mm from the origin corresponding to uncertainty in finding the fiducial pins), and $\sigma_\psi = 0.25°$ (arctan of 2.0 mm error located 500 mm from the origin and corresponding to uncertainty in finding the femoral head). The resection *x-y* plane is assigned to a position 30 mm distal of the fiducial pins coordinate frame. Since the plan is "infinitely" precise with respect to the fiducial pin coordinate frame (we set its position), we assign no standard deviations to its position. The resultant covariance matrix is similar to that of Table 5.1, in that there are only a few off diagonal terms, and as such will not show it or the corresponding error ellipses here.

5.3 Completing Registration in the Operating Room

The setup of the robot, end effector, and the femur in the operating room is shown in Figure 5.4, and the registration graph in Figure 5.5. The coincidence link [END EFFECTOR : probe → FEMUR : fiducial pins is assigned a zero mean and standard deviations of 0.1mm and 0.1°. The position of the robot is assigned such that the femur is approximately such that the fiducial pins coordinate frame on the femur is approximately y = 500 mm, z = 250 mm, and θ = 135° from the base of the robot. (This is calculated with a particularly neat "trick": the [ROBOT : base → ROBOT : mounting flange] link is first assigned directly and a [ROBOT : base → FEMUR : fiducial pins] induced link is created. This gives us the necessary transformation vector to used to command the robot via a direct link to a particular position.) On the [ROBOT : base → ROBOT : mounting flange] direct link, standard deviations of 0.25 mm and 0.5° are assigned, corresponding to the calibrated accuracy of the robot.

The resultant covariance matrix on the [ROBOT : base → FEMUR : fiducial pins] is listed

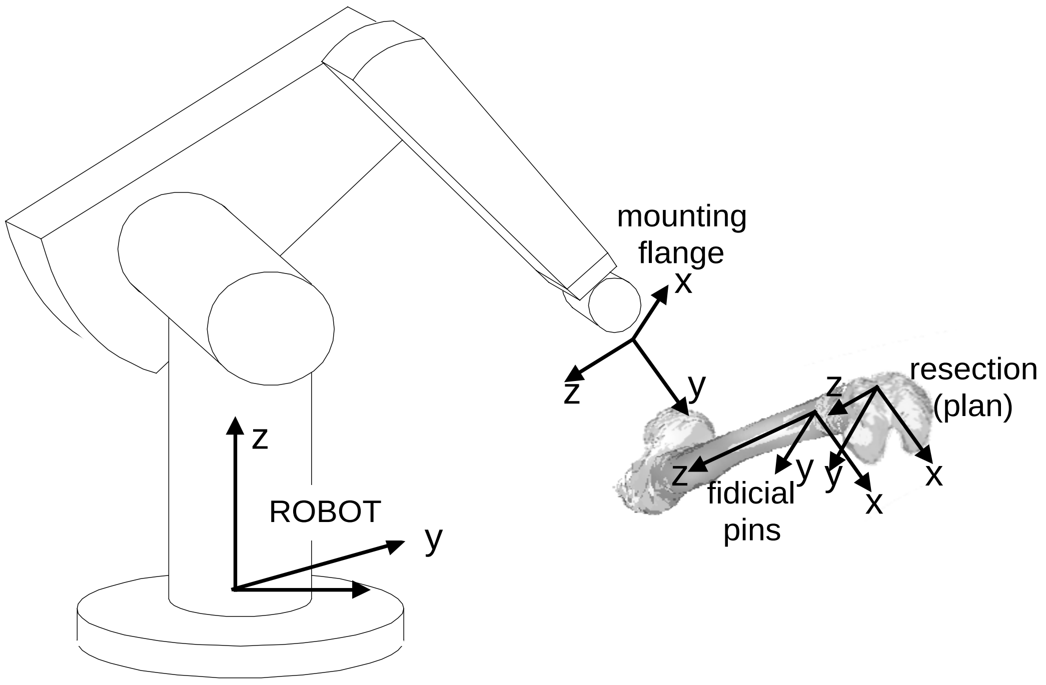


Figure 5.4: Positions of the coordinate frames of the robot, mounting flange, fiducial pins, and resection plan as configured in the operating room.

in Table 5.2. This shows us the final registration accuracy. Note that the variances concur with experimental results presented in [1][2]. The error ellipses are shown in Figure 5.6.

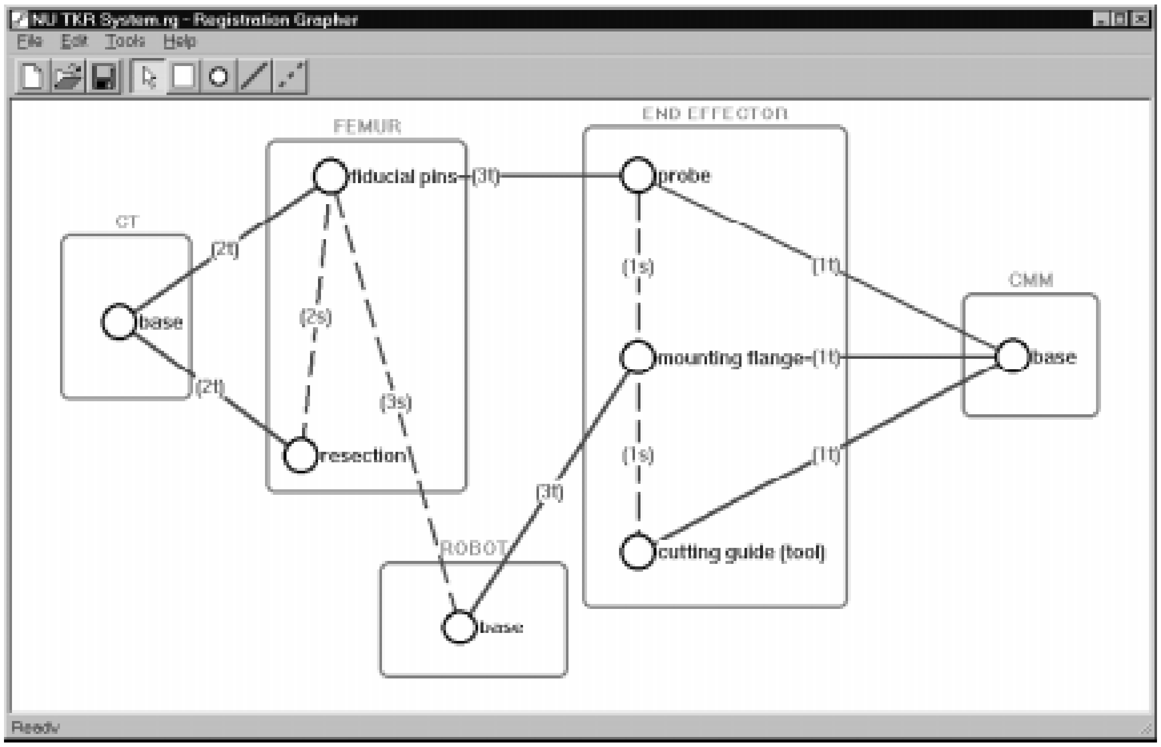


Figure 5.5: Registration graph of the full Northwestern TKR system.

$$\begin{bmatrix} 2.786 & -0.379 & 0.920 & -1.350 & 0.112 & -0.475 \\ -0.379 & 0.167 & -0.009 & 0.148 & 8.76\text{E-}05 & 0.005 \\ 0.920 & -0.009 & 1.089 & -0.696 & 0.128 & -0.513 \\ -1.350 & 0.148 & -0.696 & 0.830 & -1.10\text{E-}04 & 0.411 \\ 0.112 & 8.76\text{E-}05 & 0.128 & -1.10\text{E-}04 & 0.280 & -0.005 \\ -0.475 & 0.005 & -0.513 & 0.411 & -0.005 & 0.321 \end{bmatrix}$$

Table 5.2: Covariance matrix for the [ROBOT : base → FEMUR : fiducial pins] induced link

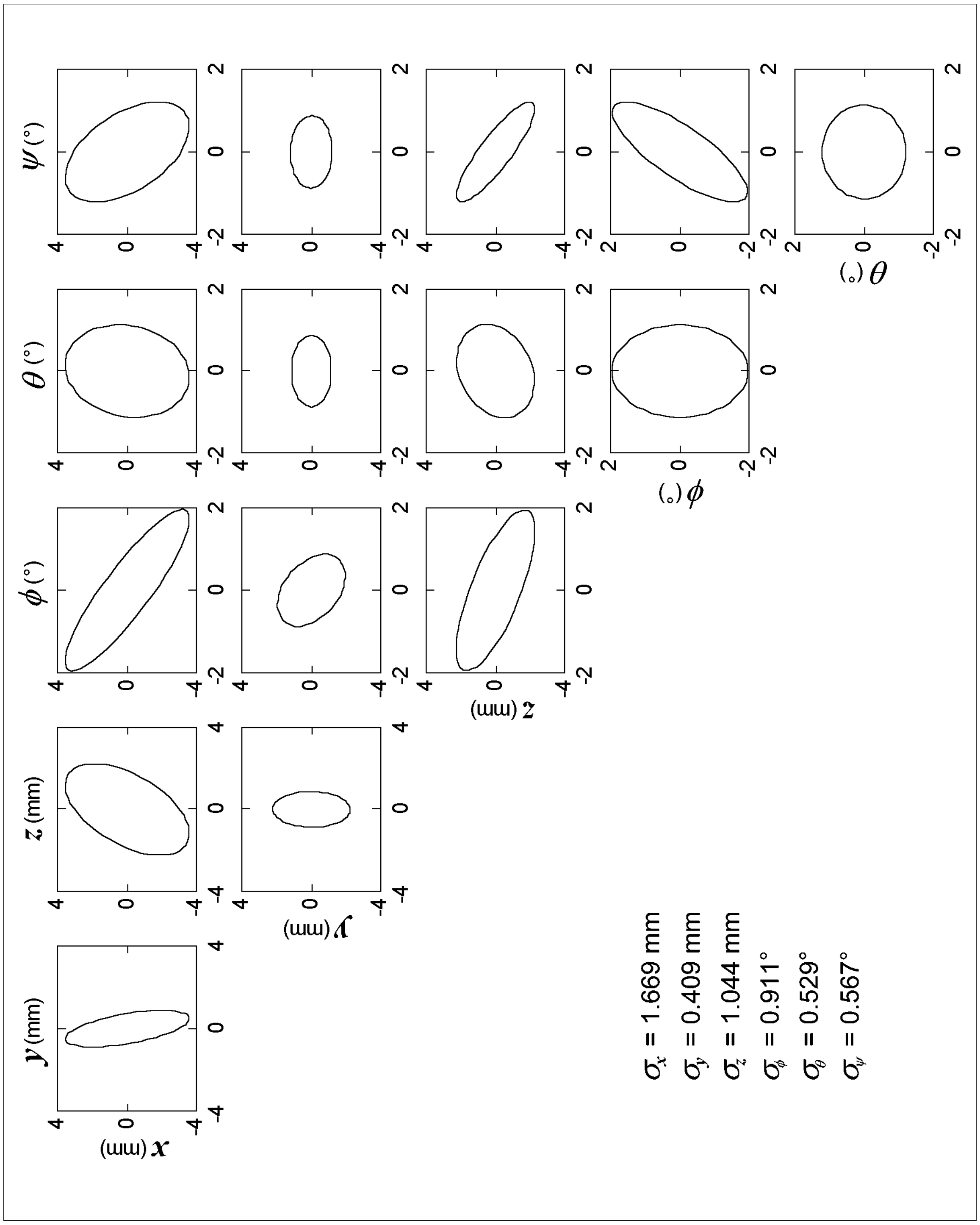


Figure 5.6: Error ellipses for the [ROBOT : base $\longrightarrow$ FEMUR : fiducial pins] induced link.

By adding an induced link between the FEMUR : resection and END EFFECTOR :cutting guide features, it is possible to ascertain the errors between the tool and plan. The graph with extra induced link is shown in Figure 5.7, the resultant covariance matrix is listed in Table 5.3, and the error ellipses are shown in Figure 5.8.

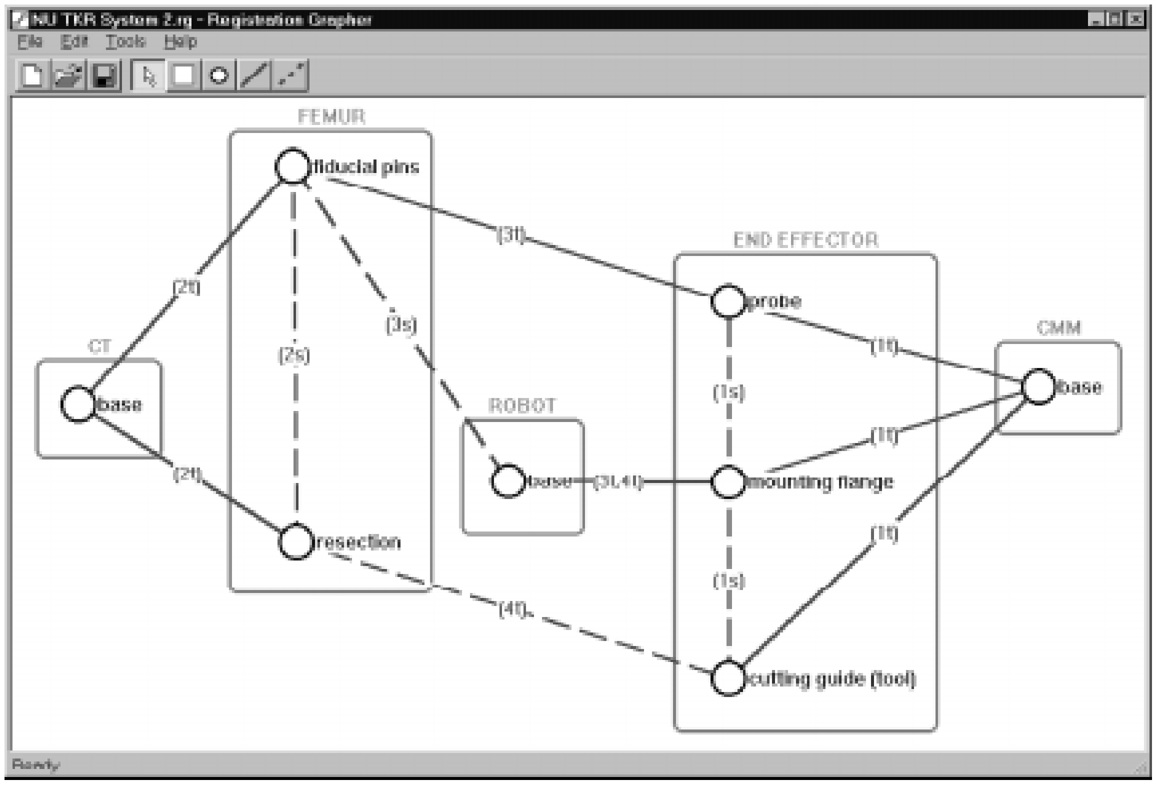


Figure 5.7: Registration graph showing the extra induced link between the FEMUR : resection and END EFFECTOR :cutting guide features.

1.547	0.007	-0.443	-0.947	0.007	-0.338
0.007	0.260	-0.066	-0.005	-8.22E-06	0.039
-0.443	-0.066	0.777	0.343	-1.03E-04	0.258
-0.947	-0.005	0.343	0.773	-8.73E-03	0.264
0.007	-8.22E-06	-1.03E-04	-8.73E-03	0.523	-1.66E-04
-0.338	0.039	0.258	0.264	-1.66E-04	0.322

Table 5.3: Covariance matrix for the [FEMUR : resection → END EFFECTOR :cutting guide] induced link

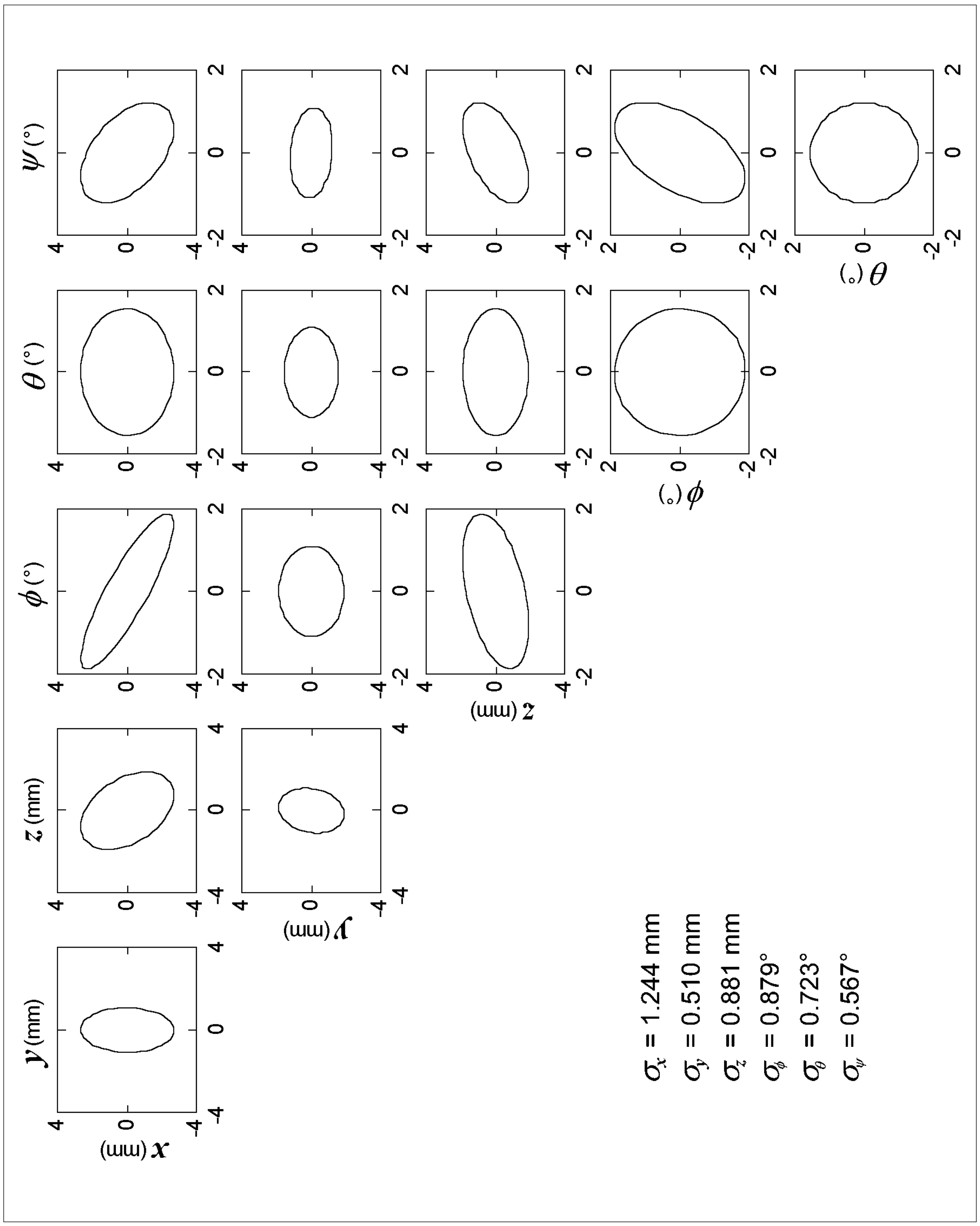


Figure 5.8: Error ellipses for the [ROBOT : base → FEMUR : fiducial pins] induced link.

6 Summary and Future Work

6.1 Research Contributions

The registration graph is a fundamentally new concept and contribution. In all of my investigation regarding how to represent aspects of registration graphically, the most similar constructs I encountered were data flow diagrams [12] and Petri nets [13]. However, these methods did not seem to properly illustrate the connectivity properties.

Writing the C++ library involved seemingly endless rewrites to make it both portable and extensible and to have a user-friendly interface - attributes of a good library. The code is mature enough to be used by others in the field. Potential users may wish to look to the world wide web home of the Laboratory for Intelligent Mechanical Systems - http://lims.mech.nwu.edu - as information regarding how to obtain the library are posted there. While the Registration Grapher program is a never ending work-in-progress, it too will be available. The C++ library and Registration Grapher program are useful tools for investigation of registration strategies. I hope that their existence will be of some benefit to the computer-assisted surgery field.

The uncertainty analysis was based primarily on the work of Smith and Cheeseman [38]. While other stronger uncertainty methods are available, it was chosen to prove the viability of the C++ library and fill the sibling role in the dichotomy of qualitative and quantitative analyses.

6.2 Future Work

Certainly, any piece of software is never completely finished. Since the library is written in C++, it will always be possible to "tweak it under the hood" while not affecting the users of the library. One near term goal is to integrate the drawing of error ellipses into the Registration Grapher program.

Other work lies in eliminating the problems associated with singularities in the Jacobians used for serial compounding and reversal of covariances. In [38], Smith and Cheeseman mention that they were investigating methods to eliminate this. One such method is to pre-rotate the coordinate frame associated with a feature such that the subsequent rotation is not in the vicinity of a Jacobian pole. Other methods undoubtedly exist.

A promising alternative method for parallel compounding is the Covariance Intersection Method developed by Julier [46]. Instead of using the Kalman equations to find an error ellipsoid that is contained completely within intersecting volume of the two covariances, the compounded covariance is instead a more natural fit through the intersecting curves on the two ellipsoids.

Finally, it may be possible to represent reduced information (infinite variances) with the inverse of the covariance matrix, the *information* matrix. This would potentially allow us to model the digitization of each point.

Appendix: User's Guide to the *Registration Grapher* Program

The Registration Grapher program is designed to be both simple and intuitive. Here, a brief overview of the program is presented that lists the commands of the program and their functions. A short tutorial is integrated into the guide that illustrates how the program is used to create a simple registration graph. The format for the registration graph file is also listed so one can extract the mean and covariance information for analysis using software tools such as MatLab.

The Main Window

Figure A.1 shows the main window of the Registration Grapher program. There are four areas of interest on the main window:

- *Menu Bar*: The menus list all commands available in the Registration Grapher program. The File menu contains commands for opening and saving registration graph files as well as the Exit command to quit the program. The Edit menu contains two commands: a Delete command to erase a highlighted graph object, and an Edit command to activate an object's dialog box. The Tools menu lists five tools used to create and edit objects. The Select Objects tool is used to highlight a graph object, move objects (rigid bodies and

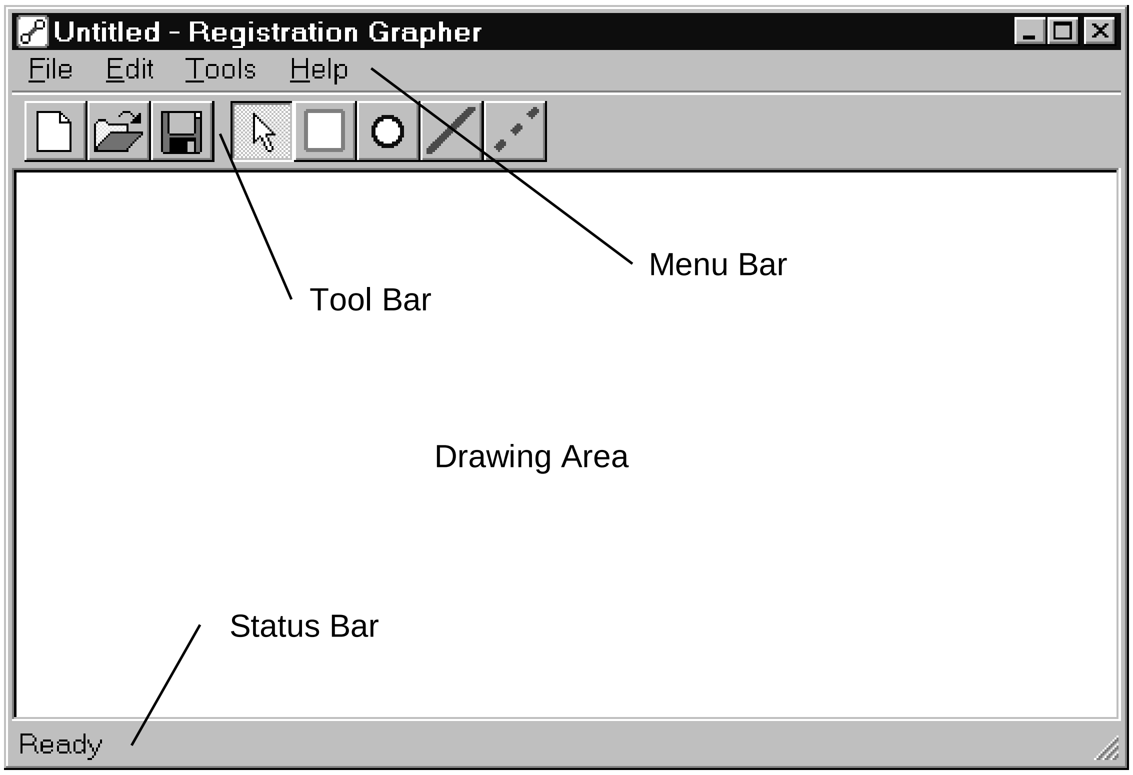


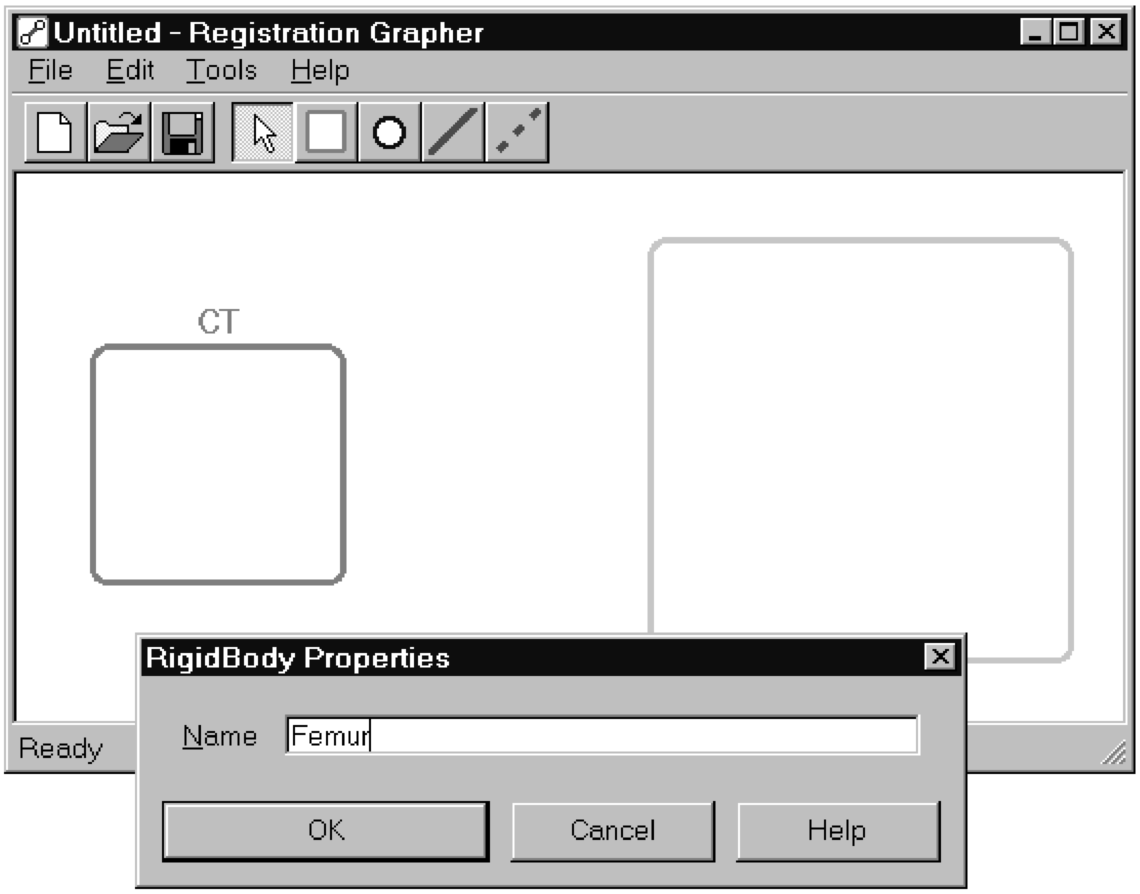
Figure A.1: Main window of the Registration Grapher program.

features), and change the size and shape of an object (rigid body objects only). The four remaining tools are used to add rigid bodies, features, and direct and induced links. The Help menu is mostly disabled as of this writing (this appendix is the "help file"), except for an About command used to list copyright and contact information.

- *Tool Bar*: The tool bar contains buttons that provide quick access to the file New / Open/ Save commands as well as the five tools listed under the Tools menu.
- *Drawing Area*: This is the drawing surface for the registration graph.
- *Status Bar*: Short explanations of commands are listed in the status bar area when the user moves the mouse over a command on the menus.

Adding Rigid Bodies

To add rigid bodies to the drawing area, select the Add Rigid Bodies tool from the Tools menu or the tool bar. Use the mouse cursor to first pick the upper left corner for the rigid body rectangle. Depress the left mouse button and drag the lower right corner of the rectangle to the desired position. Once the mouse button is removed, a dialog box appears in which the name of the rigid body can be edited. *Tutorial: Add two rigid bodies, "CT" and "Femur", as shown in Figure A.2*



Name: Femur

Figure A.2: Adding rigid bodies to the registration graph.

Adding Features

To add features to the rigid bodies, select the Add Features tool from the Tools menu or the tool bar. Use the mouse cursor to prescribe the location of the feature in a rigid body rectangle. While the left button is held, the feature can be moved around within the rectangle. Once the mouse button is removed, a dialog box appears in which the name of the feature can be edited. *Tutorial: Add the feature "base" to the CT rigid body and the features "fiducial pins" and "resection" to the Femur rigid body, as shown in Figure A.3*

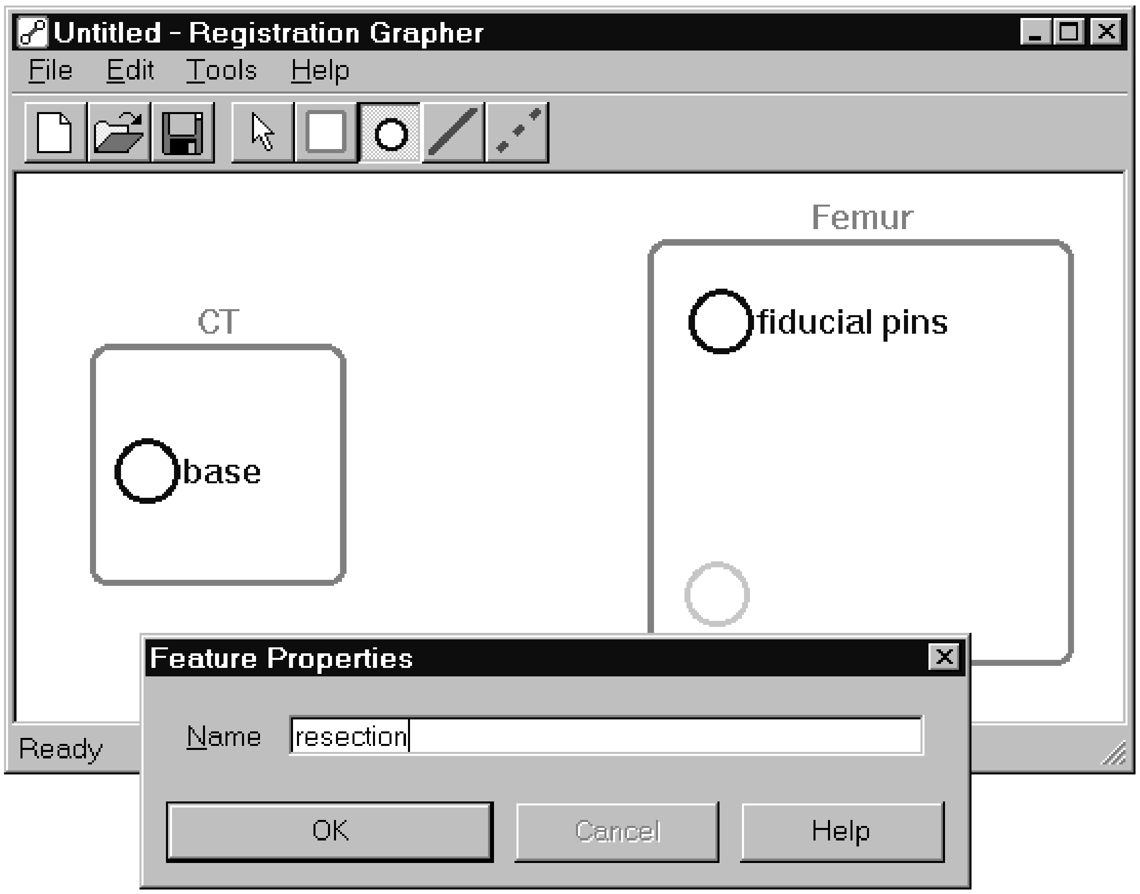


Figure A.3: Adding features to the rigid bodies.

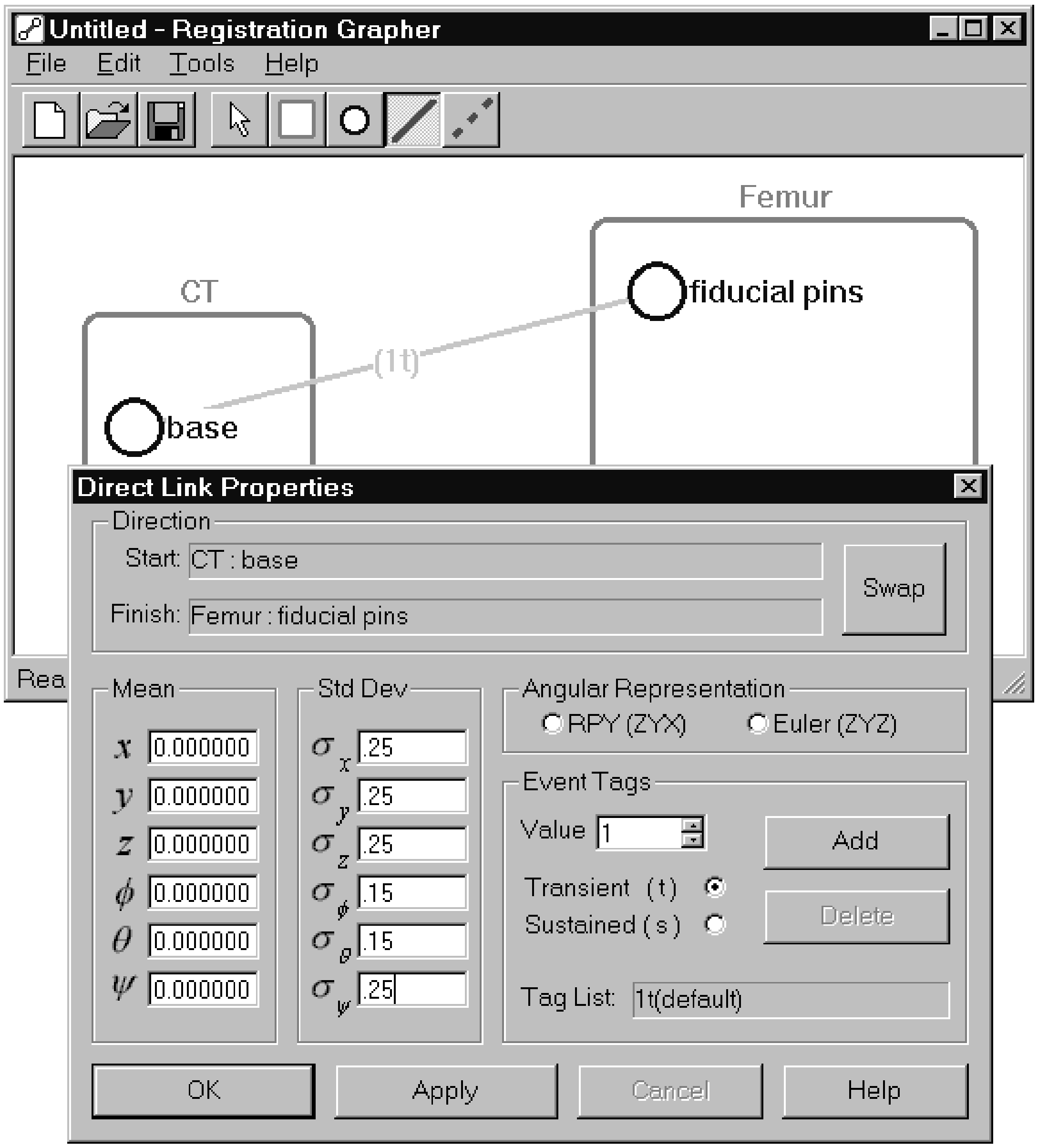
Adding Direct Links

To add direct links between features, select the Add Direct Link tool from the Tools menu or the tool bar. Use the mouse cursor to click the start feature of the link. While the left button is held, a "rubber band" appears and the endpoint can be moved around. When the mouse cursor is over the desired end feature, release the left button. Once the mouse button is removed, a dialog box appears on which the many of the links attributes can be changed.

The dialog box is used to change:

- *Link Direction*: Use the Swap button to interchange the start and finish features. The mean and standard deviation measurements will be defined with respect to the start feature's coordinate frame.
- *Mean*: Enter the mean measurements in a standard unit of length measure for x, y, and z, and in degrees for ϕ, θ, and ψ.
- *Std Dev*: Enter the standard deviation measurements in a standard unit of length measure for x, y, and z, and in degrees for ϕ, θ, and ψ.
- *Angular Representation*: Choose roll, pitch, yaw (RPY) or Euler representations for the three angles. (Note: These buttons currently have no effect. The program uses only a RPY representation.)
- *Event Tags*: Add or delete both transient ('t') or sustained ('s') event tags to the link. The list of tags is displayed in the Tag List field.

Tutorial: Add two direct links to the registration graph. The first starts at the CT : base feature and ends at the Femur : fiducial pins feature, and has mean = (0mm, 0mm, 0mm, 0°, 0°, 0°), std dev = (0.25mm, *0.25mm, 0.25mm, 0.15°, 0.15°, 0.25°), and a single event tag '1t'. The second link starts at the CT : base feature and ends at the Femur : resection feature, and has mean = (0mm, 0mm, -30mm, 0°, 0°, 0°), std dev = (0mm, 0mm, 0mm, 0°, 0°, 0°), and a single event tag '1t'. These are the same measurements used the case study in Chapter 5. See .Figure A.4*



Direction
Start: CT : base
Finish: Femur : fiducial pins

Mean
x 0.000000
y 0.000000
z 0.000000
ϕ 0.000000
θ 0.000000
ψ 0.000000

Std Dev
σ_x .25
σ_y .25
σ_z .25
σ_ϕ .15
σ_θ .15
σ_ψ .25

Angular Representation
RPY (ZYX) Euler (ZYZ)

Event Tags
Value 1
[✓ Transient (t)] Sustained (s)
Tag List: 1t(default)

Figure A.4: Adding direct links to the registration graph.

Drawing Induced Links

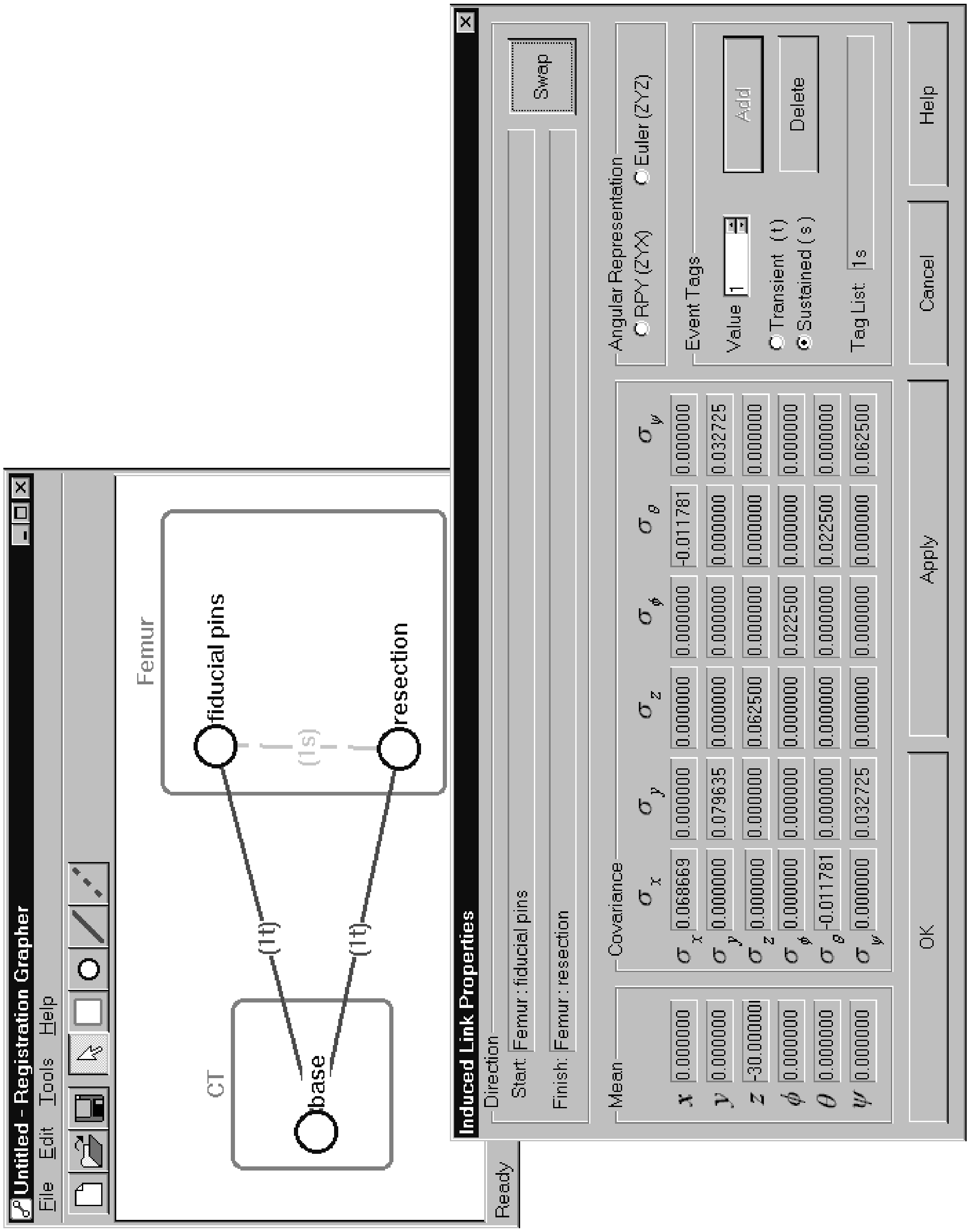
To add induced links between features, select the Add Induced Link tool from the Tools menu or the tool bar. Use the mouse cursor to click the start feature of the link. Induced links are drawn in the same manner as direct links. Once the mouse button is removed, a dialog box appears on which the many of the links attributes can be changed.

While the direct link dialog allows entry of means and standard deviations, the induced link dialog simply reports the means and covariance matrix calculated using the compounding methods described in Chapter 3. As with the direct link, the direction of the induced link can be changed, the angular representation can be selected (although disabled in current version), and event tags can be added and removed. *Tutorial: Add an induced link from the Femur : fiducial pins feature to the Femur : resection feature, with an event tag of '1s'. See Figure A.5*

Editing the Registration Graph

Once the graph is drawn, attributes of each object can be interactively changed. To work with existing objects, select the Select Objects tool from the Tools menu or the tool bar. Alternatively, the Select Objects tool can be selected by clicking on and empty part of the drawing area.

An object is highlighted simply by clicking on it. To access the objects dialog box, either select the Edit Object Attributes command from the Edit menu, or double click the object. An object can be removed from the graph by first highlighting it and then selecting Delete Object from the Edit menu, or by pressing the Delete key.



Direction
Start: Femur : fiducial pins
Finish: Femur : resection

Mean

x	0.000000
y	0.000000
z	-30.00000
ϕ	0.000000
θ	0.000000
ψ	0.000000

Covariance

	σ_x	σ_y	σ_z	σ_ϕ	σ_θ	σ_ψ
σ_x	0.068669	0.000000	0.000000	0.000000	-0.011781	0.000000
σ_y	0.000000	0.079635	0.000000	0.000000	0.000000	0.032725
σ_z	0.000000	0.000000	0.062500	0.000000	0.000000	0.000000
σ_ϕ	0.000000	0.000000	0.000000	0.022500	0.000000	0.000000
σ_θ	-0.011781	0.000000	0.000000	0.000000	0.022500	0.000000
σ_ψ	0.000000	0.032725	0.000000	0.000000	0.000000	0.062500

Angular Representation
RPY (ZYX) Euler (ZYZ)

Event Tags
Value 1
Transient (t) [✓ Sustained (s)]
Tag List: 1s

Figure A.5 Adding an induced link to the registration graph.

Registration Graph File Format

To study the ellipses obtained from the covariance matrix, it is necessary to extract the covariances from the registration graph file. The file is stored as plain ASCII. The class library contains functions that open a graph file and create a registration graph by adding objects to a `RGGraph` object based on the information parsed from the file. The file is easily opened with any program that can read text files. While the file can be edited as ASCII, it is recommended that changes be made solely with the Registration Grapher program.

The file first contains all of the graph's rigid bodies and features. The file listing alternates between a rigid body description and a set of feature descriptions. Features belong to the rigid body that immediately precedes it. After all rigid bodies and features are described, a similar technique is used to describe links and event tags. The file listing alternates between a link description and a set of event tag descriptions, where the event tags belong to the link immediately preceding. What follows is the file listing for the graph constructed in this chapter. Inspection of the file reveals that it is relatively easy to read.

```
[RigidBody]
Name=CT
TopLeft=37,84
BottomRight=160,200
[Feature]
Name=base
Center=64,145

[RigidBody]
Name=Femur
TopLeft=309,33
BottomRight=515,239
[Feature]
Name=fiducial pins
Center=344,72
[Feature]
Name=resection
Center=342,205
```

```
[Link]
Type=Direct
Start=CT:base
Finish=Femur:fiducial pins
Mean=        0.0     0.0     0.0     0.0     0.0     0.0
Covariance= 0.0625  0.0     0.0     0.0     0.0     0.0
            0.0     0.0625  0.0     0.0     0.0     0.0
            0.0     0.0     0.0625  0.0     0.0     0.0
            0.0     0.0     0.0     0.0225  0.0     0.0
            0.0     0.0     0.0     0.0     0.0225  0.0
            0.0     0.0     0.0     0.0     0.0     0.0625

[Link]
Type=Direct
Start=CT:base
Finish=Femur:resection
Mean=        0.0     0.0   -30.0     0.0     0.0     0.0
Covariance= 0.0     0.0     0.0     0.0     0.0     0.0
            0.0     0.0     0.0     0.0     0.0     0.0
            0.0     0.0     0.0     0.0     0.0     0.0
            0.0     0.0     0.0     0.0     0.0     0.0
            0.0     0.0     0.0     0.0     0.0     0.0
            0.0     0.0     0.0     0.0     0.0     0.0

[Link]
Type=Induced
Start=Femur:fiducial pins
Finish=Femur:resection
Path=Femur:fiducial pins,CT:base,Femur:resection
Mean=        0.0     0.0   -30.0     0.0     0.0     0.0
Covariance= 0.0687  0.0     0.0     0.0    -0.0118  0.0
            0.0     0.0796  0.0     0.0     0.0     0.0327
            0.0     0.0     0.0625  0.0     0.0     0.0
            0.0     0.0     0.0     0.0225  0.0     0.0
           -0.0118  0.0     0.0     0.0     0.0225  0.0
            0.0     0.0327  0.0     0.0     0.0     0.0625
[EventTag]
Type=Sustained
Value=1

[End]
```

The covariances for the induced link can be easily cut and pasted to new files for further analysis in a program such as MatLab. The ellipse parameter equations of Section 3.4 can be employed to derive the error ellipses. A future version of the Registration Grapher program may include error ellipse rendering.

Bibliography

[1] T. C. Kienzle III, S. D. Stulberg, M. Peshkin, A. Quaid, J. Lea, A. Goswami, C. H. Wu. 'A Computer-Assisted Total Knee Replacement Surgical System Using a Calibrated Robot.' *Computer Assisted Surgery*, R. H. Taylor, S. Lavallée, G. Burdea, R. Mösges, eds., MIT Press, Cambridge, Mass., 1995.

[2] J. T. Lea, D. Watkins, A. Mills, M. A. Peshkin, T. C. Kienzle III, S. D. Stulberg. 'Registration and Immobilization for Robot-Assisted Orthopaedic Surgery.' *Journal of Image Guided Surgery*, 1(2) pp. 80-87, Wiley-Liss, Inc., 1995.

[3] R. H. Taylor, B. D. Mittelstadt, H. A. Paul, W. Hanson, P. Kazanzides, J. F. Zuhars, B. Williamson, B. L. Musits, E. Glassman, W. L. Bargar. 'An Image-Directed Robotic System for Precise Orthopaedic Surgery.' *IEEE Transactions on Robotics and Automation*, 10(3):261-275.

[4] P. Kazanzides, J. Zuhars, B. Mittelstadt, B. Williamson, P. Cain, F. Smith, L. Rose, B. Musits. 'Architecture of a Surgical Robot.' *Proceedings of the IEEE Conference on Systems, Man, and Cybernetics*, Chicago, IL, October 1992.

[5] H.A. Paul, W.L. Bargar, B. Mittelstadt, P. Kazanzides, B. Musits, J. Zuhars, P. W. Cain, B. Williamson, F. G. Smith. 'Robotic Execution of a Surgical Plan.' *Proceedings of the IEEE Conference on Systems, Man, and Cybernetics*, Chicago, IL, October 1992.

[6] P. Kazanzides, J. Zuhars, B. Mittelstadt, R. Taylor. 'Force Sensing and Control for a Surgical Robot.' *Proceedings of the IEEE Conference on Robotics and Automation*, Nice, France, May 1992.

[7] R. D. Bucholz, K. R. Smith, J. Henderson, L. McDurmont. 'Intraoperative Localization using a Three Dimensional Optical Localizer.' *Proceedings of the First International Symposium on Medical Robotics and Computer Assisted Surgery*, 283-290, Pittsburgh, PA, September, 1994.

[8] *Computer Assisted Surgery*, R. H. Taylor, S. Lavallée, G. Burdea, R. Mösges, eds., MIT Press, Cambridge, Mass., 1995.

[9] B. Preising, T.C. Hsia, B. Mittelstadt. 'A Literature Review: Robots in Medicine.' *IEEE Engineering in Medicine and Biology*, 10(2) 13-22, 1991.

[10] G. Chartrand. *Graphs as Mathematical Models.* Prindle, Weber & Schmidt, Inc., Boston, MA, 1977.

[11] R. P. Paul. *Robot Manipulators: Mathematics, Programming and Control.* MIT Press, Cambridge, Mass., 1981.

[12] T. DeMarco. *Structured Analysis and System Specification.* Yourdon, Inc., New York, NY, 1979.

[13] F. Dicesare, G. Harhalakis, J. M. Proth, M. Silva, F. B. Vernadat. *Practice of Petri Nets in Manufacturing.* Chapman & Hall, London, 1993.

[14] M. Fadda, T. Wang, M. Marcacci, S. Martelli, P. Dario, G. Marcenaro, M. Nanetti, C. Paggetti, A. Visani, S. Zaffagnini. 'Computer-Assisted Knee Arthroplasty at Rizzoli Institutes.' *Proceedings of the First International Symposium on Medical Robotics and Computer Assisted Surgery*, 26-30, Pittsburgh, PA, September, 1994.

[15] J. L. Moctezuma, F. Gosse, H. J. Schultz. 'A Computer and Robotic Aided Surgery System for Accomplishing Osteotomies.' *Proceedings of the First International Symposium on Medical Robotics and Computer Assisted Surgery*, 31-35, Pittsburgh, PA, September, 1994.

[16] S. Lavallée, P. Sautot, J. Troccaz, P. Cinquin, P. Merloz. 'Computer Assisted Spine Surgery: A Technique for Accurate Transpedicular Screw Fixation Using CT Data and

a 3D Optical Localizer.' *Proceedings of the First International Symposium on Medical Robotics and Computer Assisted Surgery*, 315-322, Pittsburgh, PA, September, 1994.

[17] L. P. Nolte, L. J. Zamorano, Z. Jiang, Q. Wang, F. Langlotz, E. Arm, H. Visarius. 'A Novel Approach to Computer Assisted Spine Surgery.' *Proceedings of the First International Symposium on Medical Robotics and Computer Assisted Surgery*, 323-328, Pittsburgh, PA, September, 1994.

[18] Y. S. Kwoh, I. S. Reed, J. Y. Chen, H. M. Shao, T. K. Truong, E. Jonckheere. 'A New Computerized Tomographic-Aided Robotic Stereotaxis System.' *Robotics Age*, 17-22, June 1985.

[19] D. Glauser, P. Flury, N. Villotte, C.W. Burckhardt. 'Conception of a Robot Dedicated to Neurosurgical Operations.' *Proceedings of the 5th International Conference on Advanced Robotics*, 899-904, Pisa, Italy, 1991.

[20] J. M. Drake, M. Joy, A. Goldenberg, D. Kreindler. 'Computer and Robotic Assisted Resection of Brain Tumors.' *Proceedings of the 5th International Conference on Advanced Robotics*, 888-892, Pisa, Italy, 1991.

[21] J. J. Santos-Munné, M. A. Peshkin, S. Mirkovic, S. D. Stulberg, T. C. Kienzle. 'A Stereotactic and Robotic System for Pedicle Screw Placement.' *Interactive Technology and the New Paradigm for Healthcare*, K. Morgan, R. M. Satava, H. B. Sieburg, R. Mattheus, J. P. Christensen, eds., 326-333, IOS Press and Ohmsha, 1995.

[22] J. J. Santos-Munné. 'Stereotactic System for Linear-Trajectory Medical Interventions.' unpublished master's thesis, Mechanical Engineering Department, Northwestern University, 1996.

[23] J. Troccaz, S. Lavallée, P. Sautot, P. Cinquin, B. Mazier, J. P. Chirossel. 'Robot Assisted Spine Surgery.' *Medi-MECHATRONICS Workshop*, Málaga, Spain, October, 1992.

[24] S. Lavallée. 'Image Guided Operating Robot: A Clinical Application in Stereotactic Neurosurgery.' *Proceedings of the IEEE Conference on Robotics and Automation*, 618-624, Nice, France, May 1992.

[25] P. Potamianos, B. L. Davies, R. D. Hibbert. 'Intraoperative Imaging Guidance for Keyhole Surgery.' *Proceedings of the First International Symposium on Medical Robotics and Computer Assisted Surgery*, 98-105, Pittsburgh, PA, September, 1994.

[26] D. A. Simon. 'Fast and Accurate Shape-Based Registration.' Ph.D. Dissertation, Technical Report, CMU-RI-TR-96-45, Carnegie Mellon University, Robotics Institute, 1996.

[27] D. A. Simon, M. Hebert, T. Kanade. 'Techniques for Fast and Accurate Intrasurgical Registration.' *Journal of Image Guided Surgery*, 1:17-29, 1995.

[28] R. E. Ellis, S. Toksvig-Larsen, M. Marcacci, D. Caramella, M. Fadda. 'A Biocompatible Fiducial Marker for Evaluating the Accuracy of CT Image Registration.' *Computer-Assisted Radiology* (1996), Elsevier International Congress Series #1124, pp. 693--698.

[29] W. E. L. Grimson, G. J. Ettinger, S. J. White, P. L. Gleason, T. Lozano-Perez, W. M. Wells, R. Kikinis. 'Evaluating and Validating an Automated Registration System for Enhanced Reality Visualization in Surgery.' *Proceedings of the First International Conference, CVRMed '95*, n. Ayache (ed.), Springer-Verlag, Nice, France, April, 1995.

[30] D. A. Simon, R. V. O'Toole, M. Blackwell, F. Morgan, A. M. DiGioia, T. Kanade. 'Accuracy Validation in Image-Guided Orthopaedic Surgery.' *Proceedings of the Second Annual International Symposium on Medical Robotics and Computer Assisted Surgery*, 185-192, Baltimore, MD, November, 1995.

[31] N. D. Glossop, R. W. Hu. 'Practical Accuracy Assessment of Image Guided Spine Surgery.' *Proceedings of the Second Annual North American Program on Computer Assisted Orthopaedic Surgery*, 96-98, Pittsburgh, PA, 1998.

[32] R. H. Taylor. 'A Synthesis of Manipulator Control Programs from Task-Level Specifications.' Ph.D. dissertation, Technical Report AIM-282, Stanford University, Stanford University Artificial Intelligence Laboratory, 1976.

[33] R. H. Taylor, V. T. Rajan. 'The Efficient Computation of Uncertainty Spaces for Sensor-based Robot Planning.' Research Report RC 13998 (#62814), IBM Research Division, 1988.

[34] R. A. Brooks. 'Symbolic Error Analysis and Robot Planning.' *International Journal of Robotics Research*, 1(4): 29-68, 1982.

[35] R. A. Brooks. 'Visual Map Making for a Mobile Robot.' *Proceedings of the IEEE Conference on Robotics and Automation*, 824-829, 1985.

[36] R. V. Hogg, J. Ledolter. *Engineering Statistics*, MacMillan Publishing Company, New York and Collier Macmillan Publishers, London, 1987.

[37] H. F. Durrant-Whyte. 'Consistent Integration and Propagation of Disparate Sensor Observations.' *International Journal of Robotics Research*, 6(3): 3-24, 1987.

[38] R. Smith, P. Cheeseman. 'On the Representation and Estimation of Spatial Uncertainty.' *International Journal of Robotics Research*, 5(4): 56-68, 1986.

[39] R. E. Kalman. 'A New Approach to Linear Filtering and Prediction Problems.' *Transactions of the ASME Journal of Basic Engineering*, 35-45, March 1960.

[40] G. Welch, G. Bishop. 'An Introduction to the Kalman Filter.' University of North Carolina, Department of Computer Science, TR 95-041, 1995.

[41] B. W. Kernigan, D. M. Ritchie. 'The C Programming Language: ANSI C Version.' *Prentice-Hall*, 1988.

[42] B. Stroustrup. 'The C++ Programming Language.' *Addison-Wesley*, 1997.

[43] A. Stepanov, M. Lee. 'The Standard Template Library.' Technical, Hewlett-Packard, 1995. (available online at http://www.cs.rpi.edu/~musser/stl.html)

[44] M. Nelson. *C++ Programmer's Guide to the Standard Template Library*, IDG Books Worldwide, Inc. Foster City, CA, 1995.

[45] P. J. Besl, N. D. McKay, 1992. 'A Method for Registration of 3-D shapes.' *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239-256.

[46] S. J. Julier, J. K. Uhlmann, H. F. Durrant-Whyte. 'A New Approach for Filtering Nonlinear Systems.' *Proceedings of the American Control Conference*, Seattle, WA, 1628-1632, 1995.