Int J CARS (2008) 3:395–403
DOI 10.1007/s11548-008-0243-4

ORIGINAL ARTICLE

Designing tracking software for image-guided surgery applications: IGSTK experience

Andinet Enquobahrie · David Gobbi · Matthew W. Turek · Patrick Cheng · Ziv Yaniv · Frank Lindseth · Kevin Cleary

Received: 10 January 2008 / Accepted: 4 June 2008 / Published online: 4 July 2008

Abstract
Objective Many image-guided surgery applications require tracking devices as part of their core functionality. The Image-Guided Surgery Toolkit (IGSTK) was designed and developed to interface tracking devices with software applications incorporating medical images.
Methods IGSTK was designed as an open source C++ library that provides the basic components needed for fast prototyping and development of image-guided surgery applications. This library follows a component-based architecture with several components designed for specific sets of image-guided surgery functions. At the core of the toolkit is the tracker component that handles communication between a control computer and navigation device to gather pose measurements of surgical instruments present in the surgical scene. The representations of the tracked instruments are superimposed on anatomical images to provide visual feedback to the clinician during surgical procedures.
Results The initial version of the IGSTK toolkit has been released in the public domain and several trackers are supported. The toolkit and related information are available at http://www.igstk.org.
Conclusion With the increased popularity of minimally invasive procedures in health care, several tracking devices have been developed for medical applications. Designing and implementing high-quality and safe software to handle these different types of trackers in a common framework is a challenging task. It requires establishing key software design principles that emphasize abstraction, extensibility, reusability, fault-tolerance, and portability. IGSTK is an open source library that satisfies these needs for the image-guided surgery community.

Keywords Trackers · Open source software · Image-guided surgery · Software design principles · State machines

A. Enquobahrie (✉) · M. W. Turek
Kitware Inc., 28 Corporate Drive, Clifton Park, NY 12065, USA
e-mail: andinet.enqu@kitware.com

M. Turek
e-mail: matt.turek@kitware.com

D. Gobbi
School of Computing, Queen's University, Kingston, ON K7L 3N6, Canada
e-mail: dgobbi@cs.queensu.ca

P. Cheng · Z. Yaniv · K. Cleary
Imaging Science and Information Systems (ISIS) Center, Department of Radiology, Georgetown University Medical Center, Washington, DC 20007, USA
e-mail: cheng@isis.georgetown.edu

Z. Yaniv
e-mail: yaniv@isis.georgetown.edu

K. Cleary
e-mail: cleary@isis.georgetown.edu

F. Lindseth
SINTEF Health Research and The National Center for 3D Ultrasound in Surgery, Trondheim, Norway
e-mail: Frank.Lindseth@sintef.no

Introduction

Image-Guided Surgery Toolkit (IGSTK) overview

Image-Guided Surgery Toolkit is an open source C++ library designed for fast prototyping and development of image-guided surgery applications [1,2]. The toolkit is being developed with support from the National Institute of Biomedical

Imaging and Bioengineering (NIBIB) at the National Institute of Health (NIH). Both industry and academic partners have contributed to the toolkit. The toolkit provides the basic components required to build image-guided surgery applications. The initial version of the toolkit was released in February 2006 at the SPIE Medical Imaging conference at San Diego. Since then, applications built using the toolkit have been demonstrated at various scientific conferences (SPIE 2006, 2007, 2008 and SMIT 2007). Furthermore, an FDA approved single center clinical trial for electromagnetically tracked lung biopsy based on IGSTK has begun at Georgetown University Medical Center (Washington, DC, USA).

Since an active user community is essential for the continued success of an open source toolkit, the IGSTK developers have been proactive in expanding the user base. A mailing list is constantly monitored to provide user support. The list has led to extensive and productive discussions aimed at improving the utility of the toolkit for fast prototyping and development of surgical applications. Discussions from the mailing list have resulted in new collaborations and have contributed to improvements in the components. The developers' team attempts to respond to users' requests by fixing bugs and implementing essential new functionality in a timely fashion. Currently, new developments include the integration of automatic and manual-based image reslicing, the extension of tracker support to additional trackers, and the implementation of a video grabber component. The initial developments of these components are archived in the IGSTK sandbox (test-bed repository) to allow users to follow new developments and provide early feedback.

IGSTK architecture

IGSTK follows a component-based architecture. The toolkit contains several components with a well defined set of behaviors governed by state machines. The state machine ensures that each component is always in a deterministic state and all state transitions are valid and meaningful. State machines were included from the beginning as an integral part of the toolkit design and intended to produce a safe and reliable software library suitable for safety critical applications.

By analyzing typical clinical applications, the required components for image-guided surgery applications were identified. Figure 1 shows a UML collaboration diagram of the main components in IGSTK. The main components include *View*, *Spatial Objects*, *Spatial Object Representations*, *Image Readers* and *Tracker* components.

The *View* component is used to display the graphical representations of surgical scenes. This component provides visual feedback to clinicians to assist them with instrument placement during image-guided procedures. For GUI-based application development, the view component is linked with GUI libraries using widget classes. IGSTK provides widgets for Qt and FLTK GUI libraries.

The *Spatial Objects* component defines a common structure for geometrical objects. Different spatial objects that define the shape and the physical characteristics of typical anatomical structures and surgical devices are provided in the toolkit. The graphical representations of spatial objects are characterized using the *Spatial Object Representations* component. This component describes properties such as color and surface properties.

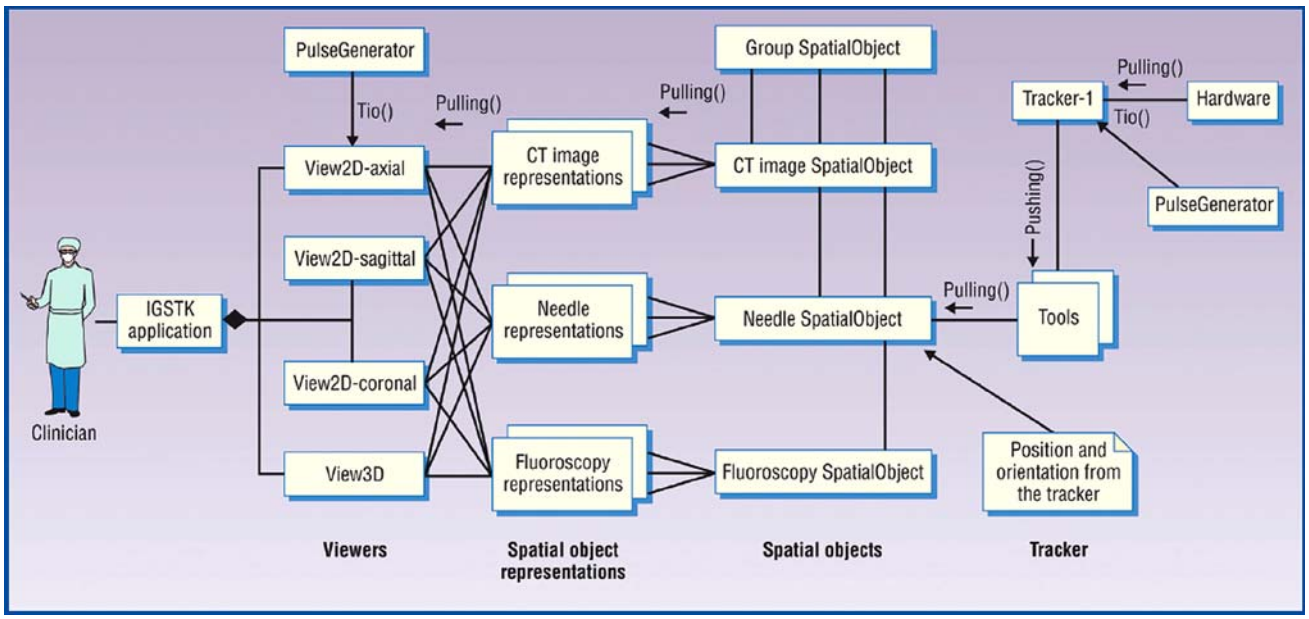

Fig. 1 IGSTK component architecture

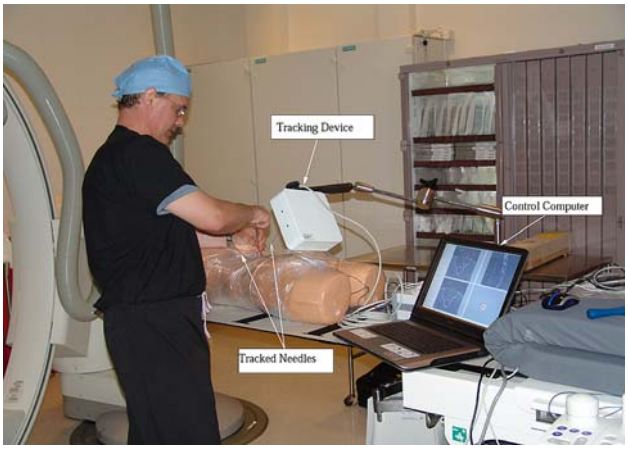

Fig. 2 Typical image-guided system using IGSTK: vertebroplasty spinal therapy using an electromagnetic tracking device

The *Image Reader* component loads data into the scene generation and representation process. DICOM image and mesh data readers are available in IGSTK.

The *Tracker* component, which is the main focus of this paper, handles communication between the control computer and the tracking devices to gather pose information from surgical instruments present in the scene. IGSTK provides interfaces for several commonly used tracking devices.

In addition to the above main components, IGSTK provides infrastructure and service classes such as state machine, loggers, pulse generators, registration, and calibration classes. All the components are implemented using the C++ programming language. The components also make extensive use of classes from the Visualization Toolkit (VTK) [3] and Insight Toolkit (ITK) [4] open source toolkits.

Time management and synchronization is an integral part of the architectural design. It is critical to ensure that the positions and the orientations of objects in the view remain consistent with the current relationships in the surgical scene. For this purpose, pulse generators are used to synchronize time across the components. As shown in Fig. 1, the tracker and view classes contain their own pulse generators. The pulse generators can be set to different frequencies depending on the requirements of the application. The frequency of the view pulse generator determines the view refresh rate. The frequency of the tracker pulse generator determines how fast tracking information is read from the tracking device. At every pulse, during the tracking cycle, the tracker component queries the tracking device to gather position and orientation information of surgical instruments (tracker tools). Tracking information of multiple tools is independently gathered as shown in Fig. 1. Transform objects are time-stamped to indicate the start and the expiration times of the transform. The view component compares the expiration and the start times of the tool transform and the refresh render time to turn on or off the visibility of the tool representation in the view.

Tracker overview

An image-guided surgery system consists of a control computer; software for image processing, registration and visualization; and a device for tracking surgical instruments and anatomical structures. A typical image-guided system based on IGSTK is shown in Fig. 2. In this experimental setup, an image-guided vertebroplasty spinal therapy was performed on a phantom model using an electromagnetic tracking device [5]. The tracking device reports position and orientation information from the surgical tools and anatomical structures. Various tracking devices based on different working principles and technologies are available for surgical use [6]. The three main types are mechanical, optical, and electromagnetic trackers as shown in Fig. 3.

Fig. 3 Tracking systems. **a** Mechanical tracker (photo courtesy of Robert Galloway, Vanderbilt University), **b** optical tracker: Optotrak Certus (Northern Digital Inc., Waterloo, ON, Canada), **c** electromagnetic tracker: Aurora electromagnetic tracker (Northern Digital Inc. Waterloo, ON, Canada)

A typical mechanical tracker uses a probe that is linked to articulated (multi-jointed) arms. Mechanical trackers are highly accurate and stable, but they are bulky and their utility is limited to specific interventional procedures where they will not interfere with the surgical procedure. Furthermore, mechanical trackers can only track a single object in the surgical scene. Therefore, mechanical trackers are typically not used today in image-guided systems. The flexibility and performance of optical and electromagnetic trackers have made them the trackers of choice for current image-guided surgery systems.

Optical trackers use cameras to track fiducial markers that are attached to the surgical instrument. These trackers use triangulation techniques to determine the position and orientation of the instruments. The cameras could be infrared-based (Polaris trackers from NDI), video-based (MicronTracker from Claron Technology) or laser-based (laserBird2 from Ascension Technology). These systems provide high frequency refresh rates (on the order of 30 Hz) making them useful for medical procedures. Unlike mechanical trackers, optical trackers can track multiple objects. Optical trackers use either active or passive markers (such as retro-reflective infrared markers for Polaris trackers and high intensity contrast markers for the MicronTracker). In general, optical trackers are accurate and have a large field of measurement. However, the tracking device has to be in a line-of-sight with the object to be tracked. This makes them unusable for tracking flexible objects inside the body.

A line-of-sight between the tracking device and tracked objects is not required for electromagnetic trackers. Electromagnetic trackers use an electromagnetic field generator and small electromagnetic coils that can be embedded in surgical instruments. Similar to optical trackers, multiple objects can be tracked concurrently. The main drawbacks of electromagnetic trackers are (1) the surgical environment must be devoid of any ferromagnetic material that can interfere with the electromagnetic field and degrade the measurement accuracy and (2) instruments to be tracked must be modified to include sensor coils.

Ultimately, the choice of tracking device depends on the requirements of the medical procedure and availability of tracked tools. Major factors that need to be considered include the line-of-sight requirements, update rate, number of tools that need to be tracked, size of the measurement volume, accuracy, cost, and conditions in the surgical environment.

Tracker software design

Designing high quality and safe tracking software that handles different types of tracking devices in a common framework is a challenging task. It requires establishing appropriate software design principles [7]. The software design is driven by the high level functions of the tracking software. Tracking software performs the following main functions:

1. Communicate with a tracking device.
2. Configure and initialize tracking tools.
3. Gather and store pose measurements for each tracking tool in the surgical scene.
4. Convert measurements into internal data representation for use by other components in an image-guided system.

Software requirements

An analysis of tracking needs in several image-guided applications has led us to develop the following ten major requirements for tracking software.

1. *Tracking device abstraction.* A variety of tracking devices based on different working principles and technologies are available for clinical use. Recurring operations common among tracking devices should be abstracted and implemented in generic classes. Furthermore, device specific intricacies should be delegated to concrete subclasses.
2. *Tracking tool abstraction.* Tracking tools can be wired, wireless, active, or passive. Different tracking devices support one or more of these types of tools. For fast and easy application development, the common parameters and behaviors of these tools should be abstracted.
3. *Extensibility.* With new tracking devices continuously being developed, tracking software should be easily extensible to handle new devices.
4. *Portability.* For wide usability and applicability, tracking software should be portable across compilers and platforms. Cross-platform configuration and build technologies should therefore be used.
5. *Decomposition.* Complexity should be handled by decomposing large problems into smaller ones. For example, the communication interface with a tracking device should be implemented separately from measurement data buffering and transmission to other components.
6. *Reduced latency.* Surgical applications require timely measurements of the positions of surgical instruments. Hence, communication with the tracking device and data collection processes should not add too much overhead to the overall system.
7. *Fault-tolerance.* Several hazardous conditions related to tracking could occur during a medical procedure. For example, a loss of communication with the tracker could occur if a tracking tool cable or tracking device cable is disconnected. Power interruption could also occur in the main control computer. The tracking

software should be able to detect and compensate for software faults.

8. *Transparency.* The software design should allow users to record and replay tracking measurements for testing and validation purposes. This design is also useful for setting up virtual environments to train clinicians in surgical procedures.
9. *Deterministic behavior.* The most critical requirement of image-guided surgery software should be patient safety. To ensure patient safety, the software must have deterministic behavior during runtime. Software methodologies that guarantee determinism must be utilized.
10. *Time synchronization and dynamic objects handling.* Position measurements of surgical instruments are time-specific. The measurements should be stored in a dynamic object that has a notion of time validity. Time synchronization techniques should be used to synchronize time-dependent tasks such as scene generation and position measurement data gathering.

IGSTK tracker component

The IGSTK tracker component was designed and implemented with the above ten requirements as guiding principles. Table 1 shows the software technologies and/or the IGSTK implementation that fulfill the above requirements. Three main classes provide the tracker support: (1) ***igstk::Tracker***, (2) ***igstk::TrackerTool***, and (3) ***igstk::Communication*** classes. The ***igstk::Tracker*** class presents a generic interface for tracking the positions of objects in a surgical scene. Derived subclasses provide tracker specific implementations for several widely used tracking systems as shown in Fig. 4. The current tracker classes include ***igstk::PolarisTracker*** and ***igstk::AuroraTracker*** for NDI trackers, and ***igstk::MicronTracker*** for the Claron Technology MicronTracker.

Tracker tools are abstracted in ***igstk::TrackerTool***. The tracker class can track multiple tools simultaneously. A communication class (***igstk::Communication***) class was implemented to establish the communication between the tracker

Table 1 Tracking software requirements and IGSTK implementation

Requirement	IGSTK implementation/technology
Tracking device abstraction	igstk::Tracker abstract class
Tracking tool abstraction	igstk::TrackerTool abstract class
Extensibility	Tracker and Tracker tool class provide implementation of essential behaviors that can be easily subclassed to implement tracking-device specific classes (such as Polaris, Aurora, and MicronTracker)
Portability	Cross-platform software development technologies (Dashboards, CMake build system)
Decomposition	Component-based architecture
Reduced latency	C++ implementation
Fault-tolerance	State machine
Transparency	Loggers
Deterministic	State machine
Time synchronization and dynamic object handling	Pulse generators and transform time stamps

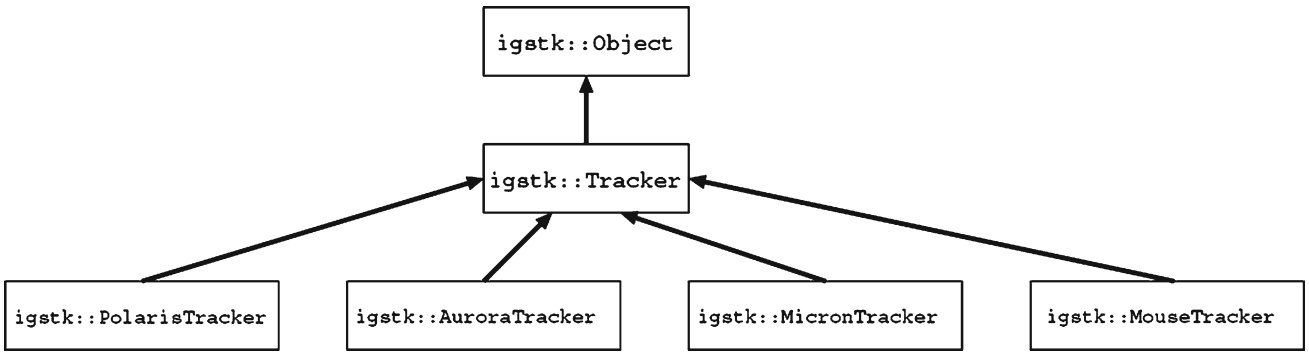

Fig. 4 Tracker class hierarchy

class and the hardware tracking device. Currently supported interfaces include RS232 over a serial port or TCP/IP sockets.

State machines

All the tracker classes are designed for state machine control. State machines ensure that the tracker and tracker tool classes are always in a known configuration. State machines contain a set of states, state inputs, and state transitions. IGSTK provides an ***igstk::StateMachine*** class that offers a set of public methods for programming, executing, and querying state machine logic. Figures 5 and 6 show the state machine diagrams for the *Tracker* and *TrackerTool* classes respectively.

The tracker class contains the following major states:

1. *Idle*. Initial state.
2. *CommunicationEstablished*. This state is entered if the tracker class establishes communication with the tracking device successfully. For example, for MicronTrackers, the camera calibration file is loaded and the cameras are setup by invoking relevant commands in the MicronTracker library.
3. *TrackerToolAttached*. This state is entered if a tracker tool is instantiated, configured, and successfully attached to the tracker.
4. *Tracking*. The tracker will enter this state if a request (*RequestStartTracking*) is made by the user to start tracking.

The tracker tool class contains the following major states:

1. *Idle*. Initial state.
2. *Configured*. The tracker tool makes a transition to this state if all the required parameters of the tool are specified. The required parameters depend on the type of the tracker tool. For example, for the MicronTracker tool, a marker name is required, whereas for a wireless polaris tracker tool, a SROM filename is required.
3. *Attached*. Once the tracker tool is instantiated and properly configured, a request will be made to attach it to the tracker. This request will be processed by the tracker class. During this time, the tracker class will verify if the parameters specified for the tracker tool matches with what the tracker identifies by querying the actual hardware. Once the tracker tool is validated, it will make the transition to the *Attached* state.
4. *Tracked*. During tracking, if the tracker tool is identified in the measurement volume, it will stay in the *Tracked* state.
5. *NotAvailable*. During tracking, the tracker tool could move out of the measurement volume or lose line-of-sight. If that happens, the tracker tool will make the transition to the *NotAvailable* state.

In addition to these major states, transitional states exist that the tracker waits in until the requests are accomplished successfully. For example, the tracker makes a transition to

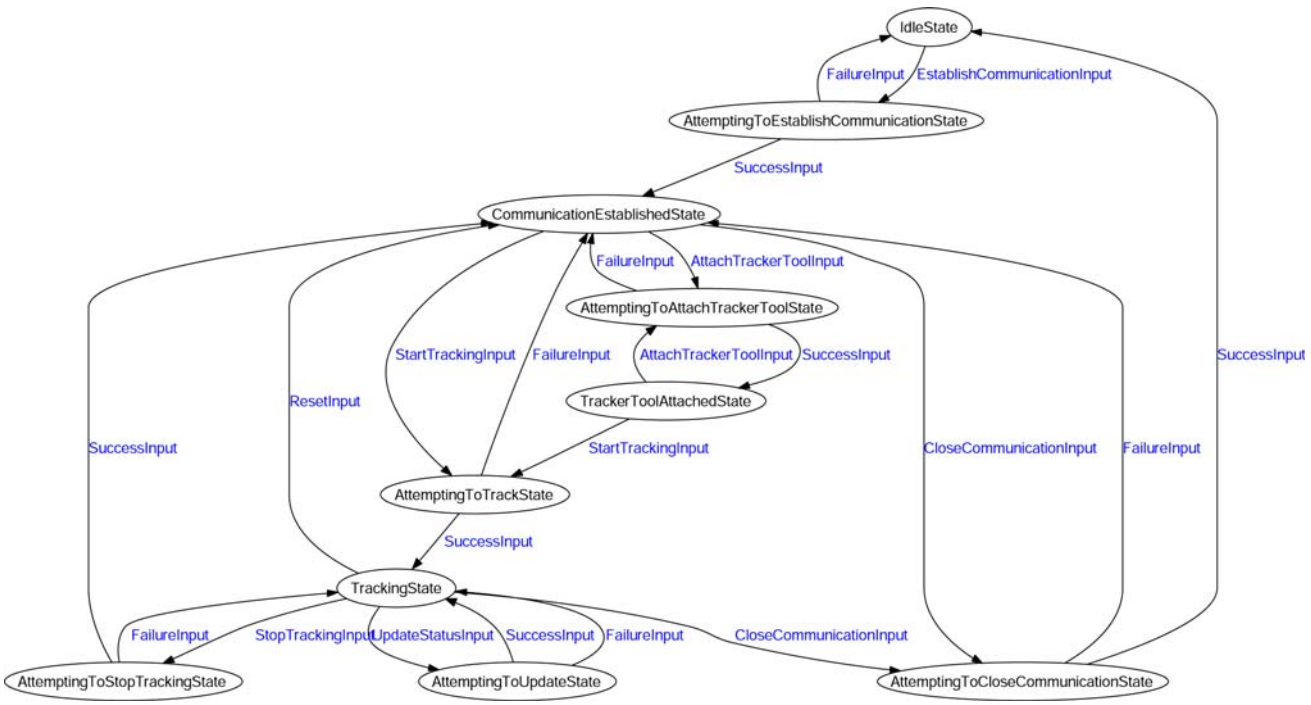

Fig. 5 Tracker state machine diagram

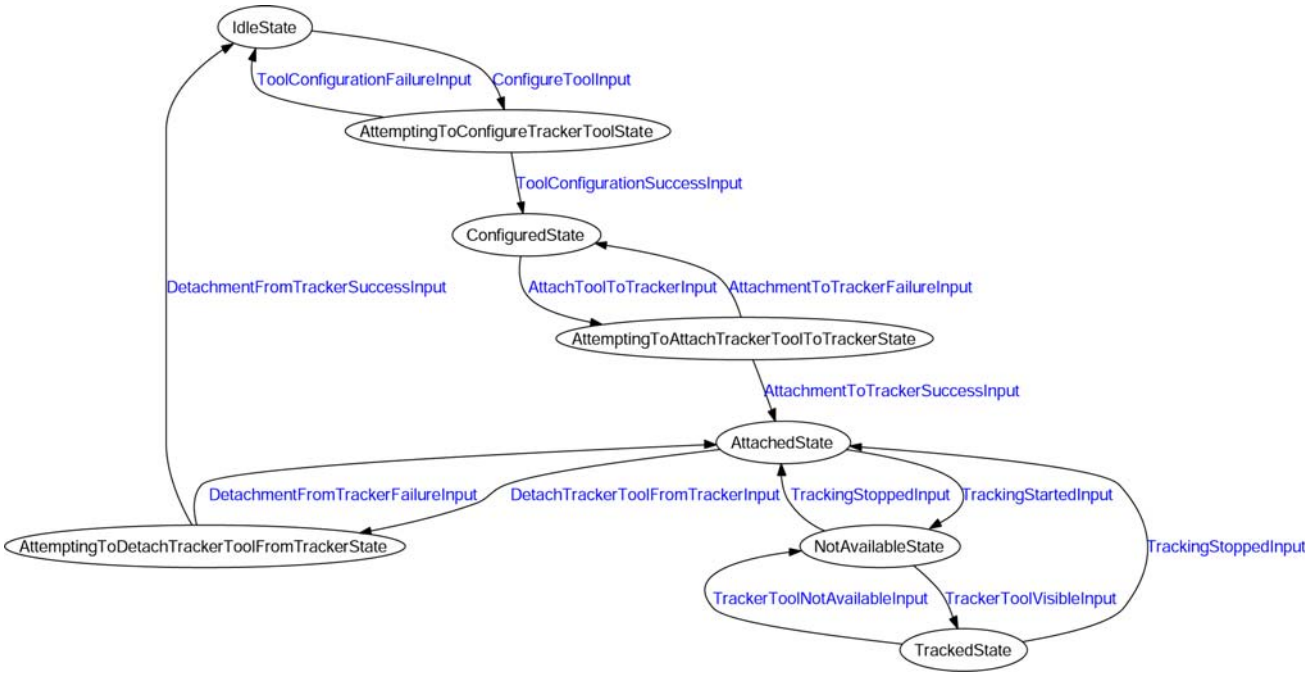

Fig. 6 Tracker tool state machine diagram

AttemptingToEstablishCommunicationState when a ***RequestOpen()*** method is invoked.

Multithreading and data buffering

All components in IGSTK other than the tracker component run in a synchronous mode. For the tracker component, however, concurrent execution is used since the tracking hardware has its own update cycle independent of the update rate of the application's main event-handling cycle. If position measurement gathering is designed to run synchronously with the rest of the system, it will introduce latency and temporal aliasing in the behavior of the system. Hence a software design decision was made to spawn a separate thread for tracking device communication. As shown in Fig. 7, a separate thread is dedicated to communication with the tracking device. Position measurements gathered from the tracking device will not be immediately used as they are received. Instead, they will be buffered in a transform container inside the tracker. The transform will be copied from the buffer to the tracker tool objects when the pulse generator generates a pulse (as shown Fig. 7) indicating that it is time for the tracker tool transforms to be updated in the main application.

Pose measurement data representation

After reading a position and orientation measurement from the tracker hardware, the tracker class generates an ***igstk::Transform*** object to store the position and orientation

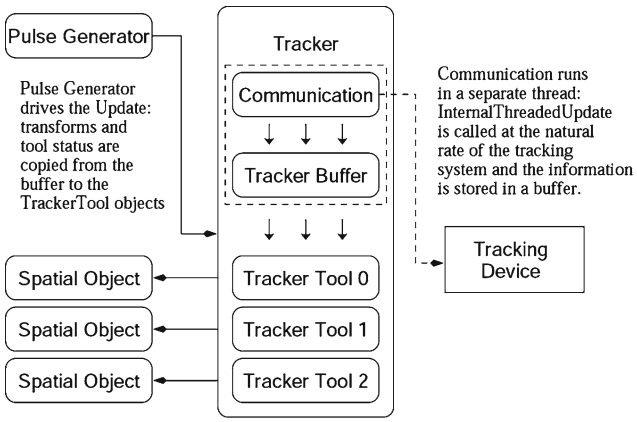

Fig. 7 IGSTK Tracker component structure

measurement data. The ***igstk::Transform*** object contains a vector with three position coordinates, a versor that describes the orientation of the tool, a timestamp that gives the time at which the measurement was made, and an expiration time after which the measurement should be considered invalid. The expiration mechanism allows other components in the toolkit to know when the transform will no longer reflect the spatial position of a tracked object.

IGSTK in use: needle biopsy application with multiple tracker support

Needle biopsy is a common medical procedure for diagnosis of lung, breast, liver and prostate cancer. During the

Fig. 8 Needle biopsy application with multiple tracker support

procedure, a clinician uses X-ray or other image guidance to insert a needle into the lesion to take a tissue sample tisse for pathological analysis. An image-guided application that provides real-time visual feedback may assist clinicians in performing the procedure effectively and efficiently, as well as enabling them to target smaller lesions.

A demonstration needle biopsy application with multiple tracker support was developed using IGSTK. The application set up is shown in Fig. 8. A Polaris Vicra optical tracker (Northern Digital, Ontario, and Canada) and MicronTracker (Claron Technology Inc., ON, Canada) were used. Testing was performed using an abdominal phantom (CIRS Model 57, Norfolk, VA).

Figure 9 shows the application screenshot. The application has a four quadrant display for axial, sagittal, and coronal orientation and a 3D view. A tracking device selection menu is provided on the left. Once a selection is made, a configuration window pops up for users to specify device parameters. The parameters vary depending on the selected tracking device type. The application utilizes a fiducial-based rigid-body registration algorithm to establish the transformation between the pre-operative image and patient coordinate system. In the pre-planning stage, the entry and the target point of the needle are established along with the fiducial point coordinates. An I/O class is provided to store and load this treatment plan. The workflow of this application is outlined below.

1. Record patient demographic information.
2. Load the pre-operative CT image using the DICOM file format.
3. Verify the patient information against the DICOM tags. If there is a discrepancy, notify the clinician.
4. Identify and record fiducial points and treatment path on the pre-operative CT image using a mouse pointer. The fiducial points will be used to register the image with the patient coordinate system. A minimum of three non-collinear fiducial points are required.

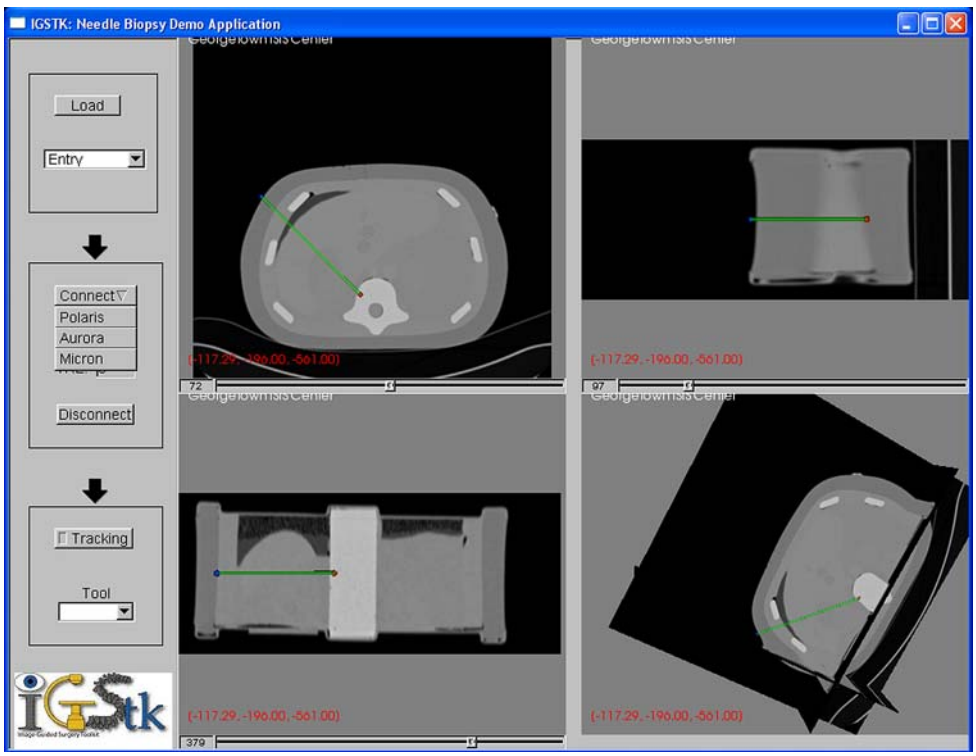

Fig. 9 Screenshot of the needle biopsy application

5. Select a tracking device type and establish device-specific parameters such as port number and SROM file for a Polaris tracker and camera calibration directory and marker template files for a Micron tracker.
6. Identify the corresponding fiducials in the physical body using the tracker pointing device.
7. Perform registration to compute the transformation from patient image to tracker coordinate system.
8. Start tracking and representation of the needle (cylindrical spatial object). The needle path will be displayed overlaid on top of the image.

Having a common framework for different types of tracking devices in IGSTK allows developers to rapidly and easily develop applications with support for different types of trackers. The application source code can be downloaded from the IGSTK Sandbox. For download and other information, the reader is referred to the IGSTK website http://www.igstk.org.

Conclusion

Tracking is a critical component of an image-guided surgery application. IGSTK provides a tracker component that handles communication between a control computer and tracking devices to obtain the position and orientation information for surgical instruments in the surgical scene. Designing tracking software requires establishing software design principles that simplify handling different types of tracking devices in a common framework. Design principles such as abstraction, portability, extensibility, and fault-tolerance are vital for the acceptance, usability, and useful lifetime of the software.

Acknowledgments This project is a collaboration between Georgetown University, Kitware Inc., Arizona State University, and Atamai Inc. All of the software is freely available for download and can be used in research or commercial applications. More information can be found on the website at http://www.igstk.org. This work was funded by NIBIB/NIH grant R01 EB007195 under project officer John Haller. Additional support was provided by U.S. Army grant W81XWH-04-1-007, administered by the Telemedicine and Advanced Technology Research Center (TATRC), Fort Detrick, Maryland. The content of this manuscript does not necessarily reflect the position or policy of the U.S. Government.

References

1. Cleary K, IGSTK Team (2007) IGSTK: the book, an open source C++ software library, Gaithersburg, Maryland, Signature Book Printing
2. Enquobahrie A, Cheng P, Gary K, Ibanez L, Gobbi D, Lindseth F, et al (2007) The Image-Guided Surgery Toolkit IGSTK: An Open Source C++ Software Toolkit. J Digit Imaging 20(Suppl 1):21–33. doi:10.1007/s10278-007-9054-3
3. Schroeder W, Martin K, Lorensen B (2006) The visualization toolkit: an object-oriented approach to computer graphics, 4th edn. Kitware Inc., Clifton Park, NY
4. Ibanez L, Schroeder W (2005) The ITK software guide, 2nd edn. Kitware Inc., Clifton Park, NY
5. Ding J, Khan N, Cheng P, Wilson E, Watson V, Cleary K, Yaniv Z (2008) Accuracy analysis of an image-guided system for vertebroplasty spinal therapy based on electromagnetic tracking of instruments. In: SPIE medical imaging, visualization, image-guided procedures, and modeling; proceedings, vol 6918
6. Peters T, Cleary K (eds) (2008) Image-guided interventions: technology and applications. Springer, Berlin. ISBN 0387738568
7. Ghezzi C, Jazayeri M, Mandrioli D (2002) Fundamentals of software engineering. Prentice Hall, Englewood Cliffs, NJ