Applying a Real-Time Interface to an Optical Tracking System

S. Bruckner[1], R. Seemann[1] and W. Elmenreich[2]
Institut für Technische Informatik
Technische Universität Wien, Austria
[1]*jawz@cg.tuwien.ac.at,* [2]*wil@vmars.tuwien.ac.at*

Abstract

Computer-aided surgery is a young and exciting field of research. It is concerned with visualizing drills and other tools in relation to radiologic data allowing the surgeon to angulate these tools. Motion tracking hardware used in these applications provides high accuracy and update rates. However, current software implementations lack the important real-time constraints required for robotics or profound motion analysis. Hitherto efforts are limited to making the process fast. In this paper we present an architecture enabling applications to access common motion tracking hardware in a hard real-time environment. Our goal is to design and implement an open-source driver for motion tracking applications and to make it freely available for research and further development.

1. Introduction

In the past century radiologic data has become fundamental for planning surgical intervention. Conventional two-dimensional X-ray has been replaced by sliced techniques like CT (computer tomography) or NMR (nuclear magnetic resonance). One big drawback of these methods is their static nature. To overcome this limit static pictures and movement data can be combined. State-of-the-art infrared cameras detect movements of special markers which can be attached to patients or tools. Computer navigation allows the surgeon accurate views of the patient's body during surgery. Systems have been developed that can assist the clinician in diagnosis, treatment planning, and the treatment itself.

One example is examination of jaw movement. In the past years this has been achieved by attaching mechanical devices to the lower jaws teeth and to the skull, letting a pencil draw on a plane. Nowadays this can be done using an infrared camera in combination with light weight markers placed on the patient's jaw. Curves of movement can be recorded and analyzed using various visualization techniques.

In these applications accurate tracking is critical to the success of the procedure. Tracking is the process of pinpointing the location of instruments, anatomical structures, and/or landmarks in three-dimensional space and in relationship to each other. The basic concept behind tracking is the following: markers are placed on a body which's position is to be determined, these markers are adapted to emit energy in response to an activation signal or reflect energy from an activable source, a sensor detects the energy emitted or reflected, and this detection is translated into positional information using various algorithms [1].

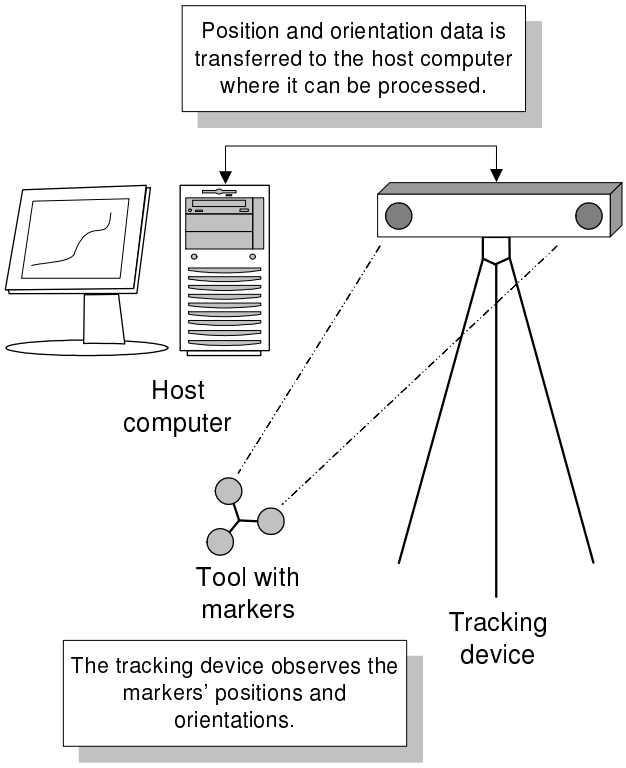

Figure 1: The motion tracking process

Numerous software packages exist to accomplish this task, but most of them lack one essential property: Real-time. Many medical applications require real-time capabilities. They enable experts to interpret motion data, ensure accurate visualization for computer-aided surgery and are basis for the integration of robotics.

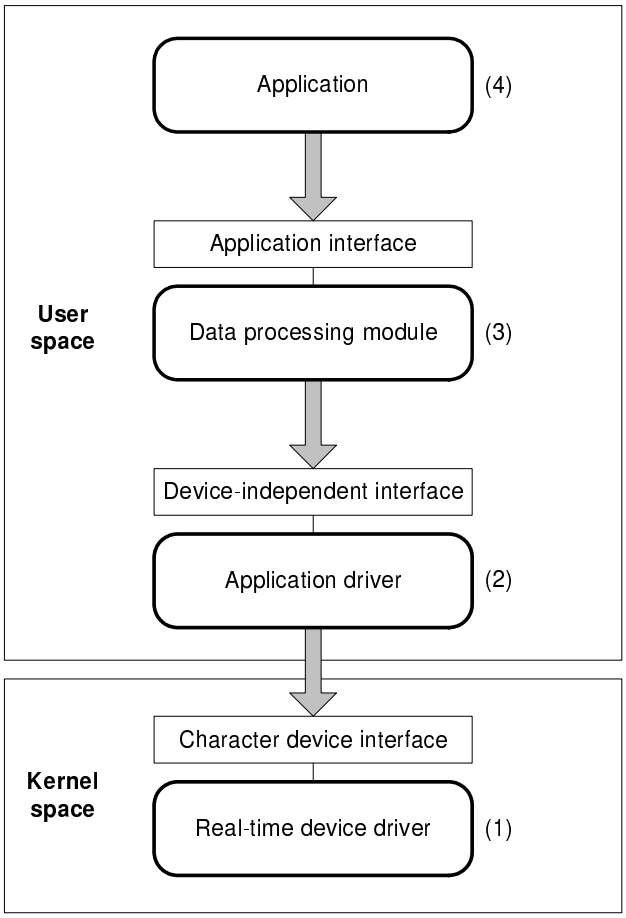

Figure 2: Architecture overview

In the following paper we present a driver model that allows the use of common tracking devices in a hard-real-time environment.

Related work regarding motion tracking for various purposes can can be found in [2] and [3]. An overview over current tracking systems is given in [4].

The remainder of this paper is structured as follows: Section 2 presents our architecture for interfacing motion tracking hardware. In Section 3 we discuss implementation details for the Northern Digital Inc. Polaris system. Finally, Section 4 concludes this paper and gives an outlook on our future work.

2. Architecture

One major feature of a good device driver is that it ”provides mechanism, not policy”. This means that it should mimic all interfacing capabilities of the device (the ”mechanism”), but nothing more. It should not try to interpret exchanged data in any possible user context (the policy), because that is job of a user application itself [5].

A mistake commonly made by authors of device drivers which aim to utilize a device for a single application, is that the driver depends to much on that application and might be difficult to use for other tasks.

Therefore, our driver architecture consists of two distinct components: A device-level driver, which provides access to the hardware through a real-time task and an application-level driver which provides a device-independent application interface.

Since we chose Linux as our host operating system we must also keep the following Linux-specific issues in mind [6]:

Kernel-space vs. user-space: The Linux operating system has two levels: Only privileged processes can run in the kernel, where they have access to all hardware and to all kernel data structures and system calls. Normal application programs can run their processes only in user space, where these processes are shielded from each other, and from direct access to hardware and to critical data of the operating system.

Real-time environment: In real-time systems it is essential that all timing delays are both short and deterministic. The Linux operating system does not natively support real-time features. Our approach therefore relies on RTAI (real-time application interface) which provides a real-time capabilities based on the Linux kernel. This extension is similar to RT-Linux, but provides a richer API. Since our driver is an open-source project we chose this free extension to the Linux platform rather than a commercial real-time operating system.

Our basic architecture is outlined in Figure 2. Device communication takes place within a real-time kernel module (1) which collects data from the tracking hardware and produces timestamped output which is stored in a sufficiently large ring buffer. A Linux character device driver allows to configure the hardware using the standard ioctl() function call. The read() function empowers the application-level part of the driver (2) to access the acquired data. This module performs conversions and provides structured access to the data through a device-independent interface. A major part of the system is a data processing module (3) which improves information quality for the application (4) through the means of mathematical methods such as interpolation and extrapolation techniques [7].

Figure 3 gives a better understanding of data and control flow during the tracking process. As stated in [8] communication between two subsystems is either controlled by the sender’s request (push style) or by the receiver’s request (pull style). The push method empowers the supplier to send messages at any time while the consumer has to wait for incoming data. In the pull mechanism, whenever the

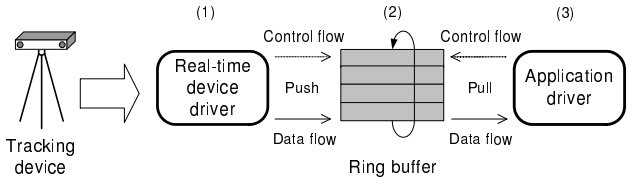

Figure 3: Data and control flow during motion tracking

consumer wants to access message information, the supplier has to provide the requested data. The ring buffer (2) mentioned in the previous paragraph acts as a push consumer for the real-time device driver (1) and as a pull supplier for the application driver (3) which decouples communication between these two modules. Since both the role of the push consumer and the role of the pull supplier is taken by a memory element (the ring buffer) the negative effects of the push and pull communication do not affect the system performance. However, this buffer must be able to store all data for the maximum recording duration, since no assumptions can be made about the pull consumer's speed. If the buffer is too small data could be overwritten before it was read.

3. Implementation

We will first implement this architecture with the Northern Digital Inc. Polaris system, a widely used optical tracker. This infrared device offers 6DOF (6 degrees of freedom, determination of position and orientation). Optical tracking for the Polaris system is accomplished first by setting up multiple CCD (charge couple device) sensors to detect the energy emitted or reflected by the marker. In the case of reflected energy, the process is referred to as passive sensing and in the case of emitted energy it is referred to as active sensing. A single marker is energized per sensor cycle to emit infrared energy. During each sensor cycle, the emitted energy focused on to the sensor is collected and shifted to the sensor processing circuitry. To determine the three-dimensional position of a marker it must be detected on at least three sensor axes, to cover a minimum of three orthogonal planes. Mathematical processing using the technique of triangulation determines six degrees of freedom, defined as being the 3D coordinates and angular orientation.

The Polaris system has three modes of operation: setup, tracking and diagnostic [10].

Setup: Setup mode allows you to configure the system and the attached tools.

Tracking: In tracking mode, the system measures the position and orientation of the tools in real-time and returns the information to the host computer.

Diagnostic: This mode of operation is used for determining possible causes failures. This includes functions like checking the environment for infrared sources that might interfere with the camera.

Table 1: Polaris technical specifications [9]

General	
Accuracy	0.35 mm 3D RMS (1)
Workstation Interface	
Interface	RS-232/422
Max. Data Rate	115 kBaud
Position Sensor	
Weight	2 kg
Mounting	1/4" thread tripod mount
Dimensions	590 mm x 80 mm x 120 mm
Tool Interface Unit	
Weight	5 kg
Dimensions	320 mm x 130 mm x 300 mm
Power Requirements	
hybrid	100/120/220/240 V, 50/60 Hz, 2.5 A
passive	100-250 V, 50/60 Hz, 0.8 A
hybrid POLARIS	
Update Rate	Up to 60 Hz (max rigid body rate) (2)
Max. Markers/Tool	20 (active), 6 (passive), 6 (active wireless)
Max. Tools	9 simultaneously (3 active, 6 passive/active wireless)
Tool Change	Automatic active, Software controlled passive
Tool Options	3 switches, 4 visible LEDs per active tool
passive POLARIS	
Update Rate	Up to 60 Hz
Max. Markers/Tool	6
Max. Tools	6 simultaneously (up to 3 active wireless)
Tool Change	Software controlled

(1) based on a single marker stepped through more than 1000 positions throughout the measurement volume, using the mean of 30 samples at each point at 20°C - (2) varies according to tool combinations with a maximum rigid body rate of 60 Hz

Some commands will only work during specific modes of operation. The real-time driver will handle mode-switches transparently and provide methods to access all capabilities of the device.

Messages between the host computer and the Polaris system are always initiated by the host computer. The host computer issues a command to the system and the system responds with a reply. The Polaris system provides an RS232/422 interface. Therefore, we utilize the rt_com serial port driver which, in combination with RTAI, allows deterministic access to the hardware. However, the timing characteristics of Polaris itself are not specified exactly and have to be evaluated during the implementation. We will perform a number of statistical tests with different setups in order to determine timing boundaries. We plan to acquire more distinguished timing specifications since the vendor's

update rate declaration of ”up to 60Hz” is not sufficient for certain applications, e.g. computer-aided surgery. As stated in [11] environmental disturbances may affect precision. In addition to that it has be determined whether such disturbances influence the processing time of the camera. This information will be used to develop an application which gives the user detailed feedback about the achievable update rates for different tool configurations. The data processing module introduced in the previous section will be used to compensate possible inaccuracies in both measuring and timing. Our goal is to almost completely remove jitter and to provide a continuous stream of data.

4. Conclusion and Future Perspective

We have showed a general architecture for interfacing motion tracking hardware in real-time. It hides hardware characteristics from the application and decouples non-real-time and real-time components of the system. In a first step we will implement a device driver for the Polaris system, but we are planning to support several other devices. The source code will be freely available for research and development.

Our driver can be used for various motion tracking tasks, but its focus lies on systems for medical motion-analysis and surgical navigation. We plan to develop a reference application for evaluation and research purposes.

Further extensions may include the combination of multiple devices using sensor-fusion techniques including the integration of time-triggered protocols such as TTP/A [12].

5. References

[1] D. Simon. Intra-operative position sensing and tracking devices. In *Proceedings of the First Joint CVRMed / MRCAS Conference*, pages 62–64, June 1997.

[2] M. Ribo, A. Pinz, and A. Fuhrmann. A new optical tracking system for virtual and augmented reality applications. In *Proceedings of the IEEE Instrumentation and Measurement Technology Conference, Budapest*, volume 3, pages 1932–1936, May 2001.

[3] K. Dorfmueller and H. Wirth. Real-time hand and head tracking for virtual environments using infrared beacons. In *International Workshop on Modelling and Motion Capture Techniques for Virtual Environments, CAPTECH'98, Geneva, Switzerland*, Nov. 1999.

[4] M. Ribo. State of the art report on optical tracking. Technical Report 2001-25, VRVis, 2001.

[5] A. Rubini. *Linux Device Drivers*. O'Reilly, 1998.

[6] M.J. Bach. *The Design of the UNIX Operating System*. Prentice Hall, 1986.

[7] R.T. Azuma. *Predictive Tracking for Augmented Reality*. PhD thesis, Dept. Computer Sciences, Univ. of North Carolina, Chapel Hill, 1995.

[8] W. Elmenreich, W. Haidinger, and H. Kopetz. Interface design for smart transducers. In *IEEE Instrumentation and Measurement Technology Conference, Budapest, Hungary*, May 2001.

[9] Northern Digital Inc. *Polaris Technical Specs*. http://www.ndigital.com/polaris_technical.html (current Apr. 2002).

[10] Northern Digital Inc. *Application Programmer's Interface Guide (Manual Version 1.0)*, Dec. 2000.

[11] A. Wagner et al. Quantitative analysis of factors affecting intraoperative precision and stability of optoeletronic and electromagnectic tracking systems. *Medical Physics*. accepted for publication.

[12] W. Elmenreich and S. Pitzek. Using sensor fusion in a time-triggered network. In *Proceedings of the 27th Annual Conference of the IEEE Industrial Electronics Society, Denver, Colorado*, volume 1, pages 369–374, Nov.-Dec. 2001.