OP:Sense - Eine Rapid Development Umgebung für die Entwicklung in der robotergestützten Chirurgie

H. Mönnich[1], Philip Nicolai[1], Tim Beyl[1], J. Raczkowsky[1], H. Wörn[1]

[1] Karlsruher Institut für Technologie, Institut für Prozessrechentechnik, Automation und Robotik, Karlsruhe, Germany

Kontakt: tim.beyl@kit.edu

Abstract:

Bei der Entwicklung und Forschung im Bereich der robotergestützten Chirurgie muss viel Zeit aufgewendet werden um die Software, Sensorik und Aktorik zu einem funktionierendem Demonstrator zu kombinieren. Hierbei wird oft Zeit in Basistechniken investiert, die zwar im Prinzip bekannt und vorhanden, aber nicht im konkreten Kontext verfügbar sind. Auch bei verfügbaren Softwarepaketen wie Slicer ist eine Integration in die bestehende Infrastruktur nicht trivial. Wünschenswert wäre eine Entwicklungsumgebung, die möglichst hohe Echtzeitanforderungen erfüllt, eine einfache modellbasierte Entwicklung ermöglicht sowie geringe Anforderungen an die verwendeten Betriebssysteme stellt. Weitere erstrebenswerte Eigenschaften sind die Möglichkeit, einfache Versuche oder Auswertungen zu scripten, das Vorhandensein von einfach zu implementierenden Schnittstellen, einer integrierten Simulationsumgebung für das Robotersystem und einer direkten Formulierungsmöglichkeit für mathematische Probleme. Ein Lösungsentwurf, der diese Anforderungen bereits zu weiten Teilen erfüllt, wird hier vorgestellt. Die beschriebene Umgebung wurde und wird im Rahmen der Arbeit an vier verschiedenen EU-Forschungsprojekten des sechsten und siebten Rahmenprogrammes entwickelt und wurde bereits mit verschiedenen Partnern genutzt. Ziel ist die Etablierung eines OpenSource Projektes speziell für die robotergestützte Chirurgie.

Schlüsselworte: RAD, Chirurgie, Robotik, Haptik, Bildgebung

1 Problem

In der robotergestützten Chirurgie fehlt eine speziell hierfür angepasste Entwicklungsumgebung, die den in Tabelle 1 genannten Anforderungen genügt. Der Anforderungskatalog ist sehr lang und in großen Teilen auch der gleiche wie in anderen Feldern der Robotik in der Forschung. Es gibt verschiedene OpenSource Projekte, die Teile eines solchen Frameworks bereit stellen, dazu gehören unter anderem Orocos [1] oder Ros (Robot Operating System) [2]. Der Fokus von Orocos liegt eher darauf eine Bibliothek für C++ bereit zu stellen die Basistechniken abdeckt, eine Erweiterung erlaubt hier die Verwendung von Matlab/Simulink für einen stärker modelbasierten Entwurf. ROS hingegen entwickelt die Idee eines Betriebssystems speziell für Roboterapplikationen. ROS hat vor allem in letzter Zeit ein hohes Maß an Zuspruch aus der Robotik erhalten und es gibt viele verschiedene Module und Beispielanwendungen. Für Octave gibt es auch eine bestehende Anbindung an ROS. Ein Ansatz aus der Forschung ist eine Architektur für die Teleoperation, genannt Penelope [3]. Der Ansatz liefert zwar keine verfügbare Implementierung, gibt aber einige Hinweise auf die zu erfüllenden Anforderungen. Hierzu gehören: Modularität, Erweiterbarkeit und Wiederverwendung bestehender Software-Komponenten in einer verteilen Umgebung. Echtzeitanforderungen müssen in einer solchen verteilten Architektur eingehalten und Mechanismen etabliert werden um bei Verletzung der Anforderungen passende Maßnahmen einzuleiten. Die Autoren erhofften sich damit sinkende Kosten, einen geringen Aufwand bei der Integration neuer Bestandteile in eine Anwendung sowie einen besseren Zugang zu bestehenden Systemen. Aufgrund von kommerziellen Überlegungen haben Hersteller im Bereich der Chirurgie es bislang vermieden einen solchen Ansatz umzusetzen. In der Forschung an Universitäten hingegen ist der Nutzen für alle Beteiligten klar ersichtlich, wie beispielsweise Slicer 3D sehr gut demonstriert. Einen Ansatz aus der Forschung, der auch praktisch eingesetzt wird, stellt aRD (agile Robot Development) dar, welches am DLR entwickelt wurde. Dieser Ansatz nutzt Simulink um eine modellbasierte Entwicklung zu ermöglichen und stellt eine passende Bibliothek bereit, mit der S-Functions für Simulink einfach entwickelt werden können. Mit aRDnet wird eine passende Infrastruktur bereit gestellt um die Daten in einem Netzwerk verteilen zu können. aRDnet kann sowohl in einer harten Echtzeitumgebung (QNX) oder in weichen Umgebungen (Linux/Windows) genutzt werden. Im Prinzip ist aRDnet ein potenter Kandidat, leider ist die Implementierung nicht verfügbar und wird nur intern im DLR Institut verwendet.

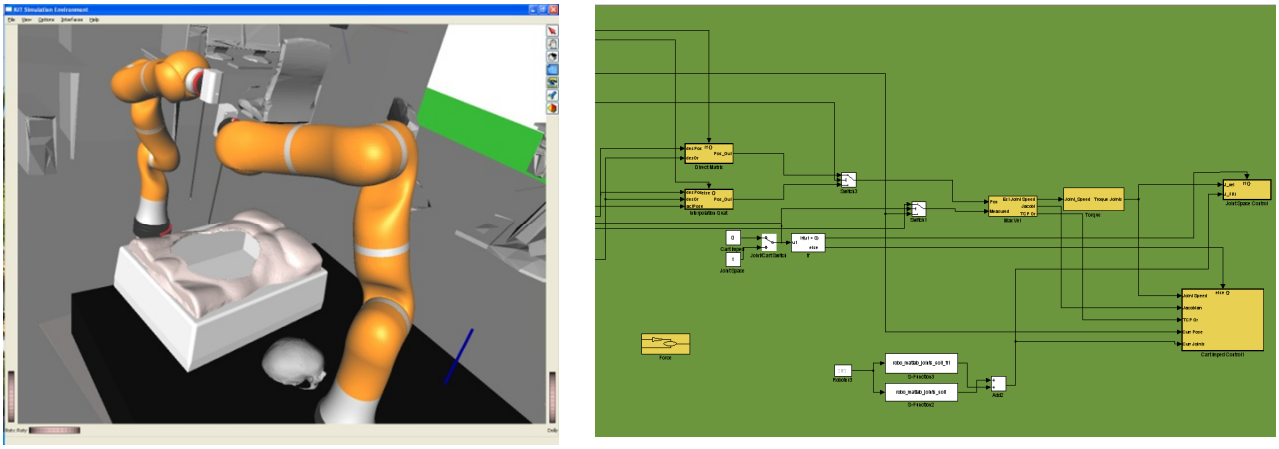
Anforderung	Anforderungen
Echtzeitfähigkeit	Bildgebung (Slicer 3D)
Modelbasierte Entwicklung	Skriptingfähigkeit & Einbindung in nummerische Programme
Verwendung von Standard Betriebssystemen (Windows / Linux)	Einfaches Debugging
Simulationsumgebung	Klar definierte Schnittstellen, Kommunikationsframework das auch über Netzwerke genutzt werden kann
Regelungsmodelle für Roboter	Einfache Erweiterbarkeit (z.B. Einbindung der Haptik)

Tabelle 1: Anforderungen an eine RAD-Entwicklungsumgebung in der Forschung und Entwicklung

2 Methoden

Bei dem hier vorgestellten Ansatz wurden Abstriche hinsichtlich der Anforderung an harte Echtzeit gemacht. Harte Echtzeit ist unter Standardbetriebssystemen wie Windows oder Linux ohne Modifikation des Systemkerns nicht möglich. Ein anderer Nachteil ist die Abgrenzung von Echtzeit- und Nicht-Echtzeit-Komponenten, womit IPC (Inter Process Communication) Mechanismen wie Shared Memory nicht mehr genutzt werden können. Ebenso liegen nicht alle Bibliotheken passend vor und eine Modifikation der Software bedeutet oft hohen Aufwand bei geringem Nutzen. Das hier vorgestellte Framework nutzt die Realtime CORBA Spezifikation und die passende Implementierung ACE TAO, die eine sehr effiziente Implementierung des Standards bietet. Schnittstellen werden in der IDL (Interface Definition Language) definiert und vom TAO Compiler direkt in C++ Code umgesetzt. Der CORBA Naming_Service wird als zentraler Dienst genutzt, der Auskunft gibt, auf welchem Rechner ein spezifischer Dienst aktiviert ist.
Um eine weiche Echtzeitumgebung zu realisieren laufen sämtliche Prozesse und Threads mit erhöhter Priorität, unter Windows kommt z.B. die höchste Stufe “REALTIME_PRIORITY_CLASS“ zum Einsatz [4]. Für die Roboter läuft ein spezieller Dienst, der den blockierenden RPC (Remote Procedure Call) mittels einer Semaphore auf den Takt des Roboters synchronisiert. Im Falle des KUKA LBR Roboters und der FRI Schnittstelle gibt die Schnittstelle über das Netzwerk den exakten, in einer harten Echtzeitumgebung generierten Takt vor. Somit muss lediglich sichergestellt werden, dass in jedem Takt eine passende Antwort zu geliefert wird. Sollte eine Deadline verpasst werden, so wird dies durch einen Zähler festgestellt und das System kann ggfs. darauf reagieren.
Für die in den Schnittstellen definierten Funktionen wurden passende Gegenstellen für Matlab/Simulink implementiert. Im Falle von skriptingfähigen Kommandos geschieht dies als MEX-Funktion, hierzu gehört z.B. ein PTP Kommando, das somit von der Matlab Kommandozeile genutzt werden kann und für Matlab Skripte verfügbar ist. Im Falle von Funktionen für die Regelung werden diese mittels S-Functions für Simulink aufgerufen. Da Simulink aus dem Model ein C Programm erstellt und kompiliert, ist durch „hand geschriebenen“ Code kaum eine Verbesserung der Leistungsfähigkeit zu erreichen. Als Simulationsumgebung wird Openrave genutzt. Openrave stellt bereits eine komplette Umgebung bereit um CAD Daten zu laden und über Matlab die Umgebung kommandieren zu können [5]. Da Openrave modular aufgebaut ist, ist es einfach möglich das System zu erweitern sofern Funktionen fehlen. Dies wurde insbesondere für 2 Fälle durchgeführt, dem automatischen Übernehmen von Mesh Daten aus der Simulation in den haptischen Renderer und die Integration einer Voxel-Carving Darstellung. Für das haptische Rendern wird Chai3D [6] genutzt, das um eine passende CORBA Schnittstelle erweitert wurde. Der haptische Renderer wird gleichzeitig als Schnittstelle zu den realen haptischen Eingabegeräten genutzt. Die einkommenden Daten werden dabei übersampelt um die Rate von 1 Khz zu erreichen die für eine haptische Rückkopplung benötigt wird. Das gleiche Verfahren wurde bei Slicer3D angewendet. Ein Plugin für Slicer 3D implementiert eine IDL Schnittstelle und ermöglicht so den Zugriff auf Slicer3D. Um auch eine übergeordnete Planung zu ermöglichen wurde die YAWL Workflow Engine eingebunden. Als einzige Komponente nutzt diese nicht CORBA sondern das SOAP Protokoll, da die Engine eine entsprechende Schnittstelle anbietet. Als Gegenstück wurde hier die gSOAP Implementierung verwendet und eine passende S-Function implementiert die einen SOAP Server implementiert und von der Engine aus aufgerufen werden kann. Die Workflow Engine übergibt dabei einen String mit einer Kommandozeile die in der S-Funktion geparst wird. Der String enthält immer ein Kommando samt passenden Argumenten und kann so diverse Aktionen in Simulink triggern und darüber den chirurgischen Workflow abarbeiten.

Abb. 1: Simulationsumgebung für die Roboter (links), Simulink Model für die Regelung der Roboter



3 Ergebnisse

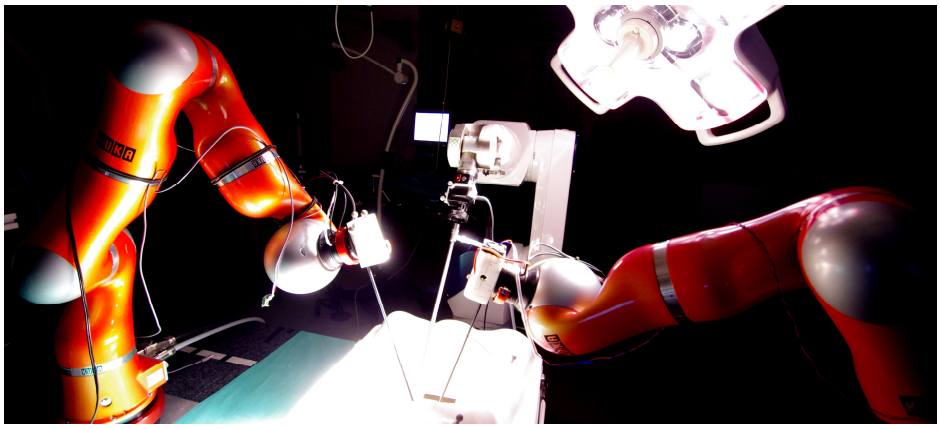


Abb. 2: Robotersystem für die haptische minimalinvasive Chirurgie, bestehend aus 2 LBR Robotern und einem RX90 Roboter (Endoskop). Die beiden LBR Roboter sind mit haptischen Eingabegeräten von Force Dimension gekoppelt.

Das vorgestellte Framework OP:Sense wird im Labor am IPR auf einem verteilten System genutzt, bei dem jeweils ein PC die folgenden Programme ausführt:

- Implementierung der Schnittstelle für das Trackinggerät, den Namensdienst und der Robotersteuerung
- Roboterregelung und Haptik
- Simulationsumgebung
- PMD Kamerasystem
- Testsetup mit Kinect Kameras

Die Roboter werden momentan mit einem Takt zwischen 5 ms und 20 ms über die FRI Schnittstelle angesteuert. Da die LBR Roboter im Gelenkraum mit einem höheren Takt selber interpolieren, ist durch eine höhere externe Taktrate nur eine marginale Steigerung der Dynamik der Roboter zu erwarten, was den Aufwand kaum rechtfertigt.

In weiteren Projekten wird OP:Sense bereits von Studenten und Mitarbeitern genutzt. Dass das System auf jedem Rechner mit Windows sehr schnell installiert und ohne Modifikationen am System betrieben werden kann, erlaubt es auch Studenten Erfahrungen im praktischen Umgang im Bereich der Robotik zu sammeln. Weiterhin wird mit OP:Sense ein Ansatz zur automatischen Repositionierung des Endoskops realisiert. Derzeit wird das Framework in zwei EU Projekten des siebten Rahmenprogramms genutzt: in „SAFROS – Patient Safety in Robotic Surgery“ und „ACTIVE – Active Constraints Technologies for Ill-defined or Volatile Environments“. Darüber hinaus wird das System momentan für eine Veröffentlichung als OpenSource Projekt im Jahr 2011 vorbereitet. Desweiteren werden erste Anwendungen am IPR vorbereitet, in denen OP:Sense für reale Anwendungen unter anderem im Bereich der Orthopädie evaluiert werden kann.

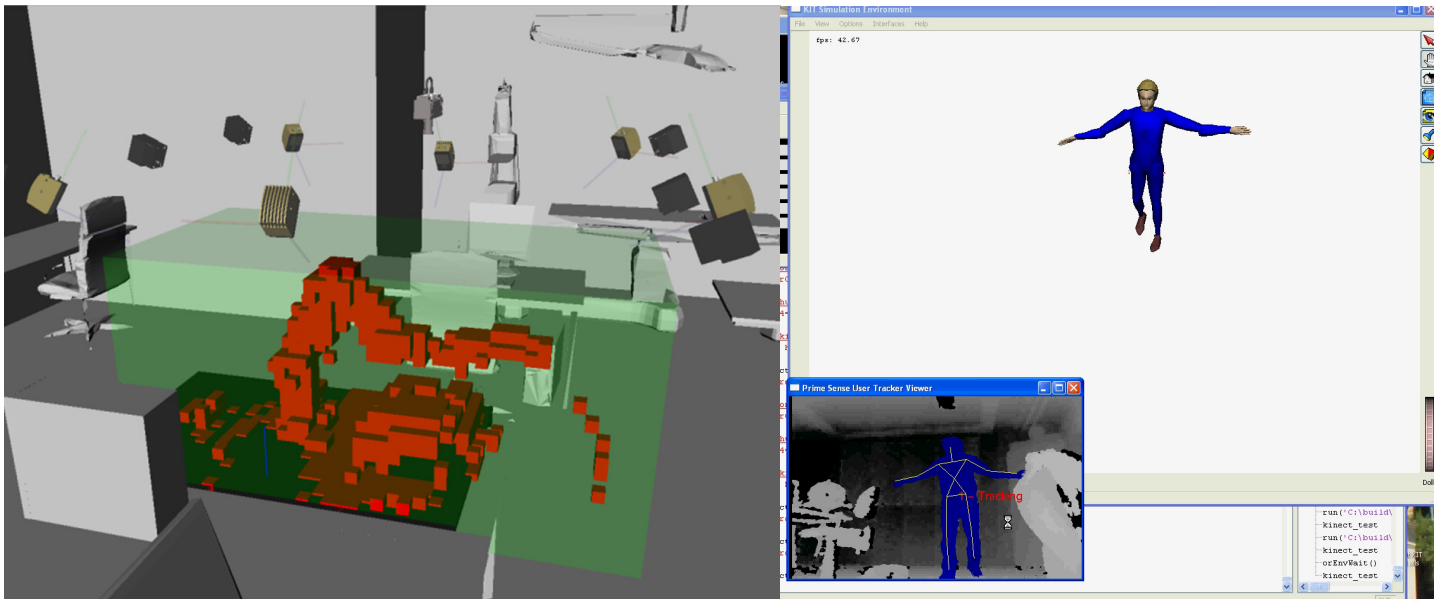


Abb. 3: Zwei Erweiterungen des Setups, die momentan mit dem Framework entwickelt werden. Links: das Resultat des Voxel-Carving Algorithmus, der die Daten von sechs PMD S3 Kameras verwendet und zur Kollisionsvermeidung genutzt werden soll. Rechts: die Einbindung der Microsoft Kinect Kamera in die Simulationsumgebung, um Chirurgen und Patienten markerlos tracken zu können.

4 Diskussion

Die vorgestellte Umgebung zielt auf die schnelle Entwicklung von Beispielanwendungen ab. Dabei wurden harte Echtzeitanforderungen absichtlich nicht berücksichtigt und das Framework stellt nur sicher, dass die Komponenten über eine nicht erreichte Deadline informiert werden. Ein Ansatz um harte Echtzeit in eine RADE Umgebung einfließen zu lassen wurde von Jörg et al. Vorgestellt [7]. Dieser stellt eine interessante Möglichkeit dar, das vorhandene Framework in diese Richtung zu verbessern. Es muss allerdings sichergestellt werden, dass die Komponenten in harter wie in weicher Echtzeit funktionieren können, damit der Kern des Frameworks nicht an Bedeutung verliert: der schnelle und unkomplizierte Aufbau eines kompletten Systems auf herkömmlichen PC Komponenten ohne Modifikation der Betriebssysteme.

5 Referenzen

[1] http://www.orocos.org/, Abgerufen 27.4.2011
[2] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully B. Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. ROS: an open-source robot operating system. In ICRA Workshop on Open Source Software, 2009.
[3] S. Galvan, A. Castellani, D. Botturi, P. Fiorini, “Advanced Teleoperation Architecture”, Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems October 9 - 15, 2006, Beijing, China
[4] http://msdn.microsoft.com/en-us/library/ms685100%28v=vs.85%29.aspx, Abgerufen 27.4.2011
[5] R. Diankov und J. Kuffner. OpenRAVE: A Planning Architecture for Autonomous Robotics. Techn. Ber. CMU-RI-TR-08-34, Robotics Institute, Carnegie Mellon University (2008).
[6] http://www.chai3d.org/, Abgerufen 27.4.2011
[7] S. Jörg, M. Nickl, G. Hirzinger, Flexible Signal-Oriented Hardware Abstraction for Rapid Prototyping of Robotic Systems, Proceedings of the International Conference on Intelligent Robots and Systems, Peking, October, 2006