Netzwerkschnittstelle zur Steuerung eines navigierten Assistenzroboters für die Chirurgie

M. Schlimbach[1], S. Sahm[1], J. Wahrburg[1]

[1] Universität Siegen, Zentrum für Sensorsysteme, Siegen, Deutschland

Kontakt: modicas@zess.uni-siegen.de

Abstract:

Im modiCAS Projekt wird ein navigierter interaktiver Assistenzroboter für die Chirurgie entwickelt. Zur Steuerung des Systems ist eine offene Schnittstelle definiert worden, die es anderen Arbeitsgruppen im Bereich der computerassistierten Chirurgie erlaubt, den Assistenzroboter zur Umsetzung ihrer Planungen zu verwenden. In diesem Beitrag werden der grundlegende Aufbau dieser Schnittstelle sowie die verschiedenen Befehlsgruppen zur Steuerung und Konfiguration des Assistenzroboters beschrieben.

Schlüsselworte: Navigation, Netzwerkschnittstelle, roboterassistierte Chirurgie

1 Problem

Im Bereich der computerassistierten Chirurgie (CAS) gibt es unterschiedliche Forschungsprojekte und kommerzielle Systeme, die auf eine reine Navigationslösung oder auf die Kombination mit einem Roboter setzen. Das modiCAS Projekt realisiert einen navigierten Assistenzroboter, um den Chirurg bei unterschiedlichen Eingriffen zu unterstützen. Ein weiterer Teil des modiCAS Projekts ist das Planungs- und Steuerungsframework modi*CAS* Planning&Control, mit dem medizinische Planungen auf Basis unterschiedlicher Modalitäten angefertigt werden können. Neben der Planung unterstützt das Framework die Umsetzung eines Eingriffes mit dem navigierten Assistenzroboter, der dazu interaktiv vom Operateur kontrolliert wird und die benötigten chirurgischen Instrumente sehr genau und zitterfrei führen und positionieren kann.

Ziel des modiCAS Projekts sind universell einsetzbare Komponenten, die durch eine sinnvolle Modularisierung und klar definierte Schnittstellen ausgetauscht werden können. Die Schnittstelle zwischen dem Assistenzsystem und dem Planungs- und Steuerungsframework wurde nach diesen Gesichtspunkten gestaltet und ist Gegenstand dieses Beitrages. Durch die offene Gestaltung der Schnittstelle soll es anderen Projektgruppen, die sich mit der computerassistierten Chirurgie beschäftigen, möglich sein, zur Umsetzung ihrer Planungen den modiCAS Assistenzroboter einzusetzen.

2 Methode

Das modiCAS Assistenzsystem kombiniert einen kommerziellen Roboterarm und eine 6D-Kamera mit einer selbst entwickelten flexiblen Steuerung zu einen integralen System. Durch die selbst entwickelte Steuerung ist es möglich einen Industrieroboter mit entsprechend offener Schnittstelle an die Anforderungen im OP anzupassen. Die Bedienung des Roboters mittels haptischer Führung ist für den Benutzer sehr intuitiv. Zur Realisierung der haptischen Führung wird am Flansch des Roboters ein 6-dimensionaler Kraft-/Momentensensor montiert, der die auf ihn einwirkenden Kräfte und Momente in und um die Achsen eines räumlichen Koordinatensystems misst. Der Benutzer fasst einen am Sensor befestigten Handgriff, drückt eine Zustimmungstaste und kann den Roboter in die gewünschte Richtung bewegen, ohne genaue Kenntnisse über die Lage von Koordinatensystemen haben zu müssen. Hierbei ist eine Einschränkung des Roboterarbeitsraums mit virtuellen Beschränkungen zum Schutz sensibler Patientenstrukturen möglich [2][3]. Des Weiteren ist das Assistenzsystem durch die Kombination Roboter mit Kamera in der Lage, das am Roboter montierte Werkzeug bei kleinen Patientenbewegungen in Echtzeit nachzuführen [1].

2.1 Systemübersicht

Der Entwurf des modi*CAS* Systems lässt sich in mehrere Ebenen unterteilen (Abbildung 1). Auf der ersten Ebene befinden sich die Aktoren und Sensoren des Assistenzsystems, die mit einem Echtzeitrechner (Realtime-Target) verbunden werden. Von diesem Echtzeitrechner werden grundlegende Basisfunktion, wie Roboterfahrbefehle und weitere echtzeitkritische Funktionen für die höher liegenden Ebenen bereitgestellt. Insbesondere sind hier eigene Regelalgorithmen implementiert, um Bewegungsmodi des Roboters zu realisieren, die an die Anforderungen im OP angepasst sind. Der Echtzeitrechner in der untersten Ebene verfügt über modulare definierte Schnittstellen, so dass einzelne Module, wie der Roboter oder die 6D-Kamera, austauschbar sind.

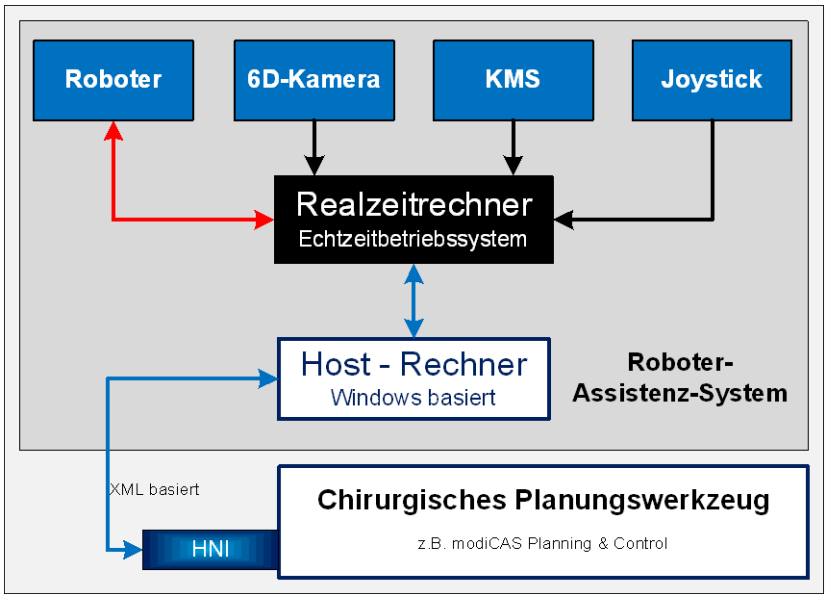

Abbildung 1: Systemarchitektur des modiCAS-Projekts

Auf der nächsten Ebene befindet sich der Host-Rechner, der die untere Anwendungsschicht darstellt. In dieser Ebene sind komplexere Funktionen, wie die Tool- oder Roboterkalibrierung implementiert, die auf die Basisfunktionen des Echtzeitrechners zurückgreifen. Der Host-Rechner bietet den Zugriff auf die aktuellen Prozessdaten des Systems, wie z.B. Pose-Daten aus der 6D-Kamera oder die aktuellen Roboterdaten. Durch Nutzung der Funktionen dieser Anwendungsschicht ist die komplette Steuerung des Assistenzsystems möglich.

Um die Funktionen von einer übergeordneten Planungs- und Steuerungssoftware aufrufen zu können, ist eine entsprechende Schnittstelle definiert worden, das so genannte **H**ost-**N**etwork-**I**nterface (HNI). Die Software des Host-Rechners umfasst die Routinen des HNI auf der Seite des Assistenzsystems. Die genaue Spezifikation des TCP basierten HNI ermöglicht anderen Arbeitsgruppen den Assistenzroboter hierüber mit eigener Software zu steuern.

In unsere selbstentwickelte medizinische Planungssoftware "modi*CAS*-Planning&Control" ist ebenfalls eine Benutzerschnittstelle für das Assistenzsystem auf Basis des HNI integriert worden. Dadurch kann das HNI getestet und optimiert werden und mit weiteren Kommandos ergänzt werden. Auf der anderen Seite können Hersteller von Assistenzsystemen die hier vorgeschlagene Schnittstelle umsetzen, wodurch sich deren Systeme mit jeder Planungssoftware mit integriertem HNI steuern lassen.

3 Ergebnisse

3.1 Aufbau der Schnittstelle

Die Datenpakete des HNI basieren auf einer XML-Syntax und der Kommunikationskanal wird von einem TCP-Socket gebildet. Der TCP-Kanal gewährleistet eine gesicherte Übertragung, bei der keine Daten ohne entsprechende Fehlermeldung verloren gehen können. An die Schnittstelle müssen keine harten Echtzeitanforderungen gestellt werden, da das Assistenzsystem über die Schnittstelle gesteuert aber nicht geregelt wird. Alle zeitkritischen Berechnungen werden innerhalb des Realzeitrechners ausgeführt. Lediglich die Framedaten müssen von der Kamera über den Realzeitrechner und den Host-Rechner ausreichend schnell zur Planungssoftware übertragen werden, um eine zügige Visualisierung zu gewährleisten. Die ist erforderlich, um eine komfortable Hand-Auge-Koordination bei der händischen Positionierung chirurgischer Werkzeuge aufgrund der visualisierten Navigations- und Bilddaten zu ermöglichen.

Die Schnittstelle erlaubt es verschiedene Informationen, wie Betriebsmodus oder Frameposen, aus dem Hostsystem abzufragen oder Befehle an den Host zu übermitteln. Das Protokoll arbeitet befehlsbasiert, das heißt, jede Aktion wird über ein Befehlspaket angestoßen. Jeder Befehl erfordert eine Antwort der Gegenstelle in Form von *Acknowledge-Packets* oder *Datapackets*. Befehle werden *in Command-Packets* versendet, die neben dem Befehl eine Reihe von Parametern in einer dynamischen Liste aufnehmen können. Sendet eine Seite einen Befehl in einem *Command-Packet* erhält sie von der Gegenseite zunächst ein *Acknowledge-Packet*, mit dem mitgeteilt wird, ob der Befehl ausführbar ist oder nicht. Nach der Ausführung des Befehls wird ein zweites *Acknowledge-Packet* gesendet, das eine Erfolgs- oder Fehlermeldung enthält. Wurden mit dem Befehl Daten angefordert, wird anstelle des zweiten *Acknowledge-Packet* ein *Data-Packet* mit den angeforderten Daten übertragen.

Zusätzlich zu dieser befehlsbasierten Kommunikation kann jede Seite wichtige Statusänderungen in Form von *Event-Packets* sofort der Gegenseite mitteilen. Für die Konfiguration stehen spezielle *Configuration-Packets* zur Verfügung, mit denen das Assistenzsystem seine Konfiguration verschicken oder eine neue Konfiguration empfangen kann.

Über einen zweiten TCP-Kanal kann das Assistenzsystem bestimmte Parameter in einem definierbaren Intervall automatisch zur Planungssoftware senden. Die gewünschten Parameter können über einen entsprechenden Befehl von der Planungssoftware zur automatischen Übertragung angefordert werden. Diese automatische Übertragung kann beispielsweise zur Übertragung der Navigationsdaten verwendet werden, die zu Visualisierungszwecken in der Planungssoftware mit einer Frequenz von ca. 10 Hz mit möglichst geringer Zeitverzögerung benötigt werden. Die separate Abfrage der Daten über einen Befehl mit den beiden Antwortpaketen würde an dieser Stelle einen zu hohen Overhead verursachen.

Die Pakete basieren auf XML (Listing 1) und sind in Arrays und Cluster organisiert, wobei ein Array eine beliebige Anzahl von Elementen gleichen Datentyps und ein Cluster eine feste Anzahl genau definierter Datentypen enthalten muss. Der Aufbau der HNI-Pakete wurde so gewählt, dass *Command-*, *Data-*, *Event-* und *Acknowledge-Packets* von der Struktur den gleichen Aufbau haben. Damit lassen sie sich im Host-Rechner automatisch abhängig vom *Pakettyp* in die entsprechende Struktur aus Arrays und Clustern umwandeln.

Listing 1: Aufbau eines Command-Packets am Beispiel des RobMove Befehls

```
<Packet>
<Name>CommandPacket</Name>
   <NumElts>4</NumElts>
   <U32><Name>PacketID</Name>              <Val>2</Val>   </U32> <!--Packet ID -->
   <U32><Name>ReferencePacketID</Name><Val>0</Val>   </U32> <!--Reference Packet ID-->
   <String><Name>Command</Name>
            <Val>APPRobMove</Val></String>                          <!--Name des Kommandos-->
    <Array>                                                         <!--Array mit Parametern-->
      <Name>ParameterList</Name>
      <Dimsize>4</Dimsize>                                          <!--Anzahl der Parameter-->
      <Cluster>                                                     <!--Erster Parameter-->
         <Name>Parameter</Name>
         <NumElts>3</NumElts>
         <String>      <Name>Name</Name><Val>TargetFrameName</Val> </String>
         <String>      <Name>Type</Name>  <Val>String</Val>        </String>
         <String>      <Name>Value</Name><Val>OTS_T_TTP</Val>      </String>
      </Cluster>
      …                                                             <!--Weitere Parameter-->
   </Array>
</Packet>
```

Für jeden Pakettyp ist die Struktur des Clusters festgelegt. Alle haben gemeinsam, dass die ersten beiden Elemente die *PacketID* und die *ReferenePacketID* speichern. Die *PacketID* ist für jedes Paket eindeutig und wird bei 0 beginnend für jedes Paket inkrementiert. Die *ReferencePacketID* wird bei Antwortpaketen verwendet, um eine Referenz auf das Paket zu geben, auf das sich die Antwort bezieht. Dadurch erlaubt das

Protokoll die zeitlich überlappende Ausführung von Befehlen. Der weitere Datenbereich ist abhängig vom Pakettyp. Bei *Command-*, *Data-*, *Configuration-Packets* folgt hier ein Array aus Clustern, wobei jedes Cluster einen Parameter repräsentiert. Da es die Definition eines Arrays verlangt, dass alle Arrayelemente vom gleichen Datentyp sind, muss es sich um Cluster des gleichen Typs handeln. Um trotzdem beliebige Daten in der Parameterliste übertragen zu können, wurde die in Listing 1 am Beispiel eines *Command-Packets* für einen Roboterfahrbefehl dargestellte Struktur eines Arrays aus Clustern entworfen.

3.2 Funktionsübersicht

Bei der Definition der Befehle wurde auf eine möglichst feine Granularität geachtet, so dass mit einer Kombination mehrerer Befehle komplexe Aufgaben durchgeführt werden können, wodurch das Assistenzsystem universell einsetzbar ist. Die Befehle teilen sich in mehrere Bereiche auf. Mit den Befehlen zur Änderung des *Runningmodes* kann der Zustand des Gesamtsystems, der von einer *State-Machine* kontrolliert wird, beeinflusst werden. Zwei wichtige Zustände sind der *Running-* und der *Wait-State*. Eine weitere Gruppe von Befehlen dient zur Konfiguration und Kalibrierung des Assistenzsystems. Sie lassen sich in die Bereiche allgemeine Konfiguration, Konfiguration des Navigationssystems und Konfiguration des Assistenzroboters unterteilen. Das Assistenzsystem unterstützt verschiede Kalibrierungsprozeduren wie z.B. zur Ermittlung der Transformation zwischen Roboter-DRB (Digital Ridged Body) und dem Tool-Center-Point des Roboters, die nach bestimmten Konfigurationsänderungen durchgeführt werden müssen. Für das Navigationssystem müssen die DRBs eingestellt und den verschiedenen Funktionen zugeordnet werden. Für den Roboter lässt sich beispielsweise die Steifigkeit sowie die Freiheitsgrade der haptischen Führung über entsprechende Befehle variieren.

Einen weiteren Bereich umfassen die Befehle zur Steuerung des Navigationssystems sowie zur Abfrage der aktuellen Framedaten (Pose und Sichtbarkeit). Weiter können mit einem Pointer und durch Bestätigung mit einem Fußtaster Punkte gezielt registriert werden. Neben der manuellen Abfrage einzelner Frames können Frameinformationen vom Host-Rechner automatsch versandt werden. Zur Auswahl der automatisch zu versendenden Frameinformationen sowie zur Bestimmung des Intervalls, in dem die Informationen versendet werden, existieren ebenfalls entsprechende Befehle.

Der letzte Bereich betrifft die Roboterfahrbefehle. Hier ist im Wesentlichen der Move-Befehl zu nennen, mit dem der Roboter zu einer Zielpose verfahren werden kann. Die Zielpose wird über eine Transformationsmatrix definiert, wobei die Koordinatensysteme, deren Transformation beschrieben wird, ebenfalls ausgewählt werden können. So ist es beispielsweise möglich, die Zielpose des Roboterwerkzeuges im Patientenkoordinatensystem oder im Koordinatensystem des Navigationssystems anzugeben. Über weitere Parameter können Rotationsanteile der Transformationsmatrix teilweise oder ganz ignoriert werden, so dass bei der Ausführung des Fahrbefehls nur eine Translation durchgeführt wird oder die Rotation um eine Achse unverändert bleibt. Dies ist beispielsweise bei rotationssymmetrischen Werkzeugen von Interesse, um unnötige Roboterbewegungen zu vermeiden. Über einen weiteren Parameter kann bestimmt werden ob die Robotertrajektorie kartesisch (Cartesian space) oder in Gelenkkoordinaten (Joint space) geplant werden soll.

4 Diskussion

Mit dem Host-Network Interface (HNI) steht eine einfach zu implementierende Netzwerkschnittstelle für das chirurgische Assistenzsystem des modiCAS-Projekts zur Verfügung. Durch die Definition weniger grundlegender Befehle kann das Assistenzsystem für unterschiedliche Anwendungen über das HNI gesteuert werden. Die offene Schnittstelle ermöglicht es anderen Arbeitsgruppen, den navigierten Assistenzroboter entsprechend eigner Planungen und Workflows zu verwenden. Durch den modularen Aufbau der modiCAS Steuerung kann diese darüber hinaus an verschiedene Roboter angepasst werden. Der erfolgreiche Nachweis der Funktionalität der HNI Schnittstelle ist im Rahmen eigener Projekte zur Untersuchung der Einsatzmöglichkeiten der Assistenzroboters in verschiedenen chirurgischen Anwendungen erbracht worden.

5 Referenzen

[1] Schneider, H.-C.; Wahrburg, J.: "Neuer Echtzeitkern zur Verbesserung der Dynamik und Sicherheit eines navigierten Chirurgie-Assistenzroboters", at - Automatisierungstechnik, Vol. 56, no. 9, pp. 483-493, 2008.

[2] Raul A. Castillo-Cruces, „Concept and Design of a Cooperative Robotic Assistant Surgery System“, Dissertation, Universität Siegen, 2008

[3] Raul A. Castillo-Cruces, Jürgen Wahrburg, “Virtual fixtures with autonomous error compensation for human robot cooperative tasks”, Robotica, Vol. 28 Part 2, pp. 267-277 , 2010

6 Danksagung

Teile dieser Arbeit wurden im Rahmen des DFG-SPP 1124 und durch das BMBF-Projekt 01EZ0740 gefördert.