UNIVERSITÄT
HEIDELBERG
ZUKUNFT
SEIT 1386

HOCHSCHULE HEILBRONN
TECHNIK WIRTSCHAFT INFORMATIK

3D-Ultraschall beim MITK

Bachelor-Thesis

zur Erlangung des akademischen Grades
Bachelor of Science (B.Sc.) im Studiengang
Medizinische Informatik

vorgelegt von

Cord Lissek

9. März 2017

Referent: Prof. Dr. Rolf Bendl

Korreferent: Dr. Christoph Maier

Betreuer: B.Sc. Peter Seitz

Zusammenfassung

In dieser Bachelorthesis wird ein Frei-Hand-3D-Ultraschallsystem entwickelt. Dieses soll 3D-Ultraschallbildgebung im MITK ermöglichen. Basis hierfür sind ein konventionelles 2D-Ultraschallgerät und ein optisches Trackingsystem. Die nötigen Teilschritte werden erläutert und beschrieben.
Ergebnis ist ein Plugin für das MITK, das aus aufgenommenen 2D-Ultraschallbildern ein 3D-Volumen berechnet.

Danksagung

Ich möchte allen Danken, die mich bis jetzt unterstützt haben.

Ganz besonders meinen Eltern und meiner Schwester, die mir immer den nötigen Rückhalt gegeben haben.

Auch möchte ich mich bei Herr Peter Seitz bedanken, der sich immer sehr hilfsbereit gezeigt hat und mich unterstützt hat wo es Ihm möglich war.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

1.1 Motivation

Konventionelle Ultraschallbildgebung ermöglicht es schnell und ohne Strahlenbelastung des Patienten Einblicke in die Patientenanatomie zu erlangen. Die Systeme sind portabel und kostengünstig. Ein Nachteil der konventionellen Ultraschallbildgebung ist jedoch, dass es nur 2D-Schnittbilder einer (offensichtlich) 3D-Patientenanatomie liefert. Dieses Problem hat man schon gelöst, es gibt auf dem Markt frei verfügbare Ultraschallsysteme, die eine 3D-Bildgebung ermöglichen. Möchte man jedoch auf ein konventionelles Ultraschallsystem zurückgreifen hat man auch die Möglichkeit 3D-Bilder aufzunehmen. Dies kann mit Hilfe eines Freihand-3D-Ultraschallsystems geschehen. Basis hierfür ist ein normales Ultraschallsystem und ein System, dass es ermöglicht, Position und Orientierung eines Gegenstandes im Raum zu erfassen. Durch die Positions- & Orientierungsinformationen lassen sich die Bilder in ein 3D-Volumen einordnen.

1.2 Ziele

Ziel dieser Arbeit ist es, ein Freihand-3D-Ultraschall-System zu entwickeln, das auf einem konventionellen Ultraschall Gerät und der Software MITK basiert. Ergebnis soll ein Plugin für das MITK sein, das aus 2D-Ultraschallaufnahmen einen 3D-Datensatz erstellt. Dafür soll ein handelsüblicher Ultraschallkopf optisch im Raum verfolgt werden. Dessen Position und Orientierung dient dann der korrekten Einordnung der getätigten Bilder in eine geeignete Struktur im MITK.

1.3 Vorgehensweise und Struktur

Zunächst wurde sich mit den Grundlagen des Freihand-3D-Ultraschalls (2.3) erläutert. Daraus wurden die nötigen Vorarbeitsschritte (2.4 und 2.5) herausgearbeitet, die für die Entwicklung des Plugins nötig sind. Anschließend wurden die nötigen Grundlagen des MITK(2.6) erarbeitet. Im Kapitel 3 wird die Durchführung der einzelnen Arbeitsschritte beschrieben und die Architektur des Plugins erläutert. Eine Beispielanwendung wird im Kapitel 4 durchgeführt.

2 Grundlagen

2.1 Szenario

Das zugrunde liegende Szenario könnte wie folgt beschrieben werden. Es liegt ein Ultraschallsystem und eine optisches Trackingsystem vor. Nun soll mit Hilfe des zu erstellendes Programmes eine 3D-Schichtbildaufnahme eines zu untersuchenden Subjektes stattfinden.

2.2 Optisches Tracking

Das optische Tracking basiert darauf, dass ein (steifer) Körper mit mind. 3 angebrachten Markern (derren Anordnung zuvor bekannt sein muss) von mehreren Kameras aufgenommen wird. Aus den unterschiedlichen Bildern kann die räumliche Position und Orientierung ausgerechnet werden. Die verwendeten Werkzeuge (Halterung und Pointer) sind passive Werkzeuge. D.h. sie werden vom Trackingsystem angeleuchtet und reflektieren dann das Licht.

2.3 Freihand 3D Ultraschall

Eine Möglichkeit einer 3D-Volumenaufnahme ohne einen dediziertes 3D-Ultraschallsystem, ist ein Freihand-3D-Ultraschall-System. Dieses besteht aus einem üblichen (2D-)Ultraschallsystem mit einem konventionellen Kopf, der räumlich mit seiner Position und Orientierung verfolgt werden kann. Diese Positionsinformationen dienen dann der richtigen Einordnung der aufgenommenen Bildinformationen in ein 3D-Volumen.

2.3.1 Bildaufnahme

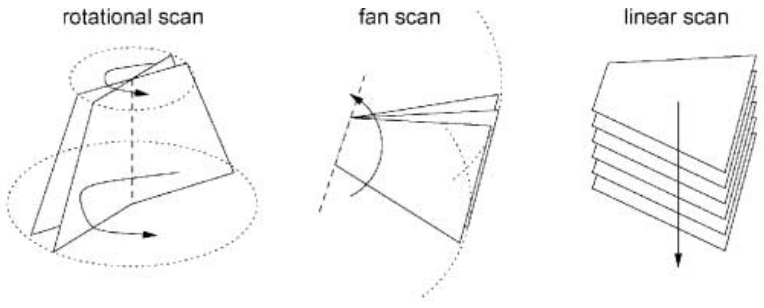

Abbildung 2.1: Unterschiedliche Möglichkeiten ein 3D-Volumen aufzunehmen, Quelle: [2]

Es bieten sich mehrere Möglichkeiten, ein 3D-Volumen aus 2D-Schnitten aufzunehmen, siehe Abb. 2.1.
Beim **Rotational Scan** wird der Schallkopf um sein Zentrum rotiert. Damit kann ein zylindrisches Volumen aufgenommen werden.
Der **Fan Scan** ist eine Art Fächeraufnahme. Der Schallkopf bleibt an das zu untersuchende Objekt angepresst, er wird nur um die Berührachse rotiert.
Am intuitivsten ist der **Linear Scan**, dort werden parallele Schnitte entlang einer Achse aufgenommen.

2.4 Kalibrierung für das optische Trackingsystem

Damit ein Objekt von einem optischen Trackingsystem verfolgt werden kann, muss dem Trackingsystem die Geometrie des zu verfolgenden Objektes bekannt sein (siehe 2.2). Hat man nun ein neues Werkzeug, welches durch optisches Tracking verfolgt werden soll, so muss dieses Werkzeug für das Trackingsystem erst bekannt gemacht werden. Dieser Vorgang nennt sich Kalibrierung. Dafür gibt es mehrere Ansätze, einer davon ist, das Tool mit Hilfe des Trackingsystems selbst zu vermessen [4]. Dabei werden Informationen des Trackingsystems und Zusatzinformationen des Benutzers verwendet, so müssen z.B. Phantommarker (also Objekte, die laut dem System Marker sein könnten, aber es tatsächlich nicht sind) in der Umgebung des Werkzeugs von Hand eliminiert werden.

2.5 Kalibrierung des Ultraschallkopfes

Hsu ([3]) beschreibt in seiner Thesis im Kapitel 1.2.4 **Probe Kalibration** das Problem, welches jedes Frei-Hand-3D-Ultraschallsystem zu bewältigen hat. In einem Weltkoordinatensystem (aufgespannt vom verwendeten Trackingsystem) wird der Ultraschallkopf verfolgt. Dies geschieht bei optischem Tracking so, dass an dem Ultraschallkopf ein Rigid-Body angebracht wird, der Marker-Kugeln enthält, die dem Trackingsystem das Verfolgen des US-Kopfes ermöglicht. Das erwähnte Problem ist nun, dass in diesem Zustand, nur der Rigid-Body der an dem US-Kopf montiert ist verfolgt werden kann. Für die korrekte Einordnung von Ultraschallbildern in ein 3D-Volumen ist es jedoch notwendig, die Orientierung und Position des Ultraschallkopfes (konkret der Ultraschallkopfspitze) zu wissen, nicht die des Rigid-Body. Das Herausfinden wie man von der Orientierung und Position des Rigid-Bodys im Raum auf die Position und Orientierung des Ultraschallkopfes schließen kann, heißt Kalibrierung.
Bei Hsu ([3]) werden einige Verfahren aufgezählt, wie eine solche Kalibrierung erfolgen kann. Aufgrund des zu Grunde legenden Setups wurde sich für ein Verfahren

entschieden, dass neben dem Ultraschallkopf um dem Trackingsystem (welches ohnehin grundlegende Bestandteile eines Frei-Hand-3D-Ultraschallsystems sind) nur ein Zeigerwerkzeug benötigt, dessen Spitze auch im Raum verfolgt werden kann.
Die Kalibrierung mit diesem Setup basiert auf folgender Idee. Man kennt die Koordinaten der Pointerspitze im Raum. Nun kann man Ultraschallbilder der Pointerspitze aufnehmen. Jetzt kennt man zusätzlich die Position der Pointerspitze im Ultraschallbild. Zusätzlich zu diesen beiden Informationen kann mit der Position und Orientierung des Rigid-Body am US-Kopf die Kalibrierung berechnet werden.

2.5.1 Grundlagen

Bevor auf die Berechnung der Kalibrierung eingegangen werden kann, müssen einige grundlegende Schreibweisen eingeführt werden, die so in der Literatur vorkommen. Transformationen können die Umrechnung von einem Koordinatensystem in das andere beschreiben. Hsu ([3]) benutzt für eine Transformation vom Koordinatensystem A in das Koordinatensystem B die Schreibweise $T_{B \leftarrow A}$. Eine Berechnung die wie eine Transformation auch durch eine Matrix ausgedrückt werden kann, aber keine echte Transformation ist (z.B. eine Skalierung) wird der Einfachheit halber auch mit einem großen T beschrieben, nur werden hier im Tiefgestellten keine 2 Buchstaben mit Pfeil verwendet sondern nur einer. So kann eine Skalierungsmatrix durch T_S beschrieben werden. Punkte im Raum, die durch Vektoren beschrieben werden können, werden durch kleine Buchstaben dargestellt. So ist p der Vektor zu einem Punkt.

2.5.2 Koordinatensysteme

Um die Kalibrierung durchführen zu können muss man sich zuerst die vorkommenden Koordinatensysteme klarmachen. Das erste Koordinatensystem ist das Weltkoordinatensystem, welches vom Trackingsystem aufgezogen wird. Im weiteren wird dieses als W für Weltkoordinatensystem bezeichnet. Koordinateneinheiten in diesem System sind Millimeter. Das zweite Koordinatensystem ist das Koordinatensystem welches vom Rigid-Body aufgespannt wird. Folgend wird das Rigid-Body-Koordinatensystem mit S für Sensor bezeichnet. Auch hier ist die Einheit für die Koordinaten Millimeter. Das dritte Koordinatensystem ist jenes, welches das Ultraschallbild beschreibt. Dieses Koordinatensystem wird mit I für Image abgekürzt. Die Einheiten der Koordinaten in diesem Koordinatensystem sind keine Millimeter mehr sondern Pixel. Eine Veranschaulichung der Koordinatensysteme und den Transformationen zwischen ihnen befindet sich in Abb. 2.2.
Nun kann die Gleichung die als Basis der Kalibrierung nach Hsu ([3]) gilt verstanden werden.

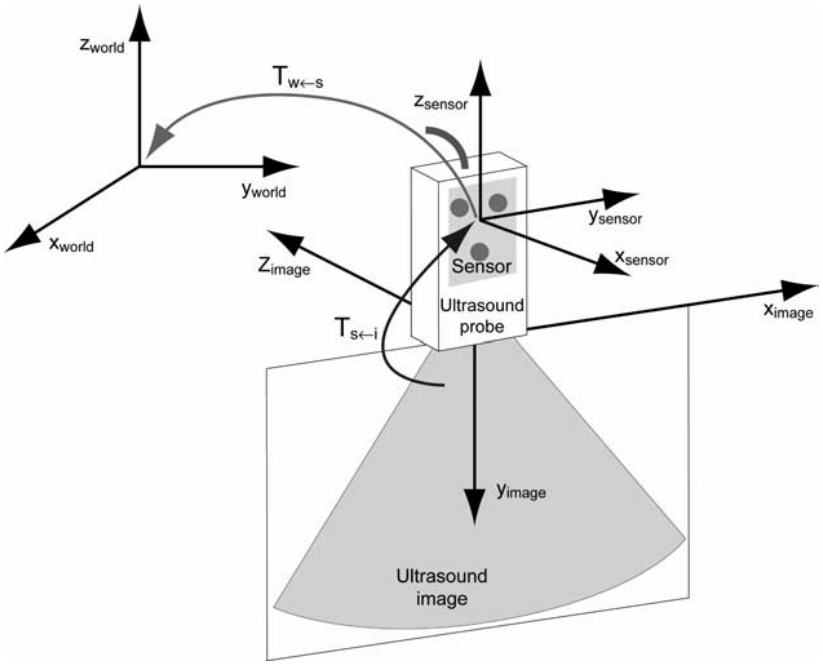

Abbildung 2.2: Die verschiedenen Koordinatensysteme bei der Kalibrierung. $T_{W \leftarrow S}$ ist die Transformation vom Rigid-Body in das Weltkoordinatensystem. $T_{S \leftarrow I}$ die Transformation vom US-Bild in das Koordinatensystem des Rigid-Bodys, Quelle: [11]

2.5.3 Mathematisches Verfahren

$$f_{stylus} = \sum_{i=1}^{N} |T_{W \leftarrow L_i} r^L - T_{W \leftarrow S_i} T_{S \leftarrow I} T_S p_i^{I'_i}| [3] \tag{2.1}$$

f_{stylus} beschreibt den Fehler zwischen der tatsächlichen Position der Pointerspitze und einer errechneten. Das Zustandekommen des Terms $T_{W \leftarrow L_i} r^L$ wird bei Hsu ([3]) nicht beschrieben, es wird lediglich erwähnt, dass es sich hierbei um die Position der Pointerspitze im Weltkoordinatensystem zum Zeitpunkt i handelt. Dies wird einfach hingenommen. $T_{W \leftarrow S_i}$ ist die Transformationsmatrix vom Koordinatensystem der US-Halterung in das Weltkoordinatensystem zum Zeitpunkt i.

$p_i^{I'_i} = \begin{pmatrix} u \\ v \\ 0 \end{pmatrix}$ ist die Position der Spitze des Tools im US-Bild (in Pixeln) zum Zeitpunkt i.

T_S ist eine Skalierungsmatrix. Sie ist notwendig, da ein Pixel im US-Bild nicht einem Millimeter im Weltkoordinatensystem entspricht [3]. Sie hat die Gestalt $T_s =$

$\begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 0 \end{pmatrix}$, wobei wo s_x & s_y die Skalierungsfaktoren, jeweils entlang der x-Achse und y-Achse, im US-Bild sind.

Die Matrix $T_{S\leftarrow I}$ beschreibt die Transformation vom Bildkoordinatensystem des US-Bildes in das Koordinatensystem des Rigid-Body.

Nun kann man sehen, was der Term rechtsseitig des Minus bei f_{stylus} also $T_{W\leftarrow S_i}T_{S\leftarrow I}T_S p_i^{I'_i}$ berechnet. Gelesen wird von Rechts nach Links, da es sich um Transformationen handelt. Die Bildkoordinaten der Pointerspitze zum Zeitpunkt i ($p_i^{I'_i}$) werden zuerst skaliert (mit Hilfe von T_S). Nun werden sie vom Bildkoordinatensystem in das der US-Halterung transformiert ($T_{S\leftarrow I}$) und von dort anschließend in das Weltkoordinatensystem (entsprechend $T_{W\leftarrow S_i}$ zum Zeitpunkt i). Nun hat man die Position der Pointerspitze in Weltkoordinaten.
Damit kann man nun die Differenz der errechneten Position der Pointerspitze und der tatsächlichen Position errechnen. Jeder Betrag eines solchen Termes zum Zeitpunkt i wird in f_{stylus} summiert.

$T_{W\leftarrow S_i}$, $p_i^{I'_i}$ und $T_{W\leftarrow L_i r^L}$ sind jeweils immer (durch das Trackingsystem, bzw. US-Bild) bekannt. $T_{S\leftarrow I}$ und T_S hingegen nicht. Sie sind jedoch für jeden Zeitpunkt i gleich. Die Skalierungsfaktoren ändern sich im Ultraschallbild nicht, auch die räumliche Beziehung von Ultraschallkopfspitze und Rigid-Body ist unveränderlich. Die Gestalt von T_S wurde schon erläutert. $T_{S\leftarrow I}$ ist eine Matrix, die Rotationen um alle 3 Koordinatenachsen und Translationen in alle 3 Achsrichtungen enthält. Bei Hsu ([3]) wird sie wie folgt aufgeführt:

$$T_{S\leftarrow I} = \begin{pmatrix} cos(\alpha)cos(\beta) & cos(\alpha)sin(\beta)sin(\gamma) - sin(\alpha)cos(\gamma) & cos(\alpha)sin(\beta)cos(\gamma) + sin(\alpha)sin(\gamma) \\ sin(\alpha)cos(\beta) & sin(\alpha)sin(\beta)sin(\gamma) + cos(\alpha)cos(\gamma) & sin(\alpha)sin(\beta)cos(\gamma) - cos(\alpha)sin(\gamma) \\ -sin(\beta) & cos(\beta)sin(\gamma) & cos(\beta)cos(\gamma) \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix}$$

Hier ist γ der Rotationswinkel um die X-Achse, β der Rotationswinkel um die Y-Achse und α der Rotationswinkel um die Z-Achse. t_x, t_y und t_z sind die jeweiligen Translationen in die Achsrichtungen [3].
Zusätzlich zu den 2 unbekannten Skalierungsfaktoren aus T_S ergeben sich also insgesamt 8 Unbekannte. Die Einheiten hierbei sind:

Faktor(en)	Einheit
α, β, γ	$^\circ$
$t_x, t_y t_z$	mm
s_x, s_y	$\frac{mm}{Pixel}$

Die Kalibrierung setzt sich nun aus $T_{S \leftarrow I}$ und T_s, die beiden Komponenten die die Unbekannten enthalten zusammen.
$T_{Kalibrierung} = T_{S \leftarrow I} T_s$

2.5.4 Berechnung

Um die Kalibrierung ausrechnen zu können müssen zunächst alle in f_{stylus} vorkommenden bekannten Informationen für N Zeitpunkte gespeichert werden. Diese Informationen sind: die Position der Pointerspitze in Weltkoordinaten, die Transformation vom Rigid-Body-Koordinatensystem in das Weltkoordinatensystem und die Koordinaten der Pointerspitze im Ultraschallbild. Nun kann man f_{stylus} in Abhängigkeit der 8 Unbekannten aufstellen. Die Unbekannten der Kalibrierung sind dann gefunden, wenn f_{stylus} minimal ist. Somit ist f_{stylus} die Basis eines Minimierungsproblems. Muratore et al. ([14]) erwähnen als Lösungsansatz für dieses Problem den Levenberg-Marquard-Algorithmus.

2.5.5 Bewertungsmöglichkeiten

Muratore et al. ([14]) benutzen zur Bewertung der Güte der errechneten Transformation den **Fiducial Registration Error**(FRE) und den **Target Registration Error**(TRE) (siehe [14], Kapitel Accuracy measurment). Dafür teilen sie die aufgenommenen Informationen in 2 Teilmengen ein, die Fiducials und die Targets. Die Fiducials fließen in die Berechnung der Kalibrierung ein, die Targets werden bei der Berechnung ausgelassen. Der FRE wird als Distanz zwischen der tatsächlichen Position der Pointerspitze und der errechneten Position angegeben. Für die Berechnung des FRE werden die Fiducials benutzt. Der TRE hat die gleiche Definition, nur ist beim TRE die Menge der Targets für die Berechnung zu verwenden.
Muratore et al. ([14]) definieren zwar die beiden Fehlerarten, eine Formel zur Berechnung wird jedoch nicht beschrieben. Deswegen wurde aus der Definition eine Formel erstellt. Als Basis dienen die in 2.5.3 eingeführten Teilkomponenten der Gleichung f_{stylus}.
Bevor die Berechnung der beiden Fehler stattfinden kann, wird zuerst die Ausgangslage geklärt. Zum Zeitpunkt der Bewertung ist die Kalibrierung des Ultraschallkopfes bereits bekannt. Also enthalten $T_{S \leftarrow I}$ und T_S keine Unbekannten mehr. $p_i^{I'_i}$ beschreibt bekanntlich die Koordinaten der Pointerspitze im US-Bild zum Zeitpunkt i. Somit kann mit
$T_{W \leftarrow S_i} T_{S \leftarrow I} T_S p_i^{I'_i}$ die mit der Kalibrierung errechnete Position der Pointerspitze in Weltkoordinaten ausgerechnet werden. Die Position ist jedoch ganz genau bekannt, $T_{W \leftarrow L_i} r^L$ beschreibt ja die tatsächliche Position der Pointerspitze in Weltkoordinaten zum Zeitpunkt i.

Nun kann die Differenz der beiden Werte berrechnet werden, das entspricht genau der Definition von TRE und FRE bei Muratore et al. ([14]). Der Sachverhalt wird in folgender Gleichung zusammengefasst.

$$RE_i = |T_{W \leftarrow L_i} r^L - T_{W \leftarrow S_i} T_{S \leftarrow I} T_S p_i^{I_i'}| \tag{2.2}$$

2.6 MITK

2.6.1 Allgemeines

Das MITK (Medical Imaging Interaction Toolkit) ist eine Open-Source-Software für medizinische Bildverarbeitung. Es wird u.a. am DKFZ in Heidelberg entwickelt.

2.6.2 Architektur/Aufbau

Das MITK hat eine spezielle Architektur und benutzt einige Bibliotheken, die unterschiedliche Aufgaben übernehmen.

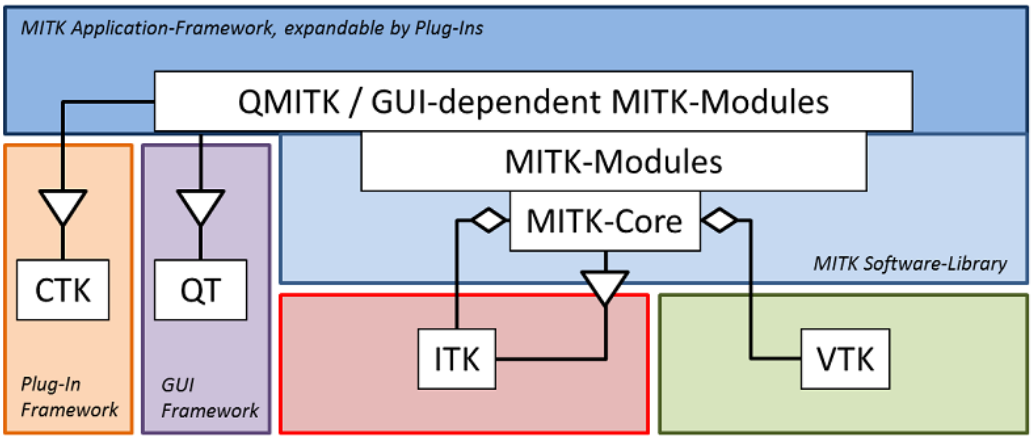

Abbildung 2.3: Übersicht über die Architektur und die verwendeten Bibliotheken die das MITK benutzt. Quelle: [9]

Eine anschauliche Übersicht bietet Abb. 2.3.

ITK Das ITK (Insight Toolkit) ist eine Open-Source-Bibliothek, die v.a. Registrierung- und Segmentierungsalgorithmen anbietet

VTK Das VTK (Visualization Toolkit) ist auch eine Open-Source-Bibliothek, die (3D-) Visualisierung ermöglicht.

CTK Das CTK (Common Toolkit) ist v.a. für DICOM-Unterstützung und Plugins verantwortlich.

QT Das QT-Framwork als Basis für die Benutzeroberflächen.

Diese Architektur wurde gewählt um dem MITK eine große Erweiterbarkeit durch neue Plugins zu ermöglichen[12].

2.6.3 Erstellen eines eigenen Plugins

Das MITK ist durch die Entwicklung von eigenen Plugins erweiterbar. Um jedoch nicht selbst alle nötigen Konfigurationen vornehmen und nötigen Dateien erstellen zu müssen, bietet das MITK ein kleines Kommandozeilentool an, welches die nötigen Konfigurationen durchführt und entsprechende Dateien anlegt, den MITK Plugin Generator [8]. Dieser braucht nur einen Wert für den Namen des zu erstellenden Plugins und führt alles weitere selbständig aus. Ergebnis sind Quelldateien, die den Quellcode des Plugins repräsentieren. Die Entwicklung eines eigenen Plugins kann nun in den erstellten Quelldateien stattfinden.

2.6.4 Verwendete Plugins

Das bestehende Ultraschall Plugin (**Ultrasound Plugin**) für das MITK [13] stellt eine Oberfläche zur Erstellung und Verwaltung von Ultraschallgeräten zur Verfügung. Auch Einstellungen am US-Bild selber, wie z.B. das Beschneiden des Bildes, sind möglich.
Das Plugin **IGT Tracking Toolbox** ermöglicht es, bekannte Werkzeuge mit Hilfe einer Trackingkamera zu verfolgen [15]. Dafür muss der Computer mit der Trackingkamera verbunden sein und in das Plugin muss eine Kalibrierungsdatei geladen werden.
Das Plugin **USNavigation Plugin** besitzt eine Oberfläche, die es ermöglicht, aus einem getrackten Werkzeug und einem Ultraschallgerät eine mitk::USCombinedModality zu erstellen (siehe Sektion Andere Klassen)
Das **Volume Visualisation Plugin** ermöglicht es, 3D-Bilder nicht nur in Schnitten anschauen zu können, es visualisiert das 3D-Bild in 3D.

2.6.5 C++ Microservices

Das MITK verwendet die C++ Micro Service Library. Das ist eine Sammlung von Komponenten die es ermöglichen soll Anwendungen serviceorientiert und modular entwickeln zu können. Ziel der Microservices ist es, Softwarekomponenten wiederverwendbar und erweiterbar zu machen [1].

```
1          us::ModuleContext* context = us::GetModuleContext();
2          object = context->RegisterService(this);
```

Abbildung 2.4: Beispielhafte Registrierung eines Objektes als Microservice

```
1          ServiceReference<MyType> myServiceRef =
2                          us::GetModuleContext()->GetServiceReference<myType>();
3          MyType myObject = us::GetModuleContext()->GetService(myServiceRef);
```

Abbildung 2.5: Beispielhaftes Abfragen eines Objektes das als Microservice registriert ist

Die Microservices ermöglichen es Entwicklern, Teile ihrer Software als Services (nach außerhalb ihrer eigenen Klassen) anzubieten und andersrum auch angebotene Services in ihrer Software zu benutzen. Prinzipiel funktionieren die C++ Microservices so. An einem beliebigem Ort im System wird ein Objekt erzeugt. Der Entwickler des Codes in dem das Objekt erstellt wird möchte, dass dieses Objekt jedoch nicht nur in seinem Code verfügbar ist, sondern im ganzen System. Er registriert es als Microservice. Dazu wird eine Referenz darauf an einer zentralen Stelle hinterlegt. Möchte nun jemand auf dieses Objekt zugreifen, so kann er sich an die zentrale Stelle wenden und dort das Objekt erfragen.

2.6.5.1 Beispielcode

Aus Codebeispielen im Git-Repository konnte sich die Arbeitsweise der C++ Microservices im MITK erarbeitet werden. Die zentrale Stelle im MITK ist ein ModuleContext, als Klasse us::ModuleContext. Möchte man nun ein Objekt als MicroService registrieren kann dies wie folgt geschehen. Es wird ein Objekt (object) vom Typ MyType registriert (Code siehe Abb. 2.4).

Nun ist das Objekt an der zentralen Stelle hinterlegt. Ein Zugreifen auf das Objekt kann erfolgen. Dazu werden ServiceReferences verwendet. Diese dienen dem Zweck, die registrierten Objekte nach dem Typen unterscheiden zu können, um nur ein Objekt einer bestimmten Klasse zu bekommen. Mit der ServiceReference kann nun beim us::ModuleContext das Objekt des bekannten Typen erfragt werden. Ein Abfragen des oben hinterlegten Objektes könnte sich nun wie in Abb 2.5 gestalten.

Zunächste muss beim us::ModuleContext die ServiceReference des Typs erfragt werden, von dem man das Objekt erwartet. Mit dieser ServiceReference kann nun beim us::ModuleContext das Objekt mit der Methode GetService erfragt werden.

2.6.6 mitk::Image

Die wichtigste und zentralste Klasse die verwendet wurde, ist das **mitk::Image**. Ein Objekt dieser Klasse repräsentiert sowohl 2D- als auch 3D-Bilder. Die räumliche Ausdehnung und Orientierung eines Bildes wird durch Geometrys beschrieben (siehe Kapitel 2.6.8).

2.6.7 Manipulation eines mitk::Image

Der Zugriff auf die Pixelinformationen eines mitk::Image kann durch Accessors geschehen. Die 2 Accessortypen sind:

1. mitk::ImagePixelReadAccessor
2. mitk::ImagePixelWriteAccessor

Mit dem ersteren kann auf Pixelinformationen zurückgegriffen werden, mit dem letzteren können Pixelwerte gesetzt werden. Anwendung findet jedoch nur der mitk::ImagePixelWriteAccessor. Der mitk::ImagePixelReadAccessor hat sich als nicht funktionsfähig rausgestellt.

2.6.8 Geometrys

Geometrys beschreiben die geometrischen Eigenschaften von Datenobjekten (wie z.B. eines mitk::Image). Eine Übersicht der Geometrys im MITK befindet sich in Abb. 2.6. Die relevanteste Geometry ist **mitk::SlicedGeometry3D**. Jedes mitk::Image enthält eine mitk::SlicedGeometry3D. Mit Hilfe der Geometrys können aus den Pixel-Koordinaten eines Bildes die Weltkoordinaten des Pixels errechnet werden (und umgekehrt). Das geschieht mit Hilfe von Transformationen. Die Transformationen werden durch ein Datenfeld des Typs **mitk::AffineTransform** gehalten. Ein Objekt des Typs mitk::AffineTransform repräsentiert eine affine Transformation, also eine Transformation, die u.a. Translationen, Rotationen und Skalierungen enthält.

2.6.9 Datenverwaltung

Im MITK wird zum Verwalten von Daten ein Konzept verwendet, das auf Objekten der Klassen **mitk::DataNode** und **mitk::DataStorage** basiert. Sämtliche Datenobjekte die verarbeitet werden sollen (wie z.B. ein mitk::Image) müssen in einen mitk::DataNode gekapselt werden. Dies hat den Zweck, Daten zusätzlich zu den Nutzdaten speichern zu können, wie z.B. Name und Sichtbarkeit des Datenobjekts. Diese mitk::DataNodes können dann einem mitk::DataStorage hinzugefügt werden. Es gibt

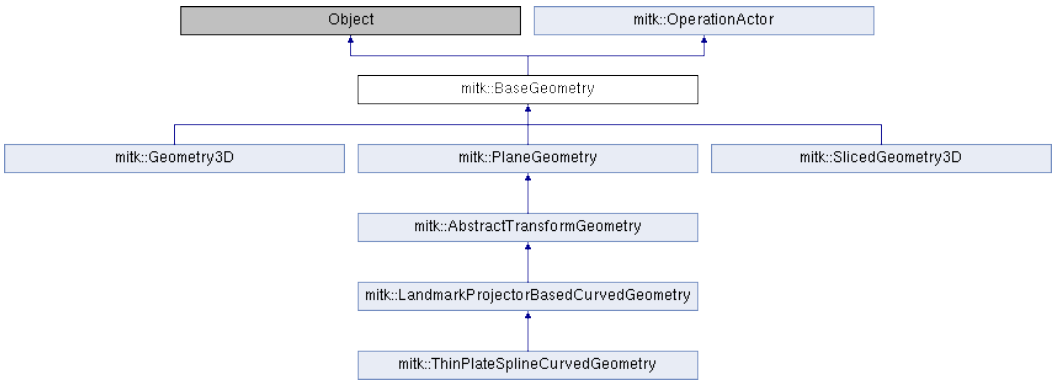

Abbildung 2.6: Klassenhierarchie der Geometrys im MITK

die Möglichkeit selbst mitk::DataStorages zu erstellen, ein MITK-Arbeitsfenster(MITK-Workbench) bietet jedoch auch ein zentrales mitk::DataStorage [7].

2.6.10 Andere Klassen

mitk::USDevice Klasse, die ein Ultraschallgerät repräsentiert. Hier gibt es die Möglichkeit, das aktuelle Bild des Ultraschallgerätes abzugreifen. Ein Objekt dieser Klasse wird im Ultrasound Plugin erstellt und per C++ Microservices außerhalb des Plugins nutzbar gemacht.

mitk::NavigationDataSource Klasse, die ein Werkzeug repräsentiert, welches von der Trackingkamera verfolgt wird. Sie bietet die Möglichkeit, die aktuelle Postion und Orientierung des Werkzeuges abzufragen. Ein Objekt hiervon wird im Plugin IGT Tracking Toolbox erstellt und per C++ Microservice verwendbar gemacht.

mitk::USCombinedModality Repräsentiert ein bildgebenden Ultraschallkopf, der im Raum verfolgt werden kann, d.h. eine Kombination aus mitk::USDevice und mitk::NavigationDataSource. Kann im USNavigation Plugin erstellt werden und wird dort auch als C++ Microservice registriert.

vtkMatrix4x4 Repräsentiert eine 4x4 Matrix.

mitk::NavigationData Stellt die Position und Orientierung eines getrackten Werkzeuges dar. Bietet auch die Möglichkeit diese Informationen als mitk::AffineTransform zu bekommen.

3 Methodik

3.1 Teilaufgaben

Aus der Ausgangslage und der Zielsetzung ergeben sich folgende Teilaufgaben:

1. Kalibrierung der Halterung für das Trackingsystem
2. Kalibrierung des Ultraschallkopfes im Bezug zur Halterung
3. Implementierung des Plugins für das MITK

3.2 Material

3.2.1 Ultraschallsystem - Esaote MyLabFive

Das verwendete Ultraschallsystem ist ein Modell MyLabFive des Herstellers Esaote (Esaote S.p.A., Genua, Italien) (siehe Foto in Abb. 3.1). Der verwendete Schallkopf ist ein CA431. Das System bietet keine API an. D.h. sämtliche Einstellungen sind an der Workingstation des Gerätes durchzuführen, die Bildübertragung an ein externes System erfolgt über den VGA-Videoausgang und den Videograbber.

Abbildung 3.1: Das verwendete Ultraschallsystem, ein Esaote MyLabFive

3.2.2 Videograbber - Epiphan Modell DVI2USB

Der Videograbber (Epiphan Modell DVI2USB (Epiphan Systems, Ottawa, Ontario, Kanada)) ermöglicht es, den Bildschirm der Ultraschall-Workingstation auf einen anderen Computer zu übertragen. Die Verbindung zum Ultraschallsystem geschieht durch ein VGA auf DVI Kabel, der andere Computer ist durch ein USB Kabel mit dem Videograbber verbunden.

3.2.3 Trackingsystem - NDI Polaris Spectra

Das verwendete Trackingsystem ist ein NDI Polaris Spectra (Northern Digital Inc., Waterloo, Kanada). Es ermöglicht Werkzeuge im Raum zu verfolgen. Dazu spannt es ein Koordinatensystem in den Bereich des Raumes, der von der Trackingkamera verfolgt werden kann.

3.2.4 US-Kopf-Halterung mit Rigid-Body

Die Halterung für den Ultraschallkopf wurde am DKFZ in Heidelberg hergestellt. Deren Sinn besteht darin, den Ultraschallkopf zu halten und diesen steif mit einem daran angebrachten Rigid Body zu verbinden. An den Rigid-Body sind 4 Maker-Kugeln angebracht, die dem Trackingsystem ermöglichen, den Rigid-Body (und somit auch den Ultraschallkopf) zu verfolgen. Im langfristig späteren Verlauf des Projektes soll die Halterung es auch dem Roboterarm ermöglichen, den Ultraschallkopf zu halten (Foto, siehe Abb. 3.2).

Abbildung 3.2: Die Ultraschallkopf-Halterung mit montiertem Ultraschallkopf. Man erkennt den Rigid-Body mit seinen 4 Marker Kugeln

3.2.5 Pointer

Der Pointer ist ein einfaches Zeigewerkzeug, dessen Spitze auch vom Trackingsystem im Raum verfolgt werden kann. Ähnlich wie die US-Kopf-Halterung hat der Pointer 4 Marker-Kugeln.

3.2.6 KUKA Leichbauroboter - lbr-iiwa-7-800

Ein Leichtbau-Roboterarm des Herstellers KUKA (KUKA AG, Augsburg, Deutschland). Dient als Halterung für den Ultraschallkopf.

3.2.7 Wasserbad - Plastikbox

Das Wasserbad ist eine einfache quaderförmige, flache Plastikbox, in die später Wasser gefüllt wird. Wird im Kalibrierungsversuch verwendet.

3.2.8 Wasserbad - Eimer

Auch hier soll später Wasser eingefüllt werden. Ein Eimer aus flexiblem Kunststoff, in das ein Loch gebohrt wurde, durch das der Pointer eingeführt werden kann. Wird zum Kalibrieren verwendet.

3.2.9 MATLAB

MATLAB (Hersteller The MathWorks, Inc., Natick, Massachusetts, USA) ist eine Software die für numerische Berechnungen benutzt werden kann.

3.2.10 NDI Track

Software des Herstellers NDI (siehe Trackingsystem), dient der Aufzeichnung von Orientierung und Position von (trackbaren) Objekten.

3.2.11 NDI 6D Architect

Software des Herstellers NDI (siehe Trackingsystem). Erstellt Dateien, die es dem Trackingsystem ermöglichen, zuvor unbekannte Objekte zu tracken.

3.2.12 CMake

Wie auf der Seite des MITK ([6]) wurde CMake (V. 3.4.0) für die Konfiguration des MITK-Projektes benutzt.

3.2.13 Microsoft Visual Studio

Für das MITK-Projekt wurde Microsoft Visual Studio 2013 verwendet.

3.3 Kalibrierung der Halterung für das Trackingsystem

Damit das Trackingsystem überhaupt für die erstellte Halterung verwendbar ist, muss dem System die räumliche Verteilung und Anordnung der sich darauf befindlichen Marker bekannt sein. Systeme des Herstellers NDI benutzen dafür .rom-Dateien. Diese Dateien enthalten die nötigen Informationen über die Anordnung der Marker, die für das Tracking nötig sind. Hat man nun einen Rigid Body mit einer unbekannten Geometrie bietet die Software **NDI 6D-Architect** drei Möglichkeiten an, eine .rom-Datei zu erstellen.

1. Mit dem optischen System selbst (Collection)
2. Mit einer vorhanden Kalibrierungsdatei für ein identisches Tool
3. Mit den Herstellungsdaten

Da weder eine vorhandene Kalibrierungsdatei noch die Herstellungsdaten vorliegen, wurde sich für die Variante Collection entschieden. Der NDI 6D-Architect bietet dafür einen interaktiven Wizard an. Im Laufe der Kalibrierung werden mehrere Male Aufnahmesequenzen des Tools durchgeführt. Um ein möglichst genaues Ergebnis zu erzielen wurde der Rigid Body so ausgerichtet, dass die Marker-Kugeln möglichst direkt in Richtung der Trackingkamera zeigen. Zudem wurde die Kamera so positioniert, dass sich das Tool möglichst in der Mitte des Trackingvolumens befindet.

3.3.1 Einstellungen

Im Laufe der Kalibrierung wurden wesentliche Einstellungen getroffen, die den Rigid-Body und sein Trackingverhalten beeinflussen.

Tool Type Der Tool Type beschreibt um welche Art von Rigid-Body, bzw. welchen Zweck dieser Rigid-Body erfüllt. Es wurde sich für den Tool Type **Reference** entschieden, da alle anderen Tool Types für die geplante Verwendung ungeeignet waren.

Lokales Koordinatensystem Der Tool Type Reference ermöglicht es, ein lokales Koordinatensystem zu definieren. Es wurde so gewählt, dass ein Marker den Ursprung markiert, die X- und die Y-Achse in der Ebene in der sich die Marker befinden verlaufen und die Z-Achse orthogonal dazu. Es wurde deshalb so gewählt, damit das Koordinatensystem des Rigid-Body ähnlich orientiert ist wie das des US-Bildes. Eine Darstellung des Koordinatensystems befindet sich in Abb. 3.3. Außerdem erleichtert dieses Koordinatensystem das definieren der Normalen (siehe nächsten Punkt).

Normalen Für jeden Marker muss ein Normalenvektor definiert werden. Das hat den Hintergrund, dem System sagen zu können, in welche Richtung der Rigid-Body gerade zeigt. Da alle Marker in einer Ebene liegen und in die selbe Richtung zeigen wurden die Normalen als parallel zur Z-Achse des lokalen Koordinatensystems gewählt.

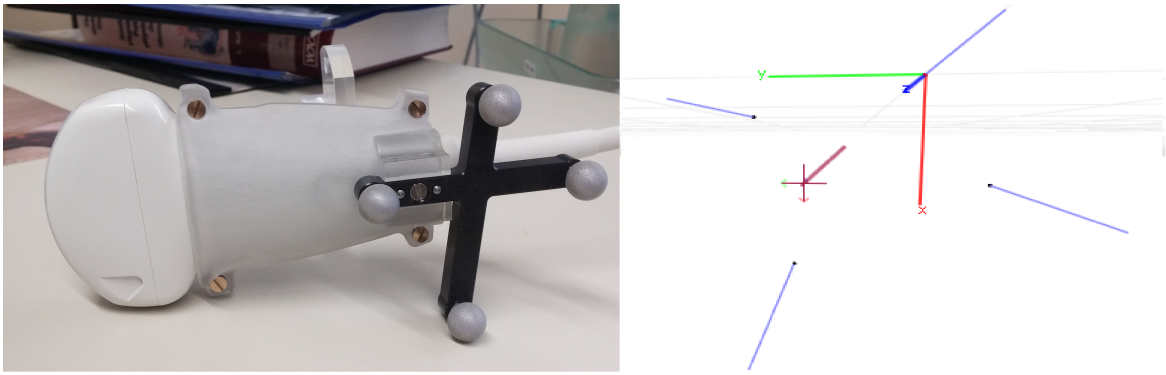

Abbildung 3.3: Rigid-Body und das definierte lokale Koordinatensystem. Zusätzlich zu den Koordinatenachsen erkennt man die definierten Normalen

Ergebnis der Kalibrierung ist eine .rom-Datei, die es nun das Tracken des Rigid-Body ermöglicht.

3.4 Kalibrierung des Ultraschallkopfes im Bezug zur Halterung

3.4.1 Versuchsaufbau

Um die nötigen Informationen gewinnen zu können, muss ein geeigneter Versuchsaufbau realisiert werden. Es wurde versucht einen ähnlichen Versuchsaufbau zu verwen-

den, wie Muratore et al. ([14]) ihn beschreiben. Dort wird ein Wasserbad beschallt, in das ein Pointer eingeführt wird.
Die Speicherung eines US-Bildes ist über das US-System selbst möglich. Ein dort aufgenommenes Bild wird zuerst im lokalen Archiv gespeichert und kann anschließend auf ein USB-Stick (z.B. im .bmp-Format) exportiert werden. Die Gewinnung und Speicherung der relevanten Trackinginformationen des Pointers und der US-Halterung ist mit der Software **NDI Track** möglich.

3.4.2 Versuch Nr. 1

Abbildung 3.4: Versuchsaufbau Nr. 1. Man erkennt das Wasserbad, den Pointer, die US-Halterung mit Rigid Body und den KUKA-Roboter

Wie bei Muratore et al. ([14]) wurde ein Wasserbad verwendet. Als Wasserbad diente eine quaderförmige Plastikbox, in die Wasser gefüllt wurde. Der Ultraschallkopf wurde hier vom KUKA-Roboter so gehalten, dass das Einführen des Pointers (augenscheinlich) orthogonal in die US-Bild-Ebene möglich war (siehe Abb 3.4). Nun wurde der Pointer am Rand der Box fixiert. Anschließend wurden die Trackinginformationen (Orientierung und Position von US-Halterung und Pointer) und das Ultraschallbild (in dem die Pointerspitze zu sehen ist) aufgenommen. Für die nächste Aufnahmesequenz wurde die Plastikbox (mit fixiertem Pointer) verschoben und die Position wieder so angepasst, dass die Pointerspitze wieder orthogonal in die Bildebene zeigt. Nun konnten wieder die Trackinginformationen und das US-Bild aufgenommen werden. Diese Prozedur wurde so oft wiederholt, bis man die ganze Breite des US-Bildes (auf einer Ebene) abgedeckt hat. Als nächstes wurde der Pointer erhöht und die Prozedur ging von vorne los. Bei der Durchführung des Versuches ist aufgefallen, dass es

einige Probleme gab. So war es sehr schwierig die Pointerspitze orthogonal in die Bildebene einzuführen. Auch war es sehr schwer, das Wasserbad so zu positionieren, dass nur noch die Pointerspitze im US-Bild zu sehen war. Deswegen wurde ein verbesserter Versuchsaufbau realisiert (siehe 3.4.3).

3.4.3 Versuch Nr. 2

Um die Probleme des ersten Versuches zu beheben, wurde der Versuchsaufbau angepasst. Bei diesem Versuchsaufbau wurde die Plastikbox durch einen Eimer ersetzt,

Abbildung 3.5: Versuchsaufbau Nr. 2. Prinzipiell wie der Aufbau vom Versuch Nr. 1, nur das Wasserbad wurde ausgetauscht

in den ein Loch gebohrt wurde in das der Pointer reingesteckt werden kann (welches abgedichtet wurde). Der Pointer ist jetzt fixiert auf einer Höhe und kann nicht mehr bewegt werden. Die Datenerfassung ist in diesem Versuch etwas anders als beim ersten.
Zuerst wurde ein relativ niedriger Wasserstand realisiert so dass sich die Pointerspitze gerade so unter dem Wasserpegel befand. Der Ultraschallkopf wurde mit dem Roboter ganz leicht in das Wasser getaucht. Der Eimer wird nun so verschoben, dass gerade noch so die Pointerspitze im Ultraschallbild zu sehen war. Dies ist um einiges einfacher verlaufen als im ersten Versuch, da der Pointer sowieso schon orthogonal in den Eimer zeigt. Bei konstantem Wasserstand wurde der Eimer jetzt immer neu positioniert, sodass Bilder aufgenommen werden konnten, die die Pointerspitze in der ganzen Breite des US-Bildes zeigen. Um Bilder in einer neuen Ebene aufnehmen zu können wurde der Wasserstand im Eimer erhöht und der Ultraschallkopf wieder neu positioniert. Die Erhöhung des Wasserpegels hat zur Folge, dass sich die der Abstand von

Ultraschallkopf und Pointerspitze erhöht, da der Ultraschallkopf nur ganz leicht ins Wasser getaucht wird. Die Pointerspitze ist nun auf einer anderen Höhe im US-Bild. Auf diese Art und Weise wurden Bilder auf 5 Ebenen aufgenommen, mit 7-8 Bildern pro Ebene. Die verwendeten Einstellungen am US-Gerät waren:
Frequenz: 5MHz, Bildtiefe: 17cm.

3.4.4 Datenaufbereitung

Die Daten des Trackingsystems bezüglich der US-Halterung und des Pointers wurden mit der Software NDI Track aufgenommen. Dabei wurden gleichzeitig mit der Auslösung der Bildaufnahme des Ultraschallgerätes die Daten des Trackingsystems gespeichert. Dies geschah in Form von 5 Frames des Trackingsystems, aus denen dann für jeden einzelnen Wert der Mittelwert der 5 gemessenen Werte errechnet wurde. Ein Frame des Trackingsystems enthält die Positions- und Orientierungsinformationen zum Pointer und der US-Halterung. Auch die Ultraschallbilder mussten noch aufbereitet werden, da das Ultraschallsystem den ganzen Bildschirm bei jeder Aufnahme gespeichert hat. Relevant waren aber nur die tatsächlichen Ultraschallbilder. Deswegen wurde einmalig die Mitte der Kuvatur des Ultraschallkopfes in Pixelkoordinaten identifiziert [11]. Dieser Punkt diente dann als Ursprung des Bildkoordinatensystems, in welchen dann die Pixel gesucht wurden, die die Pointerspitze enthalten. Die aufbereiteten Daten sind dem Anhang zu entnehmen.
Die aufgenommenen Messpunkte wurden in 2 Teilmengen unterteilt, die **Fiducials** und **Targets**. Diese Unterteilung ist notwendig, damit eine Bewertung der Transformation (siehe Kapitel 4.1.1) stattfinden kann. Hierbei sind Fiducials Punkte, die in unmittelbar in die Berechnung der Transformation einfließen. Die Targets werden bei der Berechnung ausgelassen. Die Unterteilung wurde so unternommen, dass die Targets relativ zentral im Bildbereich des US-Bildes fallen und die Fiducials sich so weit und gut wie möglich verteilen.

Diese Unterteilung dient dem Zweck, bewerten zu können, wie genau ein Ziel (Target), welches sich meistens im Zentralbereich des Bildes befindet transformiert wird.

Eine Übersicht über die Verteilung von Fiducials und Targets im US-Bild befindet sich in Abb. 3.6

3.4.5 Berechnung der Transformation

Zur Berechnung der Transformation werden nun nur die Messpunkte der Fiducials benutzt. Um nun die 8 Unbekannten aus T_s und $T_{S \leftarrow I}$ zu errechnen, wird die Gleichung f_{stylus} aus dem Kapitel 2.5.3 benutzt. Diese Gleichung beschreibt den Fehler

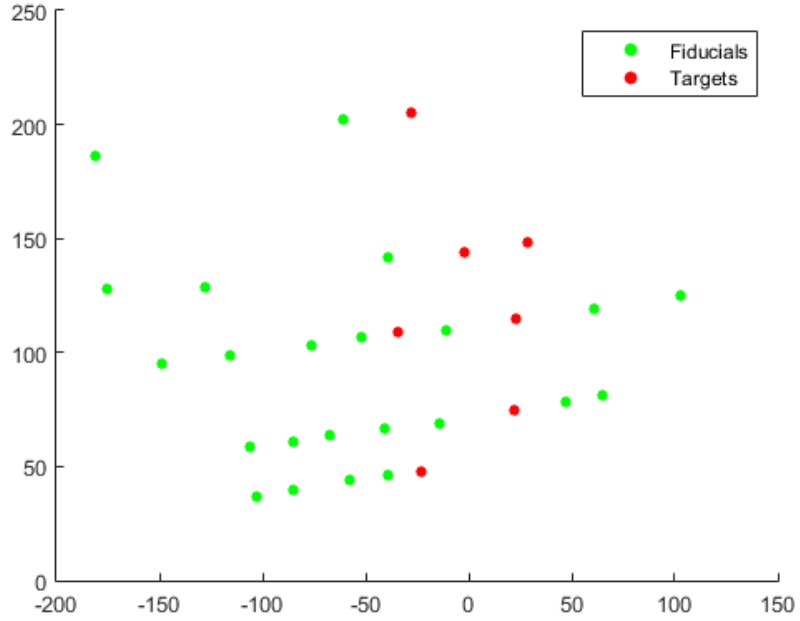

Abbildung 3.6: Verteilung von Fiducials und Targets im US-Bild-Koordinatensystem

```
1          x = lsqnonlin(@func,x0,[],[],options)
```

Abbildung 3.7: MATLAB-Code, Beispielaufruf der MATLAB-Funktion lsqnonlin zum Minimieren von nicht-linearen-Fehler-Problemen

zwischen errechneter Position und tatsächlicher Position der Toolspitze in Weltkoordinaten, noch in Abhängigkeit von den 8 Unbekannten. Dieser Fehler sollte so klein wie möglich werden. Deshalb dient die Gleichung als Basis für ein nicht-lineares (Fehler-)Minimierungsproblem. Es ist nicht linear, da zumindest 3 der Unbekannten aus $T_{S \leftarrow I}$, nämlich die Rotationswinkel α, β und γ in trigonometrischen Funktionen verrechnet werden (siehe Aufbau von $T_{S \leftarrow I}$ in 2.5.3). In der Literatur wird ein solches Minimierungsproblem mit dem Levenberg-Marquard-Algorithmus gelöst [14]. Dieser ist in der Software MATLAB implementiert und kann benutzt werden. Um den Algorithmus von MATLAB benutzen zu können, musste erst eine Funktion implementiert werden, die den Fehler jedes einzelnen Fiducials als eine Komponente in einem Vektor zurückgibt. [10]. Anschließend konnte mit dem Aufruf der MATLAB-Funktion **lsqnonlin** das Lösen des Problems angestoßen werden (siehe Abb. 3.7).

Die verwendeten Parameter hierbei sind:

@func Die zuvor implementierte Funktion, die den Fehlervektor berechnet

x0 Startvektor. Die Funktion braucht einen Startvektor für das Minimieren, der sozusagen den Anfang der Suche markiert. Das Herausfinden der Startwerte wird im Kapitel 3.4.6 beschrieben.

options Zusätzliches Feld für optionale Einstellungen, hier hat man die Möglichkeit, den verwendeten Algorithmus auf **Levenberg-Marquard** einzustellen

3.4.6 Herausfinden des Startvektors

Da für die Berechnung mit MATLAB ein Startvektor nötig ist, mussten für jede der 8 Unbekannten ein halbwegs plausibler Wert für den Anfang herausgefunden werden. α, β und γ sind die Rotationswinkel um die Koordinatenachsen. Da das Koordinatensystem des Rigid-Body aber so gewählt wurde, dass es ähnlich orientiert ist wie das des US-Bildes sollten hier relativ kleine Werte vorkommen ($\pm$ 25°). t_x ist der Abstand des Ursprungs vom Rigid-Body-Koordinatensystem zur Mitte der Kuvatur in x-Koordinaten des Rigid-Body-Koordinatensystems. Diese Differenz sollte nicht allzu groß sein, höchstens $\pm$ 3cm. Da t_x (wie die anderen beiden Translationsfaktoren auch) jedoch in mm angegeben ist, wurde hier als Startwert 30 genommen. Für t_y wurde ganz grob der Abstand vom Marker des Rigid-Body der den Ursprung markiert in die Ebene die tangential von der US-Kopf-Spitze aufgestellt wird. Diese Strecke betrug ca. 13 cm, also 130mm. Ähnlich wurde auch bei t_z verfahren. Vom Ursprungsmarker des Rigid-Body parallel zur z-Achse des Koordinatensystems bis man sich ca. in der Bildebene befindet, das waren ca. 3 cm, also 30mm.
Beim Einarbeiten in das US-System wurde durch ausprobieren herausgefunden, dass ein Pixel auf jeden Fall weniger als ein mm ist. Ein guter Startwert hier könnte 0.5 sein.

So ergibt sich als Startvektor: $x_0 = \begin{pmatrix} \alpha \\ \beta \\ \gamma \\ t_x \\ t_y \\ t_z \\ s_x \\ s_x \end{pmatrix} = \begin{pmatrix} 10 \\ 10 \\ 10 \\ 30 \\ 130 \\ 30 \\ 0.5 \\ 0.5 \end{pmatrix}$

3.4.7 Durchführung der Rechnung

Mit dem nun vorhandenen Startvektor wurde die MATLAB-Funktion lsqnonlin (siehe Abb. 3.7) bemüht. Das Ergebnis wurde als neuer Startvektor gespeichert und die Funktion erneut aufgerufen. Dies wurde in einer Schleife 1000 Mal durchgeführt. Die Ergebnisse der Rechnung befinden sich in Kapitel 4.1.

3.4.8 FRE

Der **Fiducial Registration Error**(FRE) ist ein Maß der Güte der errechneten Transformation. Die Berechnung des FRE erfolgt mit Gleichung 2.2 in MATLAB, die zugrunde liegende Punktemenge sind die Fiducials. Die Fiducials sind die Punkte, die in die Berechnung der Transformation mit eingeflossen sind. Die

3.4.9 TRE

Auch der **Target Registration Error**(TRE) ist ein Maß der Güte der errechneten Transformation. Die Berechnung erfolgt auch mit Gleichung 2.2, auch in MATLAB. Hierfür werden jedoch nicht die Fiducials sondern die Targets als Punktemenge benutzt. Der TRE zeigt, wie gut ein ein Ziel im US-Bild in Weltkoordinaten umgerechnet wird, ohne dass der Punkt in die Berechnung der Kalibrierung mit eingeflossen ist. Die Ergebnisse für den TRE befinden sich in Kapitel .

3.5 Tool Geometry Test

Der NDI 6D-Architect ermöglicht es, die Geometrie des Rigid-Body auf die Vorgaben des Herstellers (Marker Geometry Constraints, [5]) zu überprüfen, auf Basis der .rom-Datei. Dafür werden die Strecken zwischen den Markern betrachtet. Es wird jeder Marker mit jedem anderen Marker verbunden. So entstehen Segmente. Die entstandenen Segmente dienen dem Tool Geometry Test als Basis. NDI schreibt folgende Regeln in den Marker Geometry Constraints vor.

1. Die Distanz zwischen Markern sollte nich kleiner sein als 40mm
2. Die Unterschiede zwischen den Längen der einzelnen Segmente sollten mindestens 3.5mm sein.

(siehe [5]).

Das Wesentliche des Tool Geometry Test kann der Abb. 3.8 entnommen werden. Man erkennt, dass sich Marker A und C zu nah sind(weniger als 40mm) und dass die Segmente AB und BD zu ähnlich lang sind (die Differenz der Längen der beiden Segmente ist < 3.5mm), siehe Abb. 3.9.

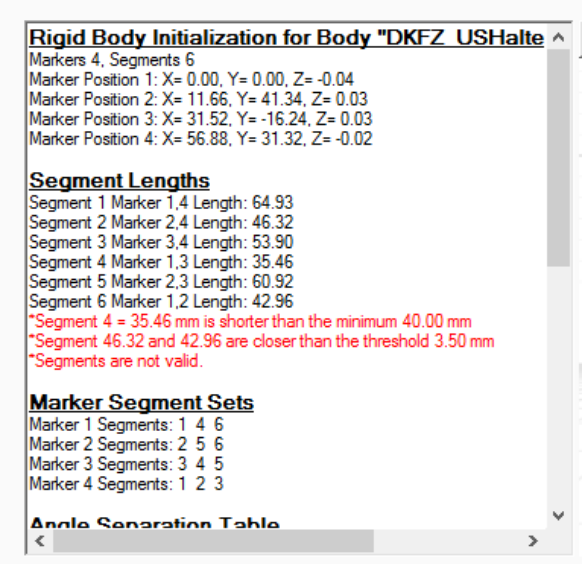

Abbildung 3.8: Ergebnisse des Tool Geometry Test, durchgeführt im NDI 6D-Architect

3.6 Anforderungen an das Plugin

Um das gewünschte Szenario ermöglichen zu können ergeben sich folgende Anforderungen an das Plugin:

1. Erkennen und verfolgen des Ultraschallkopfes im Raum
2. Erkennen des Ultraschallgerätes und das entsprechende Abgreifen der Bilder
3. Bild- und Positionsinformationen entsprechend verarbeiten
4. Die verarbeitenden Daten entsprechend darstellen

3.7 Implementierung

3.7.1 Umgebung

Das implementierte Plugin (3D-Ultrasound Plugin) besitzt diverse Verbindungen zu anderen Plugins im MITK, die wiederum die Kommunikation mit der Hardware oder die Visualisierung übernehmen. Eine Übersicht befindet sich in Abb. 3.10.

NDI Polaris Spectra Repräsentiert die Trackingkamera.

EsoateMyLab Five Repräsentiert das Ultraschallsystem.

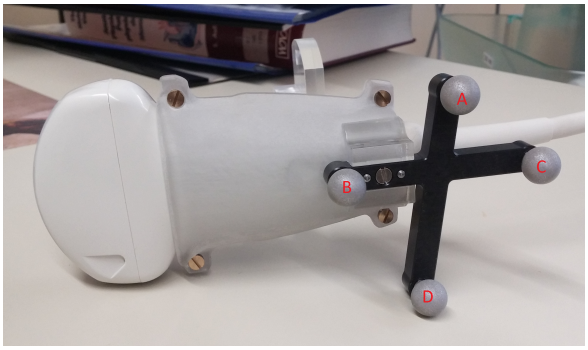

Abbildung 3.9: Die Marker des Rigid-Body mit Beschriftung. Der Marker A wurde im Tool Geometry Test mit 1 beschriftet, B mit 2 usw.

IGT Tracking Toolbox Übernimmt die Kommunikation mit der Trackingkamera.

Ultrasound Plugin Etabliert Verbindung mit dem Ultraschallsystem.

3D Ultrasound Plugin Das implementierte Plugin. Hier werden die Daten der Trackingkamera und des Ultraschallgerätes verarbeitet und das 3D-Bild wird berechnet.

Volume Visualistion Plugin Liegt ein Volumenbild vor, kann es in 3D mit dem Plugin visualisiert werden

3.7.2 Datenaustausch

Der Datenaustausch erfolgt auf drei Arten:

Hardwareverbindungen Die Verbindung mit der Trackingkamera erfolgt direkt per USB-Kabel. Das Ultraschallsystem ist indirekt mit dem System über den Videograbber verbunden (siehe Doppelstriche, Abb. 3.10)

Implizite Verbindungen durch C++ Microservices Die beiden Plugins, die die Kommunikation mit der Hardware übernehmen stellen nach erfolgreich hergestellter Verbindung die Hardware als C++ Microservice zur Verfügung. Das USNavigation Plugin greift auf diese in den darunter erstellten Plugins durch diese Microservices zu. Genauso greift das 3D-Ultrasound Plugin auf die mitk::CombinedModality die im USNavigation Plugin erstellt wurde.

Verbindug durch das DataStorage Das 3D-Ultrasound-Plugin speichert das erstellte 3D-Bild im DataStorage ab. Dort kann es vom Volume Visualisation Plugin wieder aufgegriffen werden.

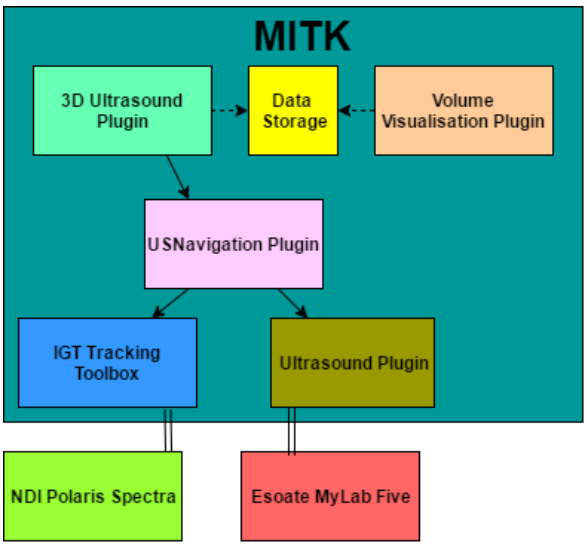

Abbildung 3.10: Übersicht über die verwendeten Plugins, dem implementiertem 3D-Ultrasound Plugin und ihre Verbindungen. Doppelstriche sind Hardwareverbindungen, durchgezogene Pfeile sind implizite Verbindungen, die durch Verwendung von C++ Microservices entstehen, gestrichelte Pfeile sind Verbindungen zum DataStorage

3.7.3 C++ Microservices in der Praxis

Das Konzept der C++ Microservices wurde im Kapitel 2.6.5 allgemein und mit einem Beispiel erläutert. Nun soll aufgezeigt werden wo die Microservices Anwendung in der Implementierung bekommen.

1. im Ultrasound Plugin
2. im Plugin IGT Tracking Toolbox
3. im US Navigation Plugin
4. im zu implementierenden 3D Ultrasound Plugin

Die ersten beiden Plugins stellen die Verbindung zur Hardware (zur Trackingkamera und damit zum getrackten Werkzeug und zum Ultraschallgerät) her. Ist die Hardwareverbindung geglückt, werden Objekte die die Hardware repräsentieren (mitk::USDevice und mitk::NavigationDataSource) in den C++ Microservices registriert. Das US Navigation Plugin holt sich diese zuvor erstellten Objekte und stellt daraus selber ein Objekt der Klasse mitk::USCombinedModality her. Dieses Objekt wird wiederum als Microservice registriert.

3.7.4 Notwendige Ausgangslage für das Plugin

Da das Plugin die Kommunikation bzw. Verbindung mit der Hardware nicht selbst übernimmt, sondern diese Aufgabe von schon bestehenden Plugins erledigt wird, müssen vor dem Benutzen des Plugins noch einige Arbeitsschritte vorgenommen werden. Diese können aus dem Diagramm in Abb. 3.11 entnommen werden. Im einzelnen erläutert sind die Schritte:

1. Verbindung zum Ultraschallgerät herstellen, in der Oberfläche des Ultrasound Plugin. Ein Objekt der Klasse **mitk::USDevice** wird hier im Hintergrund erstellt und als C++ Microservice registriert
2. Verbindung zur Trackingkamera und dem zu trackenden Rigid-Body herstellen, in der Oberfläche des Plugins IGT Tracking Toolbox. Hier wird ein Objekt einer **mitk::NavigatioDataSource** erstellt und als C++ Microservice registriert
3. Erstellen eines Objekts von mitk::USCombinedModality durch die Oberfläche des US Navigation Plugin. Die zuvor erstellten mitk::NavigationDataSource und mitk::USDevice werden hier zu einer **mitk::USCombinedModality** zusammengefügt und auch als C++ Microservice registriert.
4. Das 3D-Ultrasound Plugin initialisieren.

Hierbei können die Punkte 1. und 2. ausgetauscht werden.

3.7.5 Implementierung der Kalibrierung des US-Kopfes

Da die Kalibrierungsmatrix auch als Matrix der Gestalt 4x4 aufgefasst werden kann (Transformation in homogenen Koordinaten) wurde diese einfach in ein Datenfeld des Typs **vtkMatrix4x4** gespeichert.

3.7.6 Gewinnung einzelner 2D-Bilder mit entsprechenden Positionsinformationen

Das zentrale Datenfeld des Plugins ist eine mitk::USCombinedModality. Durch ein Objekt dieser Klasse können sowohl die aktuellen Bildinformationen des Ultraschallgerätes, als auch die aktuellen Daten des Trackingsystems zum Ultraschallkopf abgefragt werden. Kommt es nun zu einer Bildaufnahme, können beide Informationen gespeichert werden. Das US-Bild liegt als mitk::Image vor, die Daten des Trackingsystems als mitk::NavigationData. Die Daten des mitk::NavigationData-Objekts werden zunächst in eine mitk::AffineTransform und daraus wiederum manuell in eine vtk4x4Matrix

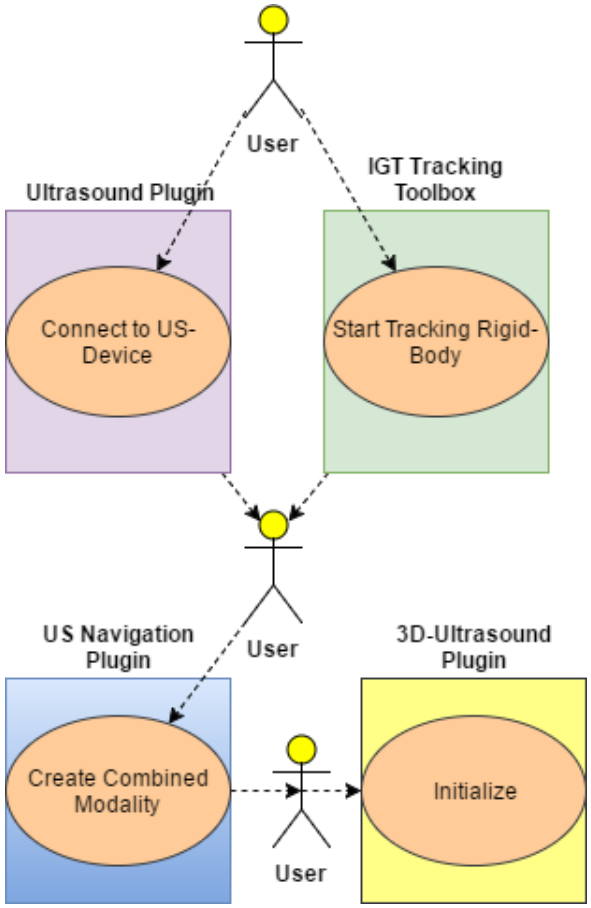

Abbildung 3.11: Diagramm welches beschreibt, wie der Nutzer die notwendige Ausgangslage für das Plugin herzustellen hat.

umgerechnet. Diese Matrix beschreibt die Transformation vom Weltkoordinatensystem in das des Rigid-Body (auf der US-Halterung). Diese Matrix wird nun (unter Beachtung der Reihenfolge der Multiplikation) mit der bereits implementierten Kalibrierungsmatrix verrechnet. Das Ergebnis ist eine Matrix, die die Umrechnung von Bildpixeln in das Weltkoordinatensystem ermöglicht. Diese Matrix wiederum wird manuell in eine mitk::AffineTransform umgewandelt und wird dann als Transformation der Geometry des gespeicherten mitk::Image gesetzt. Das aufgenommene mitk::Image hat jetzt die korrekte Geometry, die die Position, Ausdehnung und Orientierung im Raum beschreibt. Es wird nun nur lokal gespeichert und nicht dem mitk::DataStorage hinzugefügt, da das sehr die Performance belasten würde.
Beliebig viele Bilder können nun auf diese Weise aufgenommen, verarbeitet und gespeichert werden.

3.7.7 Zusammensetzten des 3D-Bildes

Sind nun alle gewünschten Schnittbilder aufgenommen worden, muss das 3D-Bild rekonstruiert werden. Zunächst werden aus allen zuvor aufgenommenen Bildern die maximalen und minimalen Koordinaten in jeder Achsrichtung identifiziert. Diese Informationen werden dazu benutzt ein ausreichend räumlich ausgedehntes, leeres 3D-Bild zu initialisieren. Anschließend werden die einzelnen Bilder nun in das leere 3D-Bild eingefügt. Dies geschieht mit Hilfe eines mitk::ImagePixelWriteAccessor und den Geometrys der jeweiligen 2D-Bilder und des leeren 3D-Bildes. Mit der Geometry eines Bildes lassen sich für jedes Pixel im Bild die entsprechenden Weltkoordinaten berechnen und umgekehrt, aus Weltkoordinaten können Pixelkoordinaten berechnet werden. Es wird nun über jedes Pixel jedes zuvor aufgenommenen Teilbildes iteriert und das folgende passiert:

1. Mit Hilfe der Geometry des 2D-Bildes wird die Weltkoordinate des Pixels (des 2D-Bildes) errechnet.
2. Die errechneten Weltkoordinaten werden nun von der Geometry des 3D-Bildes in Pixelkoordinaten des 3D-Bildes umgerechnet.
3. Diesem Pixel (im 3D-Bild) wird mit Hilfe des mitk::ImagePixelWriteAccessor der Wert des Pixels (des 2D-Bildes) gegeben.

Da vor dem Einfügen der Bilder die Ausdehnung des 3D-Bildes an die Verteilung der 2D-Bilder angepasst wurde, werden keine Pixel außerhalb des initialisierten Bildes gesetzt, keine Informationen gehen verloren. Ist die Berechnung des 3D-Bildes erfolgt, wird es dem zentralen mitk::DataStorage der MITK-Workbench hinzugefügt.

4 Ergebnisse

4.1 Kalibrierungsmatrix für den Ultraschallkopf

Die Berechnung der Kalibrierungsmatrix erfolgt nach den in 2.5 beschriebenen Grundlagen und der in 3.4 beschriebenen Methodik.

$\alpha = 18.1375$
$\beta = -0.5921$
$\gamma = -16.2693$
$t_x = -20.0765$
$t_y = 131.8941$
$t_z = 30.7289$
$s_x = 0.3441$
$x_y = 0.3543$

Setzt man die nun bekannten Faktoren in die Matrix $T_{S \leftarrow I}$ erhält man folgende Matrix

$$T_{S \leftarrow I} = \begin{pmatrix} 0.9503 & -0.2961 & -0.0966 \\ 0.3113 & 0.9132 & 0.2631 \\ 0.0103 & -0.2801 & 0.9599 \end{pmatrix} + \begin{pmatrix} -20.0765 \\ 131.8941 \\ 30.7289 \end{pmatrix}$$

Und für $T_s = \begin{pmatrix} 0.3441 & 0 & 0 \\ 0 & 0.3543 & 0 \\ 0 & 0 & 0 \end{pmatrix}$.
Wandelt man nun $T_{S \leftarrow I}$ und T_s in homogene Koordinaten um, so erhält man für die ganzheitliche Kalibrierungsmatrix

$$T^h_{kalib} = T^h_{S \leftarrow I} T^h_s = \begin{pmatrix} 0.3270 & -0.1049 & 0 & -20.0765 \\ 0.1071 & 0.3236 & 0 & 131.8941 \\ 0.0036 & -0.0993 & 0 & 30.7289 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

4.1.1 Bewertung der Matrix

Da es sich bei der Kalibrierung um einen essentiellen Teil des Entwurfes handelt, sollte diese auch bewertet werden, dies geschieht mit den schon erwähnten FRE und TRE. Die Berechnung dieser erfolgt mit Gleichung 2.2, zusätzliche Erläuterung im Kapitel 2.5.5

4.1.1.1 FRE

Der FRE wird auf Basis der Fiducials ausgerechnet.

Fiducial Nr.	FRE[mm]
1	0.5109
2	0.4961
3	2.1659
4	1.8520
5	0.8809
6	1.1720
7	1.7934
8	1.2741
9	0.2870
10	0.4427
11	1.0026
12	1.8780
13	0.4583
14	1.8904
15	0.8270
16	2.3466
17	0.9644
18	0.4191
19	1.7968
20	0.8835
21	1.4311
22	1.4449
23	2.1335
24	1.5830
25	1.9415

Das entspricht einem durchschnittlichen FRE von $\approx 1.2750mm$

4.1.1.2 TRE

Der TRE wird auf Basis der Targets ausgerechnet.

Target Nr.	TRE[mm]
1	0.8681
2	0.6921
3	1.0599
4	0.9782
5	1.4011
6	2.2214
7	2.3095

Das entspricht einem durchschnittlichen TRE von $\approx 1.3615mm$

4.2 Das 3D-Ultrasound-Plugin

4.2.1 Oberfläche

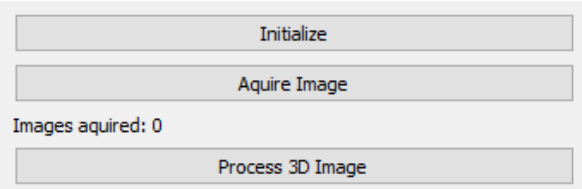

Abbildung 4.1: Die GUI des 3D-Ultrasound Plugin

Die Oberfläche des 3D-Ultrasound-Plugin bietet 3 Knöpfe (siehe Abb. 4.1). Hier hat man die Möglichkeit, die Verwendung des Plugins zu Initialisieren (Knopf Initialize), eine Bildaufnahme durchzuführen (Aquire Image) und die Berechnung des 3D-Bildes anzustoßen.

4.2.2 Visualisierung

Um sich das aufgenommene Bild in 3D anschauen zu, muss in das Plugin **Volume Visualisation Plugin** gewechselt werden. Damit kann man Bilder nicht nur entlang der 3 Schnittebenen betrachten, sondern es visualisiert Volumen in 3D. Dazu muss man das 3D-Bild im DataStorage der MITK-Workbench auswählen und eine Transferfunktion wählen, bzw. selber anpassen. Die Transferfunktion beschreibt welche Grauwerte des Ausgangsbildes welche Opazität und Farbwert in der 3D-Visualisierung erhalten. Dabei sollte die Ausgangsopazität des Grauwertes 0 auf 0 gesetzt werden, die der restlichen Grauwerte auf 1. Das hat den Hintergrund, dass alle Pixel die nicht ein Pixel eines Ultraschallbildess enthalten den Grauwert 0 erhalten. Ansonsten sind die Farben für die restlichen Grauwerte beliebig wählbar.

4.2.3 Durchgeführte Bildaufnahmen

Um die Funktionsweise des Plugins zu überprüfen, wurden 3D-Aufnahmen des vorliegenden Ultraschall-Phantoms vorgenommen. Diese wurden im Verfahren **Linear Scan** und **Fan Scan** (siehe Kapitel 2.3.1) durchgeführt.

In Abb. 4.2 erkennt man gut die parallel aufgenommenen Schichten. Abb. 4.3 zeigt den für den Fan Scan typischen Bildfächer.

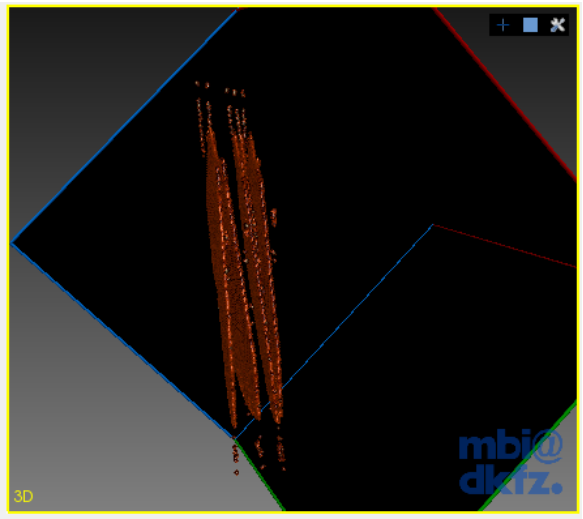

Abbildung 4.2: Mit dem Plugin durchgeführte Bildaufnahme - Verfahren Linear Scan

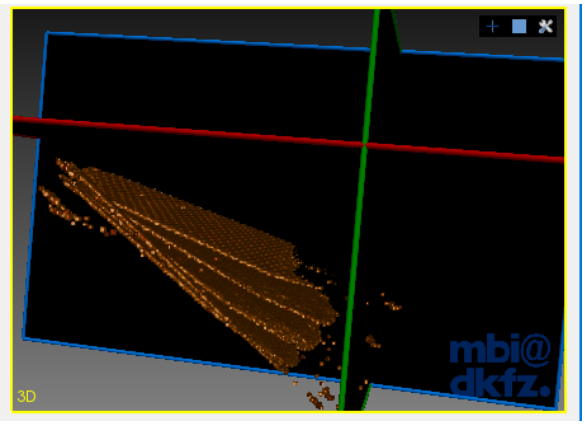

Abbildung 4.3: Mit dem Plugin durchgeführte Bildaufnahme - Verfahren Fan Scan

5 Diskussion

Das Ziel dieser Arbeit war es, mit einem vorliegenden konventionellen 2D-Ultraschallgerät in Kombination mit einem optischen Trackingsystem ein Plugin für das MITK zu entwerfen, welches Frei-Hand-3D-Ultraschallbildgebung ermöglicht. Basis hierfür ist, dass ein Ultraschallkopf im Raum von einem optischen Trackingsystem verfolgt wird. Aus den Bildinformationen des Ultraschallbildes und den Positionsinformationen des Ultraschallkopfes sollten 2D-Bilder in ein 3D-Volumen eingefügt werden. Die dafür notwendigen Grundlagen und Vorarbeitsschritte sollten beschrieben werden, ehe die auf die Implementierung des Plugins eingegangen wurde.

5.1 Kalibrierung für das optische System

Optisches Tracking basiert darauf, dass Bilder einer bekannten sichtbaren Struktur aus mehreren Winkeln aufgenommen werden und dann daraus die Position des Objektes berechnet werden kann. Damit der verwendete Ultraschallkopf im Raum verfolgt werden konnte, wurde am DKFZ für den Ultraschallkopf eine Halterung angefertigt, die den US-Kopf steif mit einem Gerüst aus Markern (Rigid-Body) verbindet. Diese Marker sollten das Tracking ermöglichen. Jedoch war die geometrische Anordnung der Marker noch unbekannt, optisches Tracking noch nicht möglich. Die Kalibrierung des Rigid-Body für das Trackingsystem erfolgte mit der Software NDI 6D Architect. Dies lief ohne Probleme ab, die Dokumentation [4] war sehr ausführlich und beschrieb alles nötige.

5.2 Kalibrierung des Ultraschallkopfes

Das optische System konnte jetzt die Halterung, die mit dem Ultraschallkopf verbunden ist verfolgen. Für ein Frei-Hand-3D-Ultraschallsystem braucht man jedoch die Position und Orientierung der Ultraschallkopfspitze. Verfahren zum Herausfinden der Umrechnung von der Position und Orientierung der Halterung zur Position und Orientierung des US-Kopfes werden in der Literatur beschrieben ([11] und [3]). Diese Umrechnung heißt auch Kalibrierung. Grob wird dabei eine Struktur im US-Bild gesucht, deren Position in Weltkoordinaten bekannt ist. Das Nachstellen der beschriebenen Versuche jedoch erwies sich zunächst als schwierig, beim ersten Versuch (Kapitel 3.4.2) stieß man auf grundlegende Probleme. Deswegen musste der verwendete Versuchsaufbau überdacht und verbessert werden. Das Resultat dieser Überlegungen war ein zweiter Versuchsaufbau (Kapitel 3.4.3). Hier konnten die zuvor aufgetretenen Probleme beseitigt werden. Das Ergebnis der Kalibrierung war eine Transformationsmatrix. Deren Güte wurde beurteilt. Dazu wurden die Fehlermaße FRE und TRE eingeführt. Es stellt sich heraus, dass die Kalibrierung zwar gelungen ist, jedoch etwas ungenauer ist, als in den in der Literatur beschriebenen Versuche. [Muratore et

al., [14]] kommen bei vergleichbaren Versuchen (30 verwendete Fiducials) zu einem FRE zwischen 0,48mm und 0,51mm und einem TRE zwischen 0,4mm und 0,65mm. Der in dieser Arbeit errechnete durchschnittliche FRE ist etwa um den Faktor 2,5 größer (1.275mm), der durchschnittliche TRE ca. um den Faktor 2 (1.3615mm). Eine mögliche Ursache hierfür könnte das Rigid-Body-Design sein. Es entspricht nicht den Vorgaben von NDI. Das könnte zur Folge haben, dass die Positionsbestimmung der US-Halterung nicht genau genug möglich ist. Auf Möglichkeiten das Rigid-Body-Design zu verbessern wird im weiteren Verlauf noch eingegangen.

5.3 Anpassung der Rigid-Body-Geometry

Der Rigid-Body der US-Halterung genügt nicht den Vorgaben von NDI. Das könnte zur Folge haben, dass das Tracken der US-Halterung nicht genau genug möglich ist. NDI beschreibt folgende Möglichkeiten (siehe [5]) um Rigid-Bodys zu verbessern.

1. Die Anzahl der Marker erhöhen
2. Die Verteilung der Marker in 2D und 3D vergrößern

Im Folgenden sollen die einzelnen Punkte diskutiert werden.

5.3.1 Möglichkeiten den Constraints zu genügen

Die Probleme bei der Tool Geometry sind, dass der Marker C zu nah an A dran liegt und die Strecken AB und BD zu ähnlich lang sind (siehe Abb. 3.9). Eine mögliche Lösung könnte sein, den Marker C etwas weiter rechts, und Marker B etwas weiter links zu platzieren. Das würde Marker C weiter von A entfernen und die Strecken AB und BD ungleich vergrößern.

5.3.2 Erhöhung der Anzahl der Marker

NDI gibt als Obergrenze für die Markeranzahl eines passiven Tools 6 an [5]. Eine einfache Erhöhung der Markeranzahl auf 6 ist aber keinesfalls leicht zu realisieren. Um die Constraints nicht zu verletzten, müssten sich alle Marker mehr als 4cm voneinander entfernt befinden und die zwischen den Markern entstehenden Strecken (Segmente) müssten alle zueinander einen Längenunterschied 3.5mm haben. Dieses unterfangen ist bei 4 Markern schon schwer genug, bei 6 Markern wäre zwar ein Tooldesign durchaus möglich, jedoch steht es immer im Konflikt mit der Benutzbarkeit. Im jetztigen Zustand soll es dafür benutzt werden, um frei Hand ein zu untersuchendes Objekt zu beschallen. Wird das Werkzeug jetzt aber länger, schwerer und größer (was bei einer

Erhöhung der Markeranzahl unausweichlich ist), so sinkt die Benutzbarkeit ab. Zudem kann es vorkommen, dass die neue Marker durch den Anwender verdeckt werden (da sie ungünstig platziert worden sind). Dann hat man ein komplexeres Tool entworfen, dessen Vorteil (mehr Marker) man aber nicht nutzen kann (da sie ständig verdeckt werden). Ein Ähnliches Problem wird man haben, wenn man das vergrößerte Tool an den Roboterarm montiert. Ist das Tool nun größer, schränkt es den Roboter in seinen Bewegungen ein, da es zu Kollisionen (mit sich selbst) kommen könnte, die das Tool beschädigen könnten.
Zusammenfassend kann man sagen, dass das erhöhen der Markeranzahl in diesem Kontext eher hinderlich wäre.

5.3.3 Die Verteilung der Marker in 2D und 3D vergrößern

Die verwendeten Marker sind in 2D durch die Kreuzform relativ gut verteilt. Sie liegen alle 4 jedoch in einer Ebene. Eine Möglichkeit hier eine Verteilung in 3D hinzubekommen könnte sein, einen oder 2 Marker in einer erhöhten Position anzubringen.

5.4 Implementieren im MITK

Die Einarbeitung in das Implementieren für das MITK stellte sich als schwierig und zeitaufwendig dar. Als Basis hierfür wurde das MITK-Git Repository und die MITK-Dokumentation auf der Homepage verwendet. Die Dokumentation vieler Klassen, aber auch vom MITK verwendeten Konzepte und Verfahren ist teilweise nicht vorhanden bzw. nicht ausführlich genug, oder für einen Einsteiger in das Thema schwer nachzuvollziehen. Nachdem jedoch die Hürden der Einarbeitung genommen wurden, kam die Bearbeitung des Problems gut voran.

5.5 Ausblick

Das Hauptziel der Arbeit, die Konzeption und Implementierung des Frei-Hand-3D-Ultraschalls für das MITK, wurde erfüllt. Es besteht eine gute Grundlage für weitere Verbesserungen, z.B. Interpolation der Bildwerte zwischen aufgenommenen Schichten um ein durchgängiges 3D-Volumen zu erlangen. Im Kontext der robotergestützten Ultraschallbildgebung könnte auch untersucht werden, ob ein Ersetzten des Trackingsystems durch Positionsinformationen des Roboters möglich wäre.

6 Literaturverzeichnis

[1] CppMicroServices: *C++ Micro Services 3.0.0 documentation.* http://docs.cppmicroservices.org/en/stable/README.html. 2017. – Letzter Zugriff: 21.02.2017

[2] Gee, Andrew ; Prager, Richard ; Treece, Graham ; Berman, Laurence: Engineering a freehand 3D ultrasound system. In: *Pattern Recognition Letters* 24 (2003), feb, Nr. 4-5, S. 757–777. – URL `https://doi.org/10.1016%2Fs0167-8655%2802%2900180-0`

[3] Hsu, Po-Wei: *Freehand three-dimensional ultrasound calibration*, University of Cambridge, Dissertation, 2008

[4] Inc., Northern D.: *NDI 6D Architect (Version 3) User Guide.* 2007

[5] Inc., Northern D.: *Polaris Tool Design Guide.* 2007. – Revision 4

[6] Manual, MITK: *CMake FAQ.* http://docs.mitk.org/2016.03/CMAKE_FAQ.html. 2016. – Letzter Zugriff: 06.03.2017

[7] Manual, MITK: *Data Management Concept.* http://docs.mitk.org/2016.03/DataManagementPage.html. 2017. – Letzter Zugriff: 28.02.2017

[8] Manual, MITK: *How to create a new MITK Plugin.* http://docs.mitk.org/2016.03/NewPluginPage.html. 2017. – Letzter Zugriff: 08.03.2017

[9] Manual, MITK: *Overview of MITK.* http://docs.mitk.org/2016.03/MitkOverview.png. 2017. – Letzter Zugriff: 08.03.2017

[10] Mathworks: *Solve nonlinear least-squares (nonlinear data-fitting) problems - MATLAB lsqnonlin.* https://de.mathworks.com/help/optim/ug/lsqnonlin.html. 2017. – Letzter Zugriff: 04.03.2017

[11] Mercier, Laurence ; Langø, Thomas ; Lindseth, Frank ; Collins, Louis D.: A review of calibration techniques for freehand 3-D ultrasound systems. In: *Ultrasound in Medicine & Biology* 31 (2005), feb, Nr. 2, S. 143–165. – URL `http://dx.doi.org/10.1016/j.ultrasmedbio.2004.11.001`

[12] MITKManual: *The Architecture of MITK.* http://docs.mitk.org/nightly/Architecture.html. 2017. – Letzter Zugriff: 21.02.2017

[13] März, K. ; Franz, A. M. ; Seitel, A. ; Winterstein, A. ; Bendl, R. ; Zelzer, S. ; Nolden, M. ; Meinzer, H. P. ; Maier-Hein, L.: MITK-US: real-time ultrasound support within MITK. In: *International Journal of Computer Assisted Radiology and Surgery* 9 (2013), dec, Nr. 3, S. 411–420. – URL `http://dx.doi.org/10.1007/s11548-013-0962-z`

[14] Muratore, Diane M. ; Galloway, Robert L.: Beam calibration without a phantom for creating a 3-D freehand ultrasound system. In: *Ultrasound in Medicine & Biology* 27 (2001), nov, Nr. 11, S. 1557–1566. – URL `http://dx.doi.org/10.1016/S0301-5629(01)00469-0`

[15] Trackingtoolbox, IGT: *The MITK-IGT Tracking Toolbox*. http://docs.mitk.org/nightly/org_mitk_views_igttrackingtoolbox.html. 2017. – Letzter Zugriff: 21.02.2017

Anhang

A Anhang

Die folgenden Tabellen enthaltenen die Messwerte des Kalibrierungsversuches. ur_z, ur_y und ur_x sind die Rotationen beschreiben die Rotationen der US-Halterung, ut_x, ut_y und ut_z, deren Translation im Weltkoordinatensystem. pt_x, pt_y und pt_z sind die Translationen der Pointerspitze im Weltkoordinatensystem. X und Y sind die Pixelkoordinaten der Pointerspitze im US-Bild.

Fid Nr.	ur_z [°]	ur_y [°]	ur_x [°]	ut_x [mm]	ut_y [mm]	ut_z [mm]	pt_x [mm]	pt_y [mm]	pt_z [mm]	x [Px]	y[Px]
1	87.72	-35.40	-138.88	24.80	47.81	-2105.31	116.41	78.71	-2219.75	-39	46
2	87.72	-35.38	-138.87	24.79	47.81	-2105.30	113.73	73.10	-2221.92	-58	44
3	87.73	-35.39	-138.89	24.78	47.81	-2105.30	111.12	63.66	-2223.13	-85	40
4	87.73	-35.39	-138.90	24.77	47.79	-2105.33	108.46	58.37	-2225.20	-103	37
5	87.86	-34.81	-138.81	19.91	47.33	-2100.41	129.61	103.70	-2209.28	47	78
6	87.88	-34.81	-138.85	19.96	47.34	-2100.33	132.40	108.90	-2207.10	65	81
7	87.87	-34.74	-138.79	19.95	47.38	-2100.29	120.09	86.63	-2216.44	-14	69
8	87.87	-34.80	-138.81	19.95	47.40	-2100.34	117.08	77.88	-2219.49	-41	67
9	87.87	-34.82	-138.81	19.94	47.39	-2100.36	114.50	68.95	-2221.46	-67	64
10	87.89	-34.79	-138.84	19.93	47.37	-2100.34	112.05	63.55	-2223.04	-85	61
11	87.88	-34.81	-138.87	19.90	47.18	-2100.30	109.77	56.32	-2224.86	-106	59
12	87.87	-34.80	-138.81	10.32	46.46	-2090.98	122.86	85.44	-2214.31	-11	110
13	87.86	-34.80	-138.82	10.32	46.47	-2090.99	133.51	106.16	-2205.35	61	119
14	87.86	-34.79	-138.79	10.31	46.47	-2090.98	139.93	118.31	-2200.20	103	125
15	87.86	-34.81	-138.80	10.31	46.46	-2090.99	118.83	72.42	-2218.23	-52	107
16	87.86	-34.79	-138.80	10.30	46.45	-2090.97	114.69	65.59	-2219.91	-76	103
17	87.86	-34.78	-138.81	10.30	46.46	-2090.98	110.50	52.59	-2225.27	-116	99
18	87.86	-34.80	-138.81	10.29	46.45	-2090.98	107.13	41.97	-2227.83	-149	95
19	87.87	-34.86	-138.73	3.31	45.81	-2084.33	123.24	75.72	-2217.00	-39	142
20	87.86	-34.86	-138.74	3.31	45.81	-2084.32	111.76	48.20	-2225.30	-128	129
21	87.84	-34.93	-138.76	3.16	45.55	-2084.26	108.52	32.24	-2229.86	-175	128
22	87.84	-34.79	-138.77	-12.84	44.32	-2068.61	140.38	106.99	-2199.56	75	217
23	87.85	-34.79	-138.77	-12.84	44.33	-2068.62	148.36	118.54	-2193.01	116	224
24	87.85	-34.81	-138.77	-12.82	44.34	-2068.63	123.48	66.03	-2213.98	-61	202
25	87.86	-34.78	-138.79	-12.82	44.33	-2068.61	110.60	27.99	-2224.75	-181	186

Abbildung A.1: Messergebnisse für die Menge der Fiducials.

Tar Nr.	ur_z [°]	ur_y [°]	ur_x [°]	ut_x [mm]	ut_y [mm]	ut_z [mm]	pt_x [mm]	pt_y [mm]	pt_z [mm]	x [Px]	y[Px]
1	87.74	-35.38	138.90	24.82	47.82	-2105.29	118.20	84.14	-2218.57	-23	48
2	87.86	-34.81	-138.80	19.92	47.34	-2100.42	125.36	96.54	-2212.75	22	75
3	87.87	-34.78	-138.82	10.32	46.47	-2090.99	127.86	95.37	-2210.17	23	115
4	87.85	-34.81	-138.79	10.31	46.46	-2090.98	121.08	77.99	-2216.53	-34	109
5	87.87	-34.87	-138.76	3.29	45.78	-2084.35	128.35	85.71	-2210.05	-2	144
6	87.86	-34.85	-138.72	3.29	45.77	-2084.34	132.96	94.64	-2206.39	29	148
7	87.85	-34.80	-138.78	-12.82	44.34	-2068.60	127.17	76.30	-2211.09	-28	205

Abbildung A.2: Messergebnisse für die Menge der Targets.

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen (einschließlich elektronischer Quellen) direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht.

_____ _____

(Ort, Datum) (Unterschrift)