RWTH

**Pflichtenheft**

|  |  |
| --- | --- |
| **Projektbezeichnung** | Medizinische-software für Operationsplanung |
| **Projektleiter** | Dr.-Ing. Matias de la Fuente |
| **Erstellt am** | 02.12.2021 |
| **Letzte Änderung am** | 31.01.2022 |
| **Status** | in Bearbeitung |
| **Aktuelle Version** | 1.1 |

**Änderungsverlauf**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Nr.** | **Datum** | **Version** | **Geänderte Kapitel** | **Art der Änderung** | **Autor** | **Status** |
| 1 | 02.12.2021 | 1.1 | Alle | Erstellung | Youssef Kharita | i.B |
| 2 | 28.01.2022 | 1.2 | Alle | Modifizierung | Youssef kharita | i.B |
| 3 |  |  |  |  |  |  |
| 4 |  |  |  |  |  |  |
| 5 |  |  |  |  |  |  |

# Inhalt

[Article I. Inhalt 2](#_Toc94281510)

[1 Einleitung 3](#_Toc94281511)

[2 Ausgangssituation (ist-Zustand) 3](#_Toc94281512)

[3 Konzept 3](#_Toc94281513)

[3.1 Ziel und Nutzen des Anwenders 3](#_Toc94281514)

[3.2 Zielgruppe 3](#_Toc94281515)

[4 Funktionale Anforderungen 3](#_Toc94281516)

[4.1 3 D-Röntgenbilder schichtweise darstellen 3](#_Toc94281517)

[4.2 3D\_bild anzeigen 3](#_Toc94281518)

[4.3 Die Bohrschablone zu fertigen 4](#_Toc94281519)

[4.4 Die Bohrlänge und Breite zu darstellen 4](#_Toc94281520)

[5 Nichtfunktionale Anforderungen 4](#_Toc94281521)

[5.1 Benutzerfreundlichkeit 4](#_Toc94281522)

[5.2 Laufzeit 4](#_Toc94281523)

[5.3 Rauschunterdrücken 4](#_Toc94281524)

[6 Entwicklungsprozess 4](#_Toc94281525)

[6.1 Systemarchitektur 4](#_Toc94281526)

[6.2 Entwicklungszyklus 5](#_Toc94281527)

[(a) Daten laden: 5](#_Toc94281528)

[(b) Bilder schichtweiser Anzeigen: 5](#_Toc94281529)

[(c) 3D Ansicht : 5](#_Toc94281530)

[7 Lieferumfang 5](#_Toc94281531)

# Einleitung

Die medizinische Versorgung der Patienten hat einen langen Weg hinter sich. In der Zeit der technologische Entwicklung wird sehr viele medizinische Software zum Nutzen der Menschen eingesetzt, da viele Operationen mit Hilfe von Software ermöglicht werden und die Planung von Operationen sich stark weiterentwickelt hat, so dass die Patienten eine sichere medizinische Versorgung erhalten. Zu diesem zwick wir entwickeln ein chirurgisches Planungssystem für die Dekompression von atraumatischen Hüftkopfnekrosen.

# Ausgangssituation (ist-Zustand)

Bislang wurden die Röntgenbilder ausgedruckt und die Messungen von Hand vorgenommen, was fehlerhaft sein kann. Inzwischen gibt es ein Softwaresystem, das dreidimensionale Röntgenbilder Schicht für Schicht darstellt und ein 3-D-Bild erzeugt.

# Konzept

## Ziel und Nutzen des Anwenders

Eine benutzerfreundliche Oberfläche und ein einfach zu bedienendes Programm, um die Planung von Operationen zu möglichen

## Zielgruppe

Das Projekt richtet sich an Krankenhäuser, um ihren Chirurgen die Möglichkeit zu geben, die Dekompression von atraumatischen Hüftkopfnekrosen besser zu planen. Im Projekt werden die Röntgenbilder schichtweise dargestellt und ermöglichen die Darstellung der Bohrlänge und des Bohrdurchmessers. Da die Bohrschablone für jeden Patienten individuell ist, wird der Plan dafür mit unserer Software berechnet und für den 3D-Druck vorbereitet.

# Funktionale Anforderungen

## 3 D-Röntgenbilder schichtweise darstellen

Lokalisierung des Bereichs der atraumatischen Hüftkopfnekrose, der gesäubert wird

## 3D\_bild anzeigen

eine 3d\_Darstellung des Hufgelenks

## Die Bohrschablone zu fertigen

Die Bohrschablone dient der Schonung der Verletzungsoberfläche und ermöglicht eine genaue Bohrlänge. Sie ist für jeden Patienten individuell und wird für jede Operation erstellt. Mit der Software digitalisieren wir die Berechnung der Oberfläche von Ihnen

## Die Bohrlänge und Breite zu darstellen

Mittels einer kleinen Bedingung (Mausklick) wird die Rohrlänge und die Breite ebenfalls berechnet.

# Nichtfunktionale Anforderungen

## Benutzerfreundlichkeit

Das Programm sollte einfach zu bedienen sein und nicht von einem Experten bedient werden müssen. Außerdem sollte es eine Schnittstelle haben und mit der Maus gesteuert werden können.

## Laufzeit

Die Laufzeit ist angemessen, und das Programm dauert im Durchschnitt 1-2 Sekunden, um seine Funktion zu erfüllen. Bei mansche Funktionalitäten wie Display Schablone dauert das Programm mehr als 1-2 Sekunden, da es um eine Umwandlung der daten von 40\*40\*40 zu 512\*512\*512

## Rauschunterdrücken

Bei der Darstellung der Bilder ist darauf zu achten, dass die Knochenoberfläche frei von Rauschen ist und dass die zu messenden Punkte genau eingefügt werden. Diese gesteht durch die Filterung das Bild.

# Entwicklungsprozess

## Systemarchitektur

Das System wird von zwei Hauptklassen entwickelt:

[**Widget**](file:///C:\Users\youss\Desktop\code\sweidmt_ws2022_youssef_kharita\doc\html\class_ui_1_1_widget.html) : die Funktionen zur Steuerung der Schnittstelle enthält und darstellen das berechnete Ergebnis

[**CTDataset**](file:///C:\Users\youss\Desktop\code\sweidmt_ws2022_youssef_kharita\doc\html\class_c_t_dataset.html) : die Funktionen der zu daten hochladen und bearbeiten und die notwendige Berechnung für weitere Funktionalen

Dabei wird CTDataset in dem Klasse Widget inkludiert und die Klasse Widget ein Objekt der Klasse CTDatatset erstellt. Das klasse Widget unterstütz ein Graphical user interface (GUI) damit wird ein Benutzer freundlichen Programms unterstützt

Weitere Bibliotheken: Eigen zu Vektor und Matrix Berechnung, Qfile zum Laden und Speichern von datat, QMessageBox zu informieren des benutzter über Fehler Eingaben, QMouseEvent für Mouse Koordinaten Ablesung.

Diagram

Description automatically generated Die Member Funktionen und Attribute sind im beigefügten doxygen

## Entwicklungszyklus

### Daten laden:

Das Funktion **load\_3d()** im Widgetwird mittels der bushbutton\_3d angetrieben dabei wird der Benutzer um einen raw Datei type angefragt. Nachdem erhalt der Path von image wird die funktion **load()** in CtDataset angerufen. Dort wird die konsistent von Datei Size geprüft und Fehler code zurückgegeben. Wenn die Angabe stimmt, wird die daten in einem Array Größe 512\*512\*130 Geschrieben was an sich entspricht 130 Röntgen Bilder mit Größe für jedes Bild 512\*512 am Ende wird die **filter\_bild()** in CtDataset angerufen. Dieser Funktion schaut für jedes Voxel die benachbarten Voxel und setzt das Voxel auf dem mittel wert. Am Ende wird die **updateSliceView()** aufgerufen, die zuständig für Bilder schichtweiser anzuzeigen.

### Bilder schichtweiser Anzeigen:

In dem **updateSliceView()** wird ein Image 512\*512 erstellt um die später zu anzeigen. Der Benutzer bestimmt den Start und Fenster breite mittleres eine Slider in die GUI denn werden die grau werte das Image angepasst an einem bestimmten Fenster mittels die Funktion **windowing()**  im CtDataset, in diesem Funktion werden die Werte in dem bestimmten Fenster betrachtet und zurück zum Bereich 255 umgerechnet was menschliche Auge sehen kann. Zwischen durch wird die werte um einen schwellen wert gesucht, um sie rot zu anzeigen. Letztlich sind die Werte des Bildes, die an sich die Farben sind in lable\_2D angezeigt. Der Benutzer kann, die die schichte wechseln dadurch ein vertikal Slider.

### 3D Ansicht:

Nachdem drücken der **Render\_3D** Botton wird die Funktion **render3D()** aufgerufen dabei wird die Funktion **rotate()** in CTDataset aufgerufen. In dem Funktion rotate() wird ersten die daten um einen bestimmten x, y, und z-winkel gedreht. Die Vorgehensweise dafür ist das ein Rotationmatrix in CTDataset als member Attribut der Klasse CTDataset zu erstellen. Die Rotationmatrix wird durch die Funktion **updateaxis()**  im Widget aktualisiert. Die Winkel Angabe erfolgt durch horizontal Slider und nach jeder Eingabe wird die Rotationmatrix aktualisiert. Die daten werden gedreht in dem die x, y z in dem Array als Vektor betrachtet und den Vektor, um die Rotationmatrix zu drehen, Dabei wird die wert an dem Punkt x, y, und z vor drehen an dem Punkt x, y, und z nach dem drehen geschrieben. Nach rotation wird die **calculatedepthpuffer()** angerufen, die die tiefen karte nach bestimmten schwellenwert rechnet und in einer Array 512 \* 512 geschrieben als member Attribut der Klasse CTDataset gespeichert. Nachdem Erstellung der tiefenkarte wird die Funktion **renderdepthbuffer()** gerufen der die unterschied zwischen jeder benachbarten Voxel in dem tiefen karte berechnet und die zu darstellen vorbereitet. Die Werte werden in einer Array shadedBuffer, das als member Attribut der Klasse Widget ist. Am Ende wird die Funktion **displayImage3D()** aufgerufen der die schadedBuffer Werte, die die Farben entstehen in einem Image Objekt schreibt und die in lable\_3d anzeigt.

A picture containing text, electronics, calculator

Description automatically generated

### Cropping das 3d image

Das wird durch Ablesen der Maus Koordinaten im lable\_3d, dabei handelt sich um einem Mauspressung und Freigabe. Das geschehen mittels Funktionen **mousePressEvent(), mousereleaseEvent()**. Durch confirm area Botton wird die gewällte Fläche als blaue box auf dem 3Dimage angezeigt. Am Ende mit dem confirm cropping botton wird die gewällte Fläche ausgeschnitten. Die Coping geschieht indem das in rotate() nur die gewällt Fläche von dem Array betrachtet. Durch undo cropping wird wieder der gesamte Array betrachtet was an sich die ganze Bild rendert.

### Bohrlange lange und breite:

Zuerst wird der Benutzer um einem Start punkt gefragt was durch ein Maus klick in dem 3d Ansicht berechnet und um einem end punkt gefragt was durch ein Maus klick in dem 2d Ansicht berechnet danach wird die Funktion **bohren()** im CTDataset aufgerufen der den bohr Vektor rechnet und normiert danach werde auf jedem Punkt des Vektor in einem Würfel Bereich die punkte die in einem Kogel liegen. Der mittel Punkt des Kogel ist die Punkte auf dem Vektor und diese Punkte wird im Image um einen explizierte wert die nicht in dem werte beriech liegt. Beim Rendern werden diese Punkte blau farbig angezeigt. Die loschen von dem Bohren erfolgt durch die zurück setzten das Daten auf dem ursprünglichen Daten.

A picture containing diagram

Description automatically generated

### Schablone:

Nach der Bestätigung der erst und end Punkt wird die daten so gedreht, dass den bohr Vektor auf dem z Axis liegt. Das Verfahren geschieht indem erstens den Winkel zwischen den bohr Vektor und die ebene YZ gerechnet und die daten um y Axis gedreht wird, zweiten wird der Winkel zwischen den bohr Vektor und die z Axis gerechnet und die daten diesen Winkel, um die x Axis zu drehen. Somit steht der bohr Vektor auf die z Axis. Da die Schablone 10 mm in dem Konchen geduckt ist, wird die Bereich ab dem Start Punkt in 10 mm gesucht auf dem z Axis und in dem Bereich -20, 20 um den x ,y Axis, am Ende soll die Schablone ein bohr loch um den bohr Vektor das geschieht in dem das expliziert wird die durch bohren gesetzt wird die Schablone entlang zu lehren. Zu eine Einsicht der Schablone, ermöglicht die Funktion **displayschablone()** die Umsetzung von Schablone Größe 512 \*512 \*512 diese ermöglicht die Schablone Schichtweise zu schauen dabei handelt sich in den Display beim rot um Substanz und grün um die Vakuum. Außer dem es gibt die Möglichkeit die Schablone als Raw Datei zu speichern

### **Save plan:**

In dieser Funktion wird die plan und Schablone in einer Text Datei gespeichert, dabei wird der Start und End Punkt, die Bohrung durchmesse und lange und die Schablone daten als 0,1 wobei 0 Wo Vakuum und 1 Wo Substanz sind gespeichert

# Lieferumfang

die Software wird mitgeliefert, Sie benötigen lediglich einen Computer und einen 3D-Drucker