BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences

Beuth Hochschule für Technik

Abschlussarbeit Master Medieninformatik

mARt: Interaktive Darstellung von MRT Daten in AR

Viola Jertschat

betreut von

Prof. Dr.-Ing. Kristian Hildebrand

9. Januar 2019

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Verwandte Arbeiten

1.1 Ein Abschnitt

Kapitel 2

Grundlagen

2.1 MRT

Die Abkürzung MRT steht für Magnetresonanztomographie. Das Verfahren, das auch Kernspintomografie genannt wird, wird in der Radiologie verwendet, um Abbildungen der inneren Organe zu erzeugen. Dazu werden Magnetfelder erzeugt,

2.2 MRT und Schlaganfälle

2.3 Volume Rendering

! Nur Grundlagen, nicht Implementierung

3D Texturen

Raymarching

Transferfunktionen

Kapitel 3

Anforderungsanalyse

3.1 Interviews

Zur Bestimmung der Anforderungen, die von der Anwendung erfüllt werden sollen wurden iterativ mehrere Interviews mit dem betreuenden Radiologen der Charité geführt.

Im Rahmen des ersten Interviews wurde zunächst der Nutzen der Anwendung beschreiben. Diese soll Neurologen im Bereich der Schlaganfallprävention unterstützen. Wie im Kapitel Grundlagen beschrieben untersucht der Neurologe dabei die Abbildungen des Gehirns eines Patienten, die durch den MRT-Scan erzeugt wurden. Auffällige Bereiche, die auf einen Schlaganfall hindeuten könnten werden dabei markiert. Diese Darstellung soll nun inklusive des markierten Bereichs dreidimensional dargestellt werden. Durch die zusätzliche Dimension soll sowohl für Ärzte als auch Patienten die Ausmaße des betroffenen Bereich verdeutlichen, was die Einschätzung eines Schlaganfallpatienten verbessern könnte. Weiterhin könnte man anhand des Modells auch Voraussagen von Therapieerfolgen anschaulich demonstrieren.

Die Anwendung soll dabei als Prototyp dienen, die die Möglichkeiten, sowie das Potential für die Verwendung in der Praxis testet. Im Vordergrund steht hierbei die Darstellung in 3D, inklusive des markierten Bereichs.

Aus dem ersten Interview ließen sich außerdem folgende Anforderungen ableiten: Die aus dem Interview ermittelten Anforderungen wurden in der folgenden Tabelle aufgelistet. Zur besseren Referenzierung wurden sie jeweils mit Bezeichner versehen. Weiterhin wurde die Priorität jeder Anforderungen auf niedrig, mittel, oder hoch geschätzt.

Anhand dieser Anforderungen wurden User Stories formuliert.

Anforderungsbezeichnung	Anforderung	Priorität
A01	3D Darstellung der MRT Daten	hoch
A02	Einblenden des gekennzeichneten Bereichs	hoch
A03	3D-Darstellung des Gehirns, die auch Schlaganfallsbereich in 3D Erkennbar werden lässt	hoch
A04	Scrollen durch Schichten der 3D-Darstellung auf mindestens einer Achse	mittel
A05	Erkennbarkeit der Struktur inneren Struktur des Gehirns (entsprechend der MRT-Bilder)	hoch
A06	Scrollbare 2D Darstellung der MRT-Bilder auch mindestens einer Achse	mittel
A07	Manipulation der Darstellung ähnlich wie bei einem Hologramm.	niedrig
A10	Interaktionselemente sollten die Darstellung nicht verdecken	niedrig
A11	Scrollen durch Verwendung eines Scrollrads	niedrig
A12	Auswahl verschiedener MRT-Sequenzen (falls vorhanden)	niedrig
A13	Unterstützung von nifti-Daten	mittel
A14	Verschiedene Ansichten können nebeneinander dargestellt werden. (z.B. vor und nach Therapie)	mittel

Tabelle 3.1: Durch Interview bestimmte Anforderungen.

Story-Bezeichnung	User Story	Anforderungsreferenz
U01	Als Nutzer möchte ich ein 3D-Modell des gescannten Gehirns sehen, um ...	A01
U02	Als Nutzer möchte ich den gekennzeichneten Bereich innerhalb des Gehirn sehen können, um einzuschätzen wie groß der Bereich tatsächlich ist.	A02
U03	Als Nutzer möchte ich den gekennzeichneten Bereich ein- und ausblenden können, um mich auf diesen, oder die Darstellung an sich konzentrieren zu können.	A02
U04	Als Nutzer möchte ich auch dann noch die Strukturen des Gehirns erkennen, wenn der gekennzeichnete Bereich eingeblendet ist.	A03
U05	Als Nutzer möchte ich eine möglichst genaue und gut erkennbare Abbildung der MRT-Bilder, damit ich eine Vorstellung habe, wie das Gehirn des Patienten aussieht.	A05
U06	Als Nutzer möchte ich durch das 3D-Modelll scrollen können, um es vergleichbar mit der 2D Darstellung zu verwenden.	A04
U07	Als Nutzer möchte ich die MRT-Darstellungen frei im Raum bewegen können, um sie meiner Position im Raum anzupassen.	A07
U08	Als Nutzer möchte ich die MRT-Darstellungen skalieren können, um die Darstellung gut zu erkennen.	A08
U09	Als Nutzer möchte ich die MRT-Darstellungen drehen können, um den besten Blickwinkel auf den für mich relevanten Bereich zu bekommen.	A09
U10	Als Nutzer will ich, dass die Darstellung nicht von anderen Elementen verdeckt wird, damit ich sie uneingeschränkt sehen kann.	A10
U11	Als Nutzer möchte ich mit einem Scrollrad durch die Darstellung scrollen können, um genaue Kontrolle darüber zu haben, welche Schichten angezeigt werden.	A11
U12	Als Nutzer möchte ich alle vorhandenen MRT-Sequenzen sehen und zwischen ihnen wählen können, damit ich alle notwendigen Informationen zu dem Scan nutzen kann.	A12
U13	Als Nutzer möchte ich Dateien im nifti-Format in der Anwendung verwenden können, damit ich sie nicht vorher umwandeln muss.	A13
U14	Als Nutzer möchte ich aus aus verschiedenen Darstellungen wählen können, die nebeneinander angezeigt werden, um direkte Vergleiche zwischen diesen ziehen zu können.	A14

Tabelle 3.2: Aus Anforderungen abgeleitete User Stories.

Kapitel 4

Konzept

Im folgenden werden Konzepte diskutiert, um die in Kapitel 3 herausgearbeiteten Anforderungen zu erfüllen.

4.1 3D Darstellung

- Da das innere der 3D Darstellung erkennbar sein soll, muss eine semi-transparente Darstellung erzeugt werden, die sowohl die Form des Gehirns abbildet, als auch die innere Struktur - Relevant sind nur die Pixel, die das Gehirn darstellen. Der Bereich darum (Schädel und Hintergrund) muss gefiltert werden. Das gewünschte Ergebniss wäre, wenn das Gehirn frei im Raum ßchwebt"(Bei Ray Casting noch mal erwähnen?)

- Daten liegen dreidimensional in Schichten vor - verschiedene Möglichkeiten zur 3D Visualisierung - - Voxel - - Volume Rendering (Ray Casting) - - Marching Cubes - - ...?

- Ist es notwendig bzw. nützlich ein Mesh zu generieren? - Gut für Darstellung der Gehirnform. Relevant ist aber vor allem das Ïnnere"des Gehirns, da sich dort der markierte Bereich befindet. - Marching Cubes eignen nich nicht, um innere Struktur Darzustellen. Eine Transparente Ansicht würde das Modell als innen "gleich"darstellen - Das Mesh müsste kontinuierlich angepasst werden, um jeweils andere Schichten beim scrollen anzuzeigen.

- Voxel könnten auch das Innere des Gehirns darstellen. Um ein deutliches und hochaufgelöstes Modell zu erhalten währen allerdings viele Voxel notwendig. Performance? - Referenzen

- Im Fokus der Darstellung, soll das Innere des Gehirns stehen. Ein Mesh, dass die äußere Form beschreibt ist also nicht sinnvoll, zumal keine der Anforderungen Funktionen beschreibt, für die ein Mesh nötig wäre (z.B. Kollsion (Unity))

Wie in dem Kapitel 1 beschrieben, gibt es bereits viele Lösungsansätze zur 3D Darstellung von MRT-Bildern. Obwohl verschiedene Methodne zum Einsatz kommen, ist der am weitesten verbreitete Ansatz der des Volume Rendering. Dies hat die folgenden Gründe: - - ...

Diese Vorgehensweise ist somit am besten für die Implementierung der Anwendung geeignet. Diese wird im Kapitel 5 genauer beschrieben.

Anforderungen an Shader??

4.2 Darstellung des gekennzeichneten Bereichs

U02 U03

4.3 Unterstützung von Dateiformaten

Die User Story U13 verlangt nach einer Unterstützung des nifti-Dateiformats.

nifti-Dateien, sind Bilddateien, die oft in der Medizin verwendet werden. Sie speichern Bildsequenzen und eigenen sich somit für MRT-Bilder. Ein weiteres für diesen Zweck weitverbreitetes Dateiformat ist DICOM. ...

Es ist weiterhin sinnvoll gängigere Dateiformate, wie JPEG oder PNG zu unterstützen.

Wie in 3 beschrieben, soll es sich bei mARt in erster Linie um einen Prototyp handeln. Der Datensatz, der dargestellt werden soll ist gering. Um die Komplexität und den Umfang der Anwendung möglichst gering zu halten kann darauf verzichtet werden, dem Nutzer die Möglichkeit zu geben, aus verschiedenen Dateitypen oder Datensätzen zu wählen. Es ist ausreichend, wenn der entsprechende Datensatz vor dem Build gewählt wird.

Da die Darstellung und interaktion eines Datensatzes also im Vordergrund stehhen soll, ist es sinnvoll, die Dateien in ein Format umzuwandeln, das einfacher zu verarbeiten ist (JPEG, PNG). Hierzu kann ein externes Tool verwendet werden (ImageJ Bezug in Implementierung).

4.4 Interaktion

U06 Als Nutzer möchte ich durch das 3D-Modelll scrollen können, um es vergleichbar mit der 2D Darstellung zu verwenden.

U07 Als Nutzer möchte ich die MRT-Darstellungen frei im Raum bewegen können, um sie meiner Position im Raum anzupassen.

U08 Als Nutzer möchte ich die MRT-Darstellungen skalieren können, um die Darstellung gut zu erkennen.

U09 Als Nutzer möchte ich die MRT-Darstellungen drehen können, um den besten Blickwinkel auf den für mich relevanten Bereich zu bekommen.

U10 Als Nutzer will ich, dass die Darstellung nicht von anderen Elementen verdeckt wird, damit ich sie uneingeschränkt sehen kann.

U11 Als Nutzer möchte ich mit einem Scrollrad durch die Darstellung scrollen können, um genaue Kontrolle darüber zu haben, welche Schichten angezeigt werden.

U12 Als Nutzer möchte ich alle vorhandenen MRT-Sequenzen sehen und zwischen ihnen wählen können, damit ich alle notwendigen Informationen zu dem Scan nutzen kann.

U14 Als Nutzer möchte ich aus aus verschiedenen Darstellungen wählen können, die nebeneinander angezeigt werden, um direkte Vergleiche zwischen diesen ziehen zu können.

4.5 Endgerät

- Hololens oder HTC Vive? - Leistung - Interaktionsmöglichkeiten - Tragekompfort - ...

Kapitel 5

Implementierung

5.1 mARt in Unity

5.1.1 2D Szenario

- Prototyp für hololens Demo - Deckt Hologram Interaktionen ab

- Aufbau Szene - Funktionsweise der GameObjekte/ Skripte

5.1.2 3D Szenario

- Grundlage: GitHub Projekt - Aufbau Szene - Funktionsweise der GameObjekte/ Skripte

5.1.3 Interaktion

5.2 Volume Rendering

Wie in Kapitel **??** beschrieben, wurde der 3D-Darstellung des Gehirns mit Hilfe von Volume Rendering umgesetzt.

Im Kapitel **??** wurde der theoretische Vorgang dieser Technik beschrieben. Dieses Kapitel fokussiert sich auf die Implementierung der einzelnen Schritte.

Die volumetrische Darstellung der Gehirns wird im Grunde genommen mit nur 3 Skripten erzeugt. Zuerst wird in VolumeRendering.cs das Mesh des Würfeln generiert. Hier werden auch die Parameter geupdated, die für das Rendering relevant sind, wie z.B.die 3D-Textur oder Farbe, sowie die Parameter, die durch die Nutzerschnittstelle manipuliert werden können. Die Parameter werden an den Shader übergeben, in dem das Rendering definiert ist. Der Cg/HLSL Code, der den Vertex- und Fragmentshader implementiert ist dabei in ein eigenes Script ausgelagert.

Im Fragmentshader wird zunächst ein Strahl definiert, der von dem aktuell betrachteten Vertex aus von der Kameraposition in die Welt "geschossen"wird. In einem selbst definierten struct werden die maximalen und minimalen Werte definiert, aus denen sich die Eckpunkte des dargestellten Würfeln zusammensetzten. Anschließend wird geprüft, ob der Strahl den Würfel schneidet.

Um die Schnittpunkte des Strahls zu ermitteln nimmt man an, dass die sechs Seiten des Würfels auf jeweils sechs Ebenen liegen, wobei davon zwei immer parallel sind. Zuerst werden jetzt alle Schnittpunkte des Strahls mit diesen Ebenen berechnet und dann geprüft, ob die Schnittpunkte innerhalb des Würfels liegen. Der Würfel wird durch zwei Eckpunkte beschrieben. Da der Würfel Koordinaten von -0,5 bis 0,5 hat können wir hierfür die jeweils kleinsten und größten Koordinaten nutzen. Die Schnittpunkte des Strahls, mit den x-, y- und z-Ebenen ergeben sich durch das Umstellen der Formel, die einen Strahl beschreibt:

$p = r_{Ursprung} + t * r_{Richtung}$

$r_{Ursprung}$ ist dabei der Ursprung des Strahls und $r_{Richtung}$ seine Richtung. p ist ein Punkt auf dem Strahl und t ein Parameter, der bestimmt wie weit der Punkt vom Ursprung entfernt ist.

Um die Schnittpunkte mit den Ebenen zu erhalten wird die Formel nach t umgestellt:

$t = (p - r_{Ursprung})/r_{Richtung}$

Für p werden jeweils die beiden Eckpunkte des Würfels eingesetzt, die den Würfel beschreiben. Dadurch sind in den zwei dreidimensionalen Vektoren t_{unten} und t_{oben} insgesamt 6 Schnittpunkte mit den Ebenen bekannt. Zwei auf jeder Ebene. Durch den Vergleich der t-Vektoren wird festgelegt, welcher Eckpunkt (und damit welche Ebene) weiter vorne liegt. Jetzt muss bestimmt werden, ob diese Schnittpunkte sich innerhalb des Würfels befinden. Dazu werden jeweils die x-, y- und z-Werte der t-Vektoren untereinander verglichen. Für den näher gelegenen t-Vektor wird der maximale bestimmt, für den weiter entfernten der minimale Wert bestimmt. Ist der Wert des näheren ts größer als der des entfernten, liegt der Schnittpunkt nicht in dem Würfel. Andersherum tut er es.

Die beiden t-Werte werden als t_{nah} und t_{fern} gespeichert. Mit dem Ursprung des Strahls und t_{fern} werden Anfang, Ende und die Länge des Strahls berechnet. Mit Hilfe der Länge kann ermittelt werden um wie weit pro Iteration am Strahl entlang gegangen werden soll. Dadurch wird der Strahl nur bis zu seinem Austritt aus dem Würfel abgetastet.

In einer for-Schleife wird jeder Strahl nun abgetastet. In jeder Iteration wird jeweils ein Punkt betrachtet. Der Punkt verschiebt sich entlang des Strahls um die zuvor berechnete Distanz. Für jeden Punkt werden zuerst die uv-Koordinaten berechnet, da sonst nur ein viertel des MRT-Bildes dargestellt würde. Für die Koordinaten werden dann die jeweiliges Isowerte aus der 3D-Textur gelesen, die zuvor mit den MRT-Bildern befüllt wurde.

Hierbei werden lediglich die uv-Koordinaten als Indices für die Textur verwendet. Der Isowert ist dabei im Alphakanal der Textur gespeichert. Da es sich nicht um eine Farbe sondern nur einen Grauwert handelt, können die anderen Farbkanäle der Textur mit der ?Magnitude? des jeweiligen Pixels befüllt werden. Darauf wird in der Sektion Transferfunktion genauer eingegangen. Der Isowert wird außerdem noch mit der Intensität multipliziert, die der Nutzer beeinflussen kann. An dieser Stelle wird aber auch geprüft, ob der betreffende Punkt überhaupt zu sehen ist, oder aufgrund der verschiebbaren Schichten nicht sichtbar sein sollte. Dazu wird zuerst der aktuell betrachtete Punkt mit der Rotationsmatrix des Modells multipliziert. Der Punkt wird dann mit den minimalen und maximalen x-, y-, und z-Werten der verschiebbaren Schicht verglichen. Das Ergebnis des Vergleichs wird dabei in einer Variable gespeichert. Ist der Punkt kleiner als das Minimum oder größer als das Maximum wird 0 gespeichert, ansonsten 1. Die beiden Werte werden anschließend mit dem Isowert multipliziert.Ist einer der Werte null, ist auch der ermittelte Isowert null, was im Alphakanal totale Transparenz bedeutet.

Dies tritt ein, da der Isowert zunächst für alle Farbkanäle verwendet wird, um eine neue Farbe zu deklarieren.

An dieser Stelle wird über die Transferfunktion der entsprechende Farbwert aus der zugehörigen Textur gelesen. Dazu wird der Isowert als Index verwendet. Die Funktionsweise und Implementierung der Transferfunktion wird in der Sektion **??** beschrieben. Die Transferfunktion wird nur abgerufen, wenn der Isowert nicht 0 ist, da sonst die Transparenz überschrieben würde.

Der Alphawert der so erhaltenen Farbe wird noch einmal halbiert, um die Darstellung semitransparent erscheinen zu lassen.

Schließlich wird der erhaltene Farbwert mit den vorhergehenden verrechnet. Die Komposition erfolgt dabei von vorne nach hinten, da der Strahl in dieser Richtung abgetastet wird, wie folgt:

$\hat{C}_i = (1 - \hat{A}_{i-1})C_i + \hat{C}_{i-1}\ \hat{A}_i = (1 - \hat{A}_{i-1})A_i + \hat{A}_{i-1}$

Wobei $\hat{C}_i$ die Farbe und $\hat{A}_i$ die Transparenz der Farbe des vordersten Voxels ist. Wenn diese Farbe einen vorher definierten Schwellenwert überschreitet wird die Schleife abgebrochen. Der Schwellenwert kann vom Nutzer manipuliert werden und bestimmt die ?Helligkeit? der Darstellung.

Die Farbe wird schlussendlich noch auf einen Wert zwischen 0 und 1 festgesetzt und mit der Farbe der Maske verrechnet.

- Transferfunktion: Erstellen einer Textur

- Illumination: Gradient vector = normal - Gradient:

5.2.1 Ray Casting

5.2.2 Transferfunktion

5.2.3 Illumination

Kapitel 6

Evaluation

6.1 Ein Abschnitt

Kapitel 7

Fazit

7.1 Ein Abschnitt