Medizinische Bildverarbeitung - X-Ray Viewer

Eldin Ramic, Alexander Straube
Hochschule München
München, Deutschland
e.ramic@hm.edu, straube@hm.edu

I. Aufgabenstellung

Im zweiten Laboratorium war das Ziel einen DICOM (Digital Imaging and Communications in Medicine) Volumenbilddatensatz einzulesen, zu verarbeiten und auf unterschiedlicher Art darzustellen.

Zuerst soll ein einzelner Schnitt (Slice) des Datensatzes dargestellt werden, wonach der komplette generierte Volumendatensatz als Gittermodell (Wireframe) angezeigt werden soll. Hierbei ist der Detailgrad des Gittermodells zu hoch und kostet viel Performance, sodass das Deteillevel der Daten zu reduzieren ist.

In der vierten Aufgabe soll der Datensatz mit einer Oberfläche dargestellt und entsprechend beleuchtet werden.

Zuletzt soll der Halterungskäfig aus den Bilddaten entfernt werden, sodass nur noch die Maus zu sehen ist.

II. Volumenbilddatensatz

Der gegebene Datensatz umfasst 385 DICOM Dateien, die jeweils Daten, wie das Pixel Array, das Pixel Spacing, Image Position Patient oder andere Header und Meta Informationen, enthalten.

Das Pixel Array ist ein Numpy Array und enthält die Pixel Daten des CT Bildes im Hounsfiled Format. Mit der Hounsfield-Skala wird in der Computertompgraphie (CT) die Abschwächung von Röntgenstrahlung in Gewebe beschrieben und in Graustufenbildern dargestllt. [1]
Das Pixel Spacing ist der physikalische Abstand im Patienten zwischen der Mitte jedes Pixels, angegeben durch ein numerisches Paar. Diese Information ist wichtig für die Darstellung des Datensatz in einem 3D Gittermodell.
Das Image Position Patient Attribut gibt die Bildposition im 3 dimensionalem Raum an. Die x-, y- und z-Koordinaten geben den Mittelpunkt des ersten übertragenen Voxels in der linken oberen Ecke des Bildes an. [2]

Im Histogramm 1 wird die Häufigkeit von bestimmten Hounsfield Units (HU) in allen Bildern des Volumendatensatzes in Relation gesetzt.

Tabelle I
Substanz - Hounsfiled-Unit Relation

Substanz	**HU**	**–**	**Substanz**	**HU**
Luft	-1000	–	Muskeln	+40
Fett	-120	–	Kontrast	+130
Wasser	0	–	Knochen	+400

Das Histogramm 1 verdeutlicht, dass Pixel mit einem HU von -1000 am Häufigsten vorkommen, was die Substanz Luft repräsentiert. Die Tabelle I zeigt die verschiedenen Substanzen, die durch HU definiert werden.

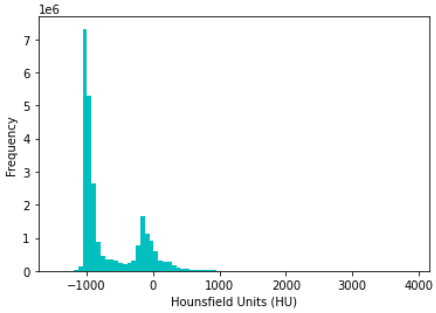

Abbildung 1. Häufigkeit von HU aller Bilddaten

III. Einlesen der Daten

Im ersten Schritt sollen die DICOM Daten aus dem Data-Verzeichnis eingelesen und in entsprechenden Datnestrukturen gespeichert werden. Mithilfe des Pydicom Python Package [3] können über den Pfad des Data-Verzeichnis alle DICOM Dateien eingelesen, und in einer Liste von FileDatasets gespeichert werden. Dabei wird die Acquisition Number aus den Header Informationen verwendet, um die Schnittbilder in die richtige Reihenfolge zu bringen.
Bei der Erstellung einer eigenen Liste mit allen Numpy Arrays, werden die Bilder zusätzlich um 180° gedreht, da ein Fehler bei der Erstellung des Volumendatensatzes auftrat.

IV. Visualisierung der Schnittbilder

Abbildung 3 zeigt das Schnittbild an der 150. Stelle im Bilddatensatz auf der linken Seite mit der Colormap ”boneünd auf der rechten Seite mit der Colormap ”gray”.

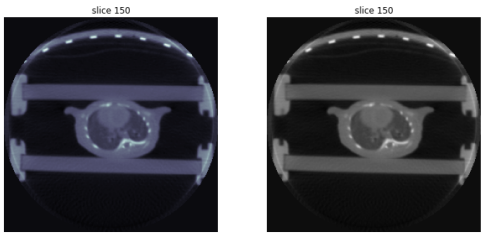

Abbildung 2. 150. Schnittbild des Volumendatensatzes

Abbildung 3 zeigt neun weitere Schnittbilder des Datensatzes, sodass man eine 3 dimensionale Abbildung einer Maus antizipieren kann.

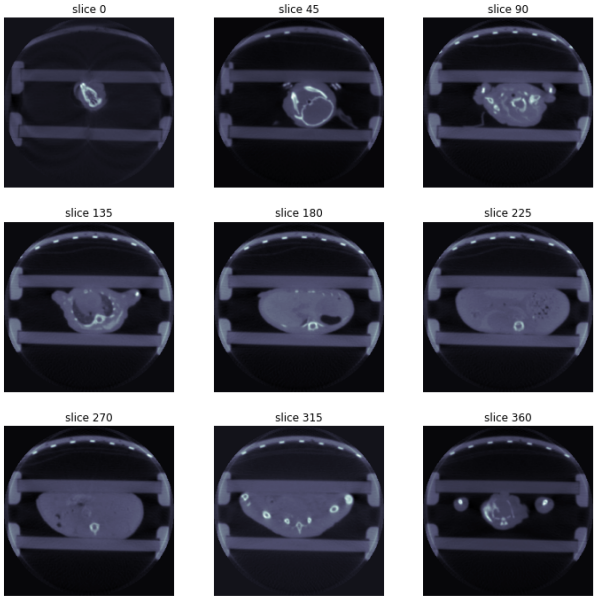

Abbildung 3. weitere Schnittbilder des Volumendatensatzes

Zusätzlich haben wir eine interaktive Plot Methode implementiert, bei der man über eine Slide-Bar alle Bilder hintereinander anschauen kann. Der zugehörige Code befindet sich im Repository als dicom_animation Methode.

V. Darstellen des Volumendatensatzes als Wireframe

In der dritten Aufgabe ist es das Ziel den Volumendatensatz als Gittermodell anzeigen zu lassen. Dafür gibt es in Python verschiedene Herangehensweisen und Bibliotheken, die auch ein interaktives Gittermodell ermöglichen. Im ersten Ansatz haben wir über den Marching Cube Algorithmus [4] versucht den Volumendatensatz in einem 3D Gittermodell darzustellen. Die skimage und matplotlib Bibliothek bieten die nötigen Funktionalitäten, sodass wir mit den Bilddaten und einem Threshold für die Hounsfield-Unit eine 3D Darstellung erzeugen können.
In Abbildung 4 wurde im linken Bild der Threshold auf 700 gesetzt. Da in der Tabelle I ein HU Wert ab 400 nur noch Knochen ist, sieht man in dem Bild fast nur das Skelett der Maus. Zudem wurde die facecolor auf weiß und die edgecolor auf blau gesetzt. Im rechten Bild werden die Oberflächen mit einer anderen Farbe beleuchtet und ein Threshold von 0 wurde gewählt. Dadurch sieht man alle Substanzen ab Wasser nach Tabelle I und zusätzlich Ränder des Scans und den Halterungskäfig. Zudem ist besitzt das Modell eine höhere Transparenz.

Für das Verständnis für die Rekonstruktion eines 2D Bilddatensatzes mit 3D Volumeninformationen über Pixel Spacing, hat uns ein wissenschaftliche Artikel [5] geholfen und kann zum Nachschlagen des Konzepts dienen.

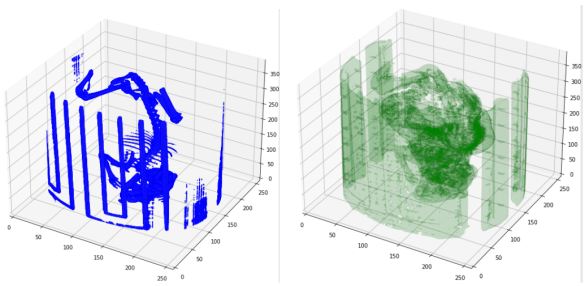

Abbildung 4. 3D Gittermodell mit unterschiedlichen Thresholds

VI. Darstellen des Volumendatensatzes mit Beleuchtung und Oberfläche

Das Ziel der vierten Aufgabe war es den Datensatz mit einer Oberfläche darzustellen und diese entsprechend zu beleuchten. Für diese Aufgabe haben wir die SimpleITK Bibliothek [6] verwendet, welches ein multi-dimensionales Analyse Toolkit mit sich bringt und Dicom Series in einem interaktiven 3D Gittermodell darstellen kann.
Abbildung 5 zeigt zwei Screenshots von interaktiven 3D Gittermodellen mit unterschiedlicher Parametrisierung. Im rechten Bild wird als Colormap wieder bone verwendet und es ähnelt eher einer 3D Punktwolke. Im rechten Bild werden Oberflächen berechnet und die Beleuchtung wird auf Grün gesetzt.

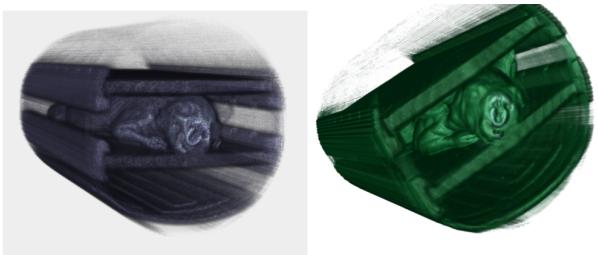

Abbildung 5. Interaktive 3D Gittermodelle mit Oberflächen und Beleuchtung

Abbildung 6 zeigt weitere, verschiedene Bilder von der interaktiven 3D Umgebung mit unterschiedlicher/m Beleuchtung, Transparenz, Blickwinkel und Threshold.

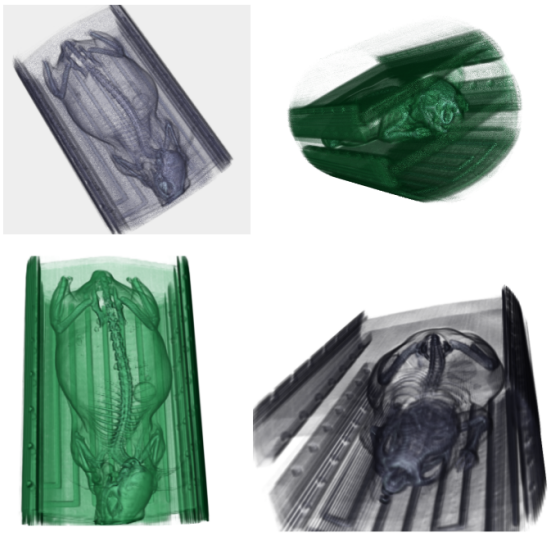

Abbildung 6. Interaktive 3D Gittermodelle mit unterschiedlicher Parametrisierung

VII. Entfernung des Halterungskäfigs aus den Bilddaten

In allen bisherigen Bildern ist immer der Halterungskäfig der Maus mit zu sehen. Daher soll in der letzten Aufgabe die Halterung herausgefiltert werden, sodass nur noch die Maus sichtbar ist. Ziel ist es die Pixel zu finden, die zu der Halterung gehören und deren HU Wert auf -1000 zu setzen.

Im Folgenden wurde versucht ein Algorithmus zu entwickeln, der alle Dimensionen des Volumens nutzt und nicht nur auf Schnittbilder einer bestimmter Orientierung. Hierbei wurde über eine Ähnlichkeitsmetrik bezüglich der Voxel überlegt, um Voxel mit gleichem oder ähnlichem HU-Wert zu segmentieren. Da die Maus teilweise die gleichen HU-Werte wie der Halterungskäfig hat, werden bei diesem Verfahren auch Teile der Maus mit segmentiert. Aus diesem Grund muss bei der Detektierung des Käfigs auch auf die z-Achse geachtet werden, da der Käfig die breiteste Struktur im Bild und im 3D-Modell ist. Idee war es anliegende Voxel auf ihren HU-Wert zu analysieren und so entsprechend den Käfig herauszufiltern. Leider hat dieser Ansatz bisher keine verwertbaren Ergbenisse geliefert, sodass wir auf eine andere Herangehensweise gewechselt haben, welche überflüssige Strukturen, wie den Halterungskäfig aus den Bildern entfernt.

Der gewählte Ansatz ist nicht sehr dynamisch und statisch auf den gegebenen DICOM Datensatz angepasst. Dabei wurden alle 2D-Bilder durchiteriert, um eine Region of Interest (ROI), die Maus im Zentrum des Bildes, auszuschneiden und in einem eigenem Numpy Array zu speichern. Danach muss die ausgeschnittene Maus in ein neu erstelltes Numpy Array mit der selben Dimension, wie die Originalbilder kopiert werden. Das neue Bild besitzt dann die HU-Werte der Maus und alle umliegenden Pixel haben den HU-Wert -1000. In Abbildung 7 kann man im Vergleich zu Abbildung 4 erkennen, dass nur noch die Maus sichtbar ist und alle überflüssigen Strukturen, wie der Halterungskäfig, herausgefiltert wurden.

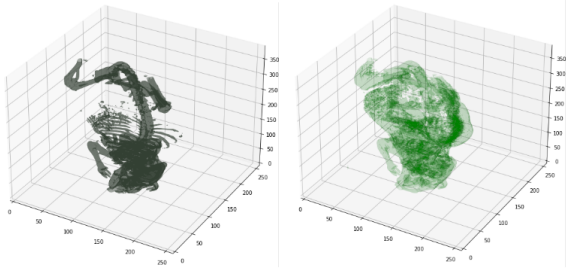

Abbildung 7. 3D Wireframe mit gefilterten Strukturen durch Ausschneiden der Maus

In einer letzten Herangehensweise wird versucht die Kanten des Käfigs in den Bildern durch die Hough-Transformation [7], und indirekt durch den Canny-Edge -Detector [8], zu bestimmen und danach den HU-Wert auf -1000 zu setzen. Dafür kann entweder die SimpleITK Bibliothek oder OpenCV herangezogen werden. Um das Ergebnis in einem Gittermodell von Matplotlib darzustellen, haben wir uns für OpenCV entschieden, sodass wir gut mit Numpy Arrays arbeiten können. Wenn nun noch die zwei vertikale Schnittebenen berechnet werden, ist die Maus ausgeschnitten und alle Voxel um die Maus herum haben den HU-Wert -1000. Abbildung 7 zeigt das Ergebnis.

VIII. Optimierungsmöglichkeiten und Erweiterungen

Der Volumenbilddatensatz wurde erfolgreich eingelesen, verarbeitet, visualisiert und der Halterungskäfig entfernt. Hier liegt auch das größte Optimierungspotential, da die Eliminierung hauptsächlich über die 2-dimensionalen Bilder stattgefunden hat. Hier könnten weitere und effizientere Verfahren zur Segmentierung oder Eliminierung entwickelt werden. Man könnte beispielsweise ein Neuronales Netz darauf trainieren die Halterung und alle weiteren überflüssigen Strukturen aus dem Volumenbilddatensatz zu entfernen.

Literatur

[1] "Hounsfield-skala — Wikipedia, the free encyclopedia," 2022. [Online; accessed 20-Juni-2022].

[2] "Pixel spacing — dicom standard browser," 2022. [Online; accessed 20-Juni-2022].

[3] "Pydicom python package - working with dicom," 2022. [Online; accessed 20-Juni-2022].

[4] "Marching cubes — Wikipedia, the free encyclopedia," 2022. [Online; accessed 20-Juni-2022].

[5] C. F. A. A. Fajar, R. Sarno, "Reconstructing and resizing 3d images from dicom files," *Journal of King Saud University - Computer and Information Sciences*, 2022.

[6] "Simpleitk," 2022. [Online; accessed 20-Juni-2022].

[7] "Hough-transformation — Wikipedia, the free encyclopedia," 2022. [Online; accessed 22-Juni-2022].

[8] "Canny-edge-detector — Wikipedia, the free encyclopedia," 2022. [Online; accessed 22-Juni-2022].