**Projekt: Medizinische Bildverarbeitung**

**Lebersegmentierung in**

**kontrastverstärkten MR-Sequenzen**

|  |  |
| --- | --- |
| Robert Brüggemann  OvGU Magdeburg  Studiengang (…)  Matrikelnummer (…)  robert.brueggemann@st.ovgu.de | Thorsten Lucke  OvGU Magdeburg  Studiengang CV-B  Matrikelnummer 207515  thorsten.lucke@st.ovgu.de |

**Kurzfassung**

Die vorliegende Arbeit beschäftigt sich mit der Segmentierung der Leber aus gegebenen medizinischen Daten.

Hierbei werden die gesamten Entwicklungsprozesse von den gegebenen Daten zur fertigen Segmentierung umrissen und erläutert als auch die einzelnen Komponenten des fertigen Programms vorgestellt.

Des Weiteren findet eine kritische Betrachtung der Ergebnisse statt samt Herausstellung von Verbesserungsansätzen.

|  |
| --- |
| Dieses Dokument entstand im Rahmen der Veranstaltung „Medizinische Bildverarbeitung“, Sommersemester 2018, Otto-von-Guericke-Universität Magdeburg. |

**1. Einleitung/Motivation**

Ziel unserer Anwendung ist die Vorplanung minimal invasiver perkutanter Punktions- und Ablationsverfahren, die häufig unter Zuhilfenahme radiologischer Bildgebung durchgeführt werden und komplementär zu Tumor- und Chemotherapie eingesetzt werden

können.

Diese Vorarbeit wird auf diagnostischen MR-Sequenzen durchgeführt, bei denen ein gadoliniumhaltiges, leberspezifisches Kontrastmittel „Gd-EOB-DTPA“ verwendet wird. Hierdurch bieten sich Vorteile wie eine starke Signalanhebung des Lebergewebes sowie eine bessere Konturierung und Definition der Zielläsionen. [1]

Unser Ziel nun ist es, mithilfe automatischer Verarbeitung dem behandelnden Mediziner/Radiologen das Lebergewebe aus dem kontrastverstärkten

Bildmaterial herauszusegmentieren, um eine zielgerichtetere und sicherere Behandlung zu ermöglichen.

**2. Vorarbeit**

**2.1 Voraussetzungen und Qualitätsmaße**

Gegeben sind uns elf Datensätze im DICOM-Format [2] mit jeweils 70 bis 100 Schichten kontrastmittelverstärkter MRT-Bilddaten sowie die jeweiligen sogenannten „Ground-Truth“-Daten, mithilfe derer wir den Dice-Koeffizienten als eines von zwei Qualitätsmaßen unseres Algorithmus berechnen können. Als zweites Qualitätsmaß sollen uns Hausdorff-Distanzen dienen.

**2.2 Auswahl der Bibliotheken**

Aufgrund der Vertrautheit beider Teammitglieder mit der sehr bequemen Programmiersprache Python [3] wird diese als Basis für die Entwicklung gewählt.

Es finden sich hierbei eine Vielzahl an Packages, mit denen die Arbeit vonstattengehen kann.

Für die Arbeit mit DICOM-Datensätzen verwenden wir das Paket pydicom [4], welches das Einlesen und Lesen der gegebenen Daten ermöglicht.

Des Weiteren spannen wir die OpenCV-Bibliothek [5] in unsere Entwicklung ein, da es eine Vielzahl Möglichkeiten zum Umgang mit und Manipulation von Bilddaten bietet, die wir an der einen oder anderen Stelle verwenden werden.

Aber auch Pythons Bibliotheken matplotlib, numpy oder scipy werden wir für unseren Algorithmus verwenden.

**2.3 Überlegungen zum Aufbau des Algorithmus**

Zum Erreichen der Segmentierung entscheiden wir uns für das grundlegende Verfahren des Region-Growing. Wir wissen, dass dieses Verfahren, welches umliegende Pixel auf einem Ähnlichkeitskriterium bewertet und somit entscheidet, ob es sich innerhalb derselben Region befindet, das Problem birgt, „auszulaufen“, d.h. über den Rand der tatsächlichen Region hinaus Pixel fälschlicherweise der Region hinzufügt. Ein Problem, um das wir uns zu gegebener Zeit kümmern werden, aber möglicherweise durch eine Art von Bildvorverarbeitung, Kantenverstärkung, Rauschunterdrückung o.ä., abschwächen können.

Ein weiteres Problem stellt die Auswahl des ersten Startpunktes dar. Des Punktes, von dem das Region-Growing auf der ersten gegebenen Schicht beginnt. Wir entscheiden uns, hierfür eine Benutzerinteraktion zu implementieren, wodurch wir den Startpunkt durch das Auge des behandelnden Arztes erhalten. Ebenso werden wir die Möglichkeit bieten, einen Punkt explizit außerhalb der Leber anzugeben, um unseren Region-Growing-Algorithmus zu optimieren. Auch hierzu an anderer Stelle mehr.

Da der Arzt nicht für jede der bis zu 100 Schichten manuell Startpunkte festlegen soll bzw. kann, benötigen wir ebenso eine Funktionalität, automatisch Startpunkte für die nächste Schicht zu kalkulieren.

Ein Problem, das sich mit dem Übertragen einer Segmentierung auf eine nächste Schicht ergibt ist, dass auch Fehler in die nächste Schicht übertragen werden.

Um diesem Effekt entgegenzuwirken und dennoch nicht zu viel Rechenleistung einzubüßen, entscheiden wir, speziell für die ersten Schichten sprichwörtlich alle Geschütze aufzufahren, um eine möglichst gute Segmentierung ohne Ausläufer zu erreichen, damit sich möglichst wenige bis keine Fehler in Folgeschichten übertragen.

Zu guter Letzt benötigen wir eine Main-Funktion und für die Ermittlung der Qualität unseres Algorithmus ebenso Funktionalitäten, die unter anderem unsere Ergebnissegmentierung mit der der „Ground-Truth“-Daten abgleichten.

**3. Umsetzung des Algorithmus/der Pipeline**

Im Folgenden sollen Funktionalitäten der einzelnen Komponenten näher erläutert werden.

**3.1 Benutzerinteraktionen**

Hier wird mittels Protokollierung von linken Mausklicks eine Liste mit Koordinaten generiert, die als Startpunkte für die Segmentierung der ersten Schichten dienen. Auch wird die Eingabe eines Punktes gefordert, der außerhalb der Leber liegt, um eine bessere Differenzierung zu erreichen. Genaueres dazu in Abschnitt 4.

Um dem Benutzer etwas zu geben, auf das er klicken kann, zeigen wir ihm in einem Fenster die Bilddaten der zuerst zu bearbeitenden Schicht an.

Da wir die Funktionen zum Umgang mit Bildern der OpenCV-Bibliothek entnehmen, offenbarte sich das Problem, dass die Anzeige (sowie auch die Filter) lediglich auf richtigen Bildern definiert waren, nicht aber auch einfachen Arrays.

Um dieses Problem zu umgehen, speichern wir die Bilddaten mittels Matplotlib als png-Bild zwischen und zeigen dieses dann in einem von OpenCV generierten Fenster an. Da sich die Bildmaße hierdurch nicht ändern, liegt auch keine eventuelle Koordinatenverzerrung vor und wir können mit der Liste von Startpunkten ohne Probleme weiterarbeiten.

**3.2 Die ersten Schichten**

Unter der Annahme, dass sich Fehler der vorigen Segmentierung auf die nächste übertragen, entschieden wir uns, der Berechnung der ersten drei Schichten deutlich mehr Bedeutung zuzutragen, um eine stabiles Fundament für die nachfolgenden Berechnungen zu erreichen.

Somit implementierten wir ein quasi-Voting-Verfahren, bei dem Punkte nur in Abgleich mit den anderen Segmentierungen übernommen wurden. Dieses Vorgehen ermöglichte bereits zu Beginn ein radikales Unterdrücken des Effekts des Auslaufens beim Region-Growing und bot uns einen guten Start in die Segmentierung der nächsten Schichten.

Eine Betrachtung, inwiefern uns dieser Bedeutungszumaß Vorteile erbracht hat, folgt im Abschnitt 4.

**3.3 Errechnung neuer Startpunkte**

Gegeben sind die Segmentierung und die Bilddaten der aktuellen Schicht, obwohl wir die eigentliche Kalkulation lediglich auf der Segmentierung ausführen.

Unter der Annahme, dass sich die Leber von einer zur nächsten Schicht nicht grundlegend verändert, implementieren wir einen Algorithmus, der mit einem Filterkernel über die gegebene Segmentierung iteriert und neue Startpunkte in seiner Mitte bestimmt.

Das tut er genau dann, wenn jeder Pixel des Kernels ein Teil der segmentierten Leber ist. Somit umgehen wir das Problem, dass wir einen Punkt außerhalb des Leberrandes wählen könnten, wenn wir einfach jeden Punkt der Segmentierung übernehmen würden.

Wir errechnen das Produkt aller Werte der Pixel. Da Werte außerhalb der Segmentierung 0 sind, muss das Ergebnis, wenn jeder Pixel innerhalb der Leber befindlich ist, ungleich 0 sein.

Als angemessene Kernelgröße erschien uns 11x11. Ungerade, um einen exakten Mittelpunkt zu haben.

Leider bergen diese vielfachen Iterationen durch den Kernel und das Bild gravierende Performanzprobleme, denen wir später auch den Abschnitt 4 widmen.

**3.4 Der Region-Growing-Algorithmus**

Hier soll nicht nochmals die Grundlage des Region-Growing-Algorithmus erläutert werden. Wir nehmen an, dass dieser grundlegende Algorithmus, der ausgehend vom gegebenen Startpunkt bzw. von den gegebenen Startpunkten eine wachsende Region generiert, geläufig ist.

Zur Optimierung der Performanz standen wir am Ende aber vor der Herausforderung, passende Parameter für diese Art von Bilddaten einzustellen und so die Ergebnisse des Algorithmus, auf die wir ebenfalls in Abschnitt 4 eingehen. so zu verfeinern.

**3.5 Abgleich mit „Ground-Truth“-Daten**

Hier erfolgt der Abgleich mit den Datensätzen der „Ground-Truth“. Der Dice-Koeffizient wird berechnet. Entsprechend unseren Vorgaben müssen unsere Segmentierungen im Durchschnitt einen Dice-Koeffizienten von 0,7 oder besser über alle elf Datensätze erreichen.

**3.6 Vorverarbeitungen auf den Bilddaten**

Zu Beginn des Projektes erschien es uns praktisch unausweichlich, eine Bildvorverarbeitung zu implementieren, um den Effekt des Auslaufens beim Region-Growing zu unterdrücken und die Effektivität des Algorithmus allgemein zu verbessern.

Tatsächlich zogen wir diverse Filter zur Kantenverstärkung oder Rauschunterdrückung aus der Bibliothek scipy in Erwägung. Nach einigen Testphasen kamen wir jedoch zum überraschenden Ergebnis, dass wir ganz ohne Bildvorverarbeitung bereits hervorragende Ergebnisse bezüglich unserer Vorgaben erzielten und diese durch eine etwaige Vorverarbeitung sogar schlechter wurden.

Wir entschieden uns, für unseren Algorithmus auf eine Vorverarbeitung gänzlich zu verzichten und den Fokus auf das Finden passender Parameter für den Region-Growing-Algorithmus und der Funktion zur Errechnung neuer Startpunkte zu legen.

**3.7 Die main-Funktion**

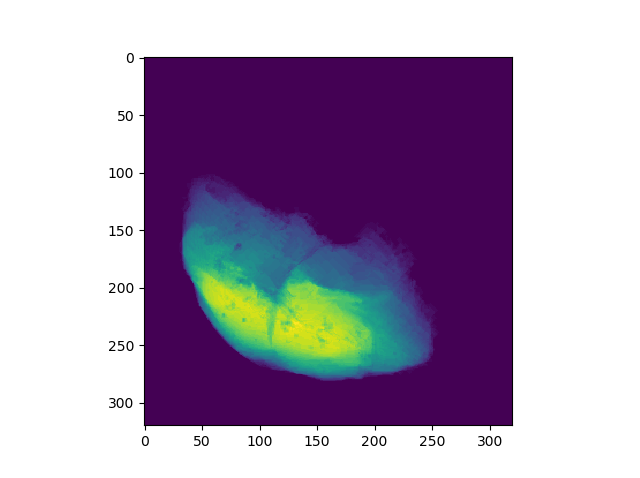
Alle Teilfunktionen werden hier verbunden und der allgemeine Programmablauf definiert: Zunächst die Benutzerinteraktion für die ersten Startpunkte, dann die Errechnung der ersten Segmentierungen, anschließend der Region-Growing-Algorithmus über die verbleibenden Schichten.

Zu guter Letzt werden die Qualitätskriterien errechnet ebenso wie die Hausdorff-Distanzen.

**4. Ergebnisse/Fazit**

**4.1 Ergebnisse**

Mit den aktuellen Ansätzen und Konfigurationen erreichen wir bei allen Datensätzen in der Regel solide einen Dice-Koeffizienten von über 0,7. Wann genau es zu Ausreißern nach unten oder gar nach oben kommen kann und inwiefern dies abhängig vom Benutzer sein kann wird nachfolgend in Abschnitt 4.2 erläutert.



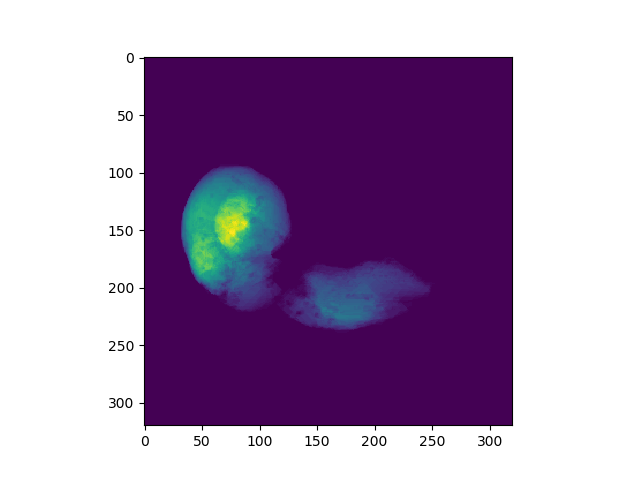
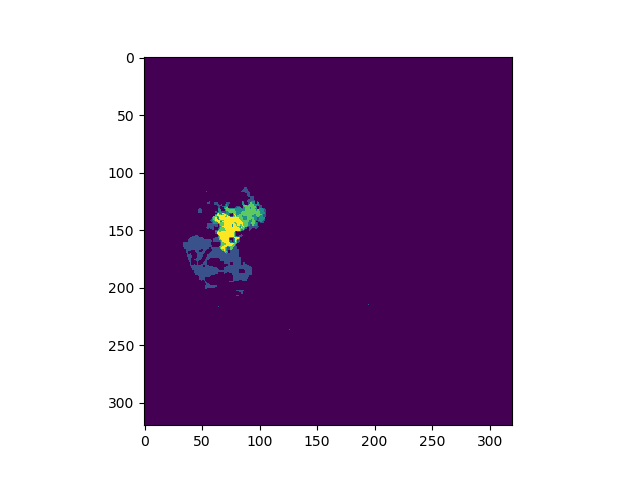
*Segmentierung des Datensatzes 9 mit einem Dice-Wert von 0,91*

Auf einer einfachen Laptop-CPU werden durchschnittlich ein bis zwei Minuten für die Segmentierung eines gesamten Datensatzes benötigt. Verbesserungsvorschläge folgen im Abschnitt 4.3.

Besonders interessant ist, dass die ursprüngliche Bedeutung, die wir der Segmentierung der ersten Schichten zugemessen haben, beinahe fehl am Platze war. Tatsächlich waren die Ergebnisse der Segmentierungen der ersten Schichten oft schlechter als die der Schichten, die folgten.

**4.2 Die Bedeutung des „Outsiders“**

Es stellte sich heraus, wie gravierend die Wahl eines Punktes außerhalb des Lebergewebes die Qualität der Segmentierung beeinflussen kann.



*Zwei Segmentierungen von Datensatz 2.*

Die Grafiken veranschaulichen diesen Effekt. Während im linken Bild ein schlecht gewählter „Outsider“-Punkt verwendet wird, liefert das rechte Bild mit einem guten Punkt deutlich bessere Ergebnisse.

Der „Outsider“-Punkt repräsentiert den Grauwert, ab dem ein Punkt als definitiv nicht mehr zum Lebergewebe zugehörig eingestuft wird. Ist dieser Grauwertunterschied zu knapp, und werden zum Beispiel durch Rauschen auch niedrigere Grauwerte innerhalb des Lebergewebes erreicht, so wird der Effekt des Auslaufens nicht mehr eingedämmt und die Ergebnisse fallen deutlich schlechter aus.

Eine Gratwanderung – der Benutzer muss den „Outsider“-Punkt so wählen, dass der Grauwert optimal Grauwerte abschneidet, die nicht mehr zur Leber gehören, aber nicht zu knapp, dass entweder ganze Teile der Leber weggeschnitten werden oder die Leber viel zu großflächig erkannt wird.

**4.3 Weitere Probleme und Verbesserungsvorschläge**

Verbesserungspotenzial hat ganz klar die Laufzeit des Programms, die erheblich unter der durch die vielen verschachtelten Iterationen komplex geratenen Berechnung der neuen Startpunkte der neuen Schichten leidet.

Überlegungen zur Parallelisierung und Multithreading standen im Raum, wurden aber aus Zeitgründen nicht näher betrachtet. Ob diese Ansätze einen Vorteil brächten, bliebe in der Zukunft zu bewerten.

Zur weiteren Verbesserung der Ergebnisse bleibt zu überlegen, ob eine etwas ausgefeiltere Bildvorverarbeitung als die eher trivialen von uns gewählten Ansätze Vorteile bringen würde.

Ein Zuschnitt der Bilddaten auf eine sogenannte Region of Interest würde ebenfalls für den Benutzer definitiv uninteressante Bereiche auslassen und nicht zuletzt auch die Laufzeit unter Umständen erheblich verbessern.

Der Code liegt derzeit ebenso in noch eher unbereinigter Form vor, durchwachsen von evolutionären Spuren und Qualitätsmetriken.

**4.4 Fazit**

**5. Referenzen**

[1] Tim König, Johannes Steffen: „Projekt: Medizinische Bildverarbeitung – Lebersegmentierung in kontrastverstärkten MR-Sequenzen“ im Rahmen der Lehrveranstaltung „Medizinische Bildverarbeitung“, Sommersemester 2018, OvGU Magdeburg

[2] DICOM : https://www.dicomstandard.org/

[3] Python: https://www.python.org/

[4] Pydicom: https://github.com/pydicom/pydicom

[5] openCV: https://opencv.org/