Gernot Christian WALZL

Stochastische Rekonstruktion der 3-dimensionalen Mikrostruktur von Lithium-Ionen-Zellen

MASTERARBEIT

zur Erlangung des akademischen Grades eines
Diplom-Ingenieurs

Masterstudium Telematik

Graz University of Technology

Technische Universität Graz

Betreuer:
Dipl.-Ing. Georg SCHARRER
Virtual Vehicle Competence Center (ViF)

Begutachter:
Univ.-Prof. Dipl.-Ing. Dr.techn. Franz AURENHAMMER
Institut für Grundlagen der Informationsverarbeitung, TU Graz

Graz, im Mai 2011

Deutsche Fassung:
Beschluss der Curricula-Kommission für Bachelor-, Master- und Diplomstudien vom 10.11.2008
Genehmigung des Senates am 1.12.2008

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

Graz, am ……. ……..
(Unterschrift)

Englische Fassung:

STATUTORY DECLARATION

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

……. ……..
date (signature)

Kurzfassung

Von den Elektroden einer Lithium-Ionen-Zelle werden mittels Elektronenmikroskopie Querschnittsbilder aufgenommen. Durch eine Bildverarbeitung werden die im Querschnittsbild vorhandenen Materialien erkannt. Hierzu wird das Bild segmentiert und anschließend werden die erhaltenen Segmente klassifiziert. Eine Klasse wird verwendet, um 2-dimensionale geometrische Formen mit der Methode der kleinsten Fehlerquadrate zu erkennen.

Die 2-dimensionalen geometrischen Formen werden statistisch analysiert. Es wird angenommen, dass es sich um Schnittbilder von 3-dimensionalen geometrischen Formen handelt. Bei der gegebenen Problemstellung handelt es sich um Ellipsoide die mit einer Ebene geschnitten werden und als Ellipsen im Querschnittsbild zu sehen sind. Durch eine statistische Auswertung der 2-dimensionalen geometrischen Formen, werden die wahrscheinlichsten Parameter der angenommenen 3-dimensionalen geometrischen Formen bestimmt.

Mit den bestimmten 3-dimensionalen geometrischen Formen wird randomisiert eine 3-dimensionale Geometrie rekonstruiert. Querschnitte der rekonstruierten Geometrie sind statistisch mit dem ursprünglichen Querschnittsbild vergleichbar. Das ursprüngliche Querschnittsbild ist in der Rekonstruktion nicht enthalten.

Die rekonstruierte 3-dimensionale Geometrie wird in einem äquidistant diskretisierten Volumen gespeichert. Die Darstellung des gesamten Volumens am Bildschirm ist effizient genug, um interaktiv rotiert u.dgl. zu werden.

Eingesetzt wird die rekonstruierte 3-dimensionale Geometrie, um Parameter eines mathematischen Modells der Elektrochemie einer Lithium-Ionen-Zelle indirekt zu bestimmen.

Die Implementierung ist durch eine automatisch generierte, grafische Benutzerschnittstelle verwendbar. Die Benutzerschnittstelle wird zur Laufzeit durch eine Analyse der gegebenen Funktionen generiert.

Stichwörter: Segmentierung, Ellipse Fitting, Mixture Model, Geometrierekonstruktion, Parameterabschätzung

Abstract

Cross-sectional images are captured from the electrodes of a lithium-ion cell by using electron microscopy. Computer vision is used to identify existing materials in the cross-sectional image. To achieve this, the image is segmented and then the resulting segments are classified. One class is used to detect 2-dimensional geometric shapes using least squares fitting.

The 2-dimensional geometric shapes are analysed statistically. It is assumed that the images are cross-sections of 3-dimensional geometric shapes. In the given problem, there are ellipsoids cut by a plane and are shown as ellipses in the cross-sectional image. A statistical analysis of the 2-dimensional geometric shapes is used to determine the most likely parameters of the assumed 3-dimensional geometric shapes.

Using the determined 3-dimensional geometric shapes, a 3-dimensional geometry is reconstructed. Cross-sections of the reconstructed geometry are statistically comparable to the original cross-sectional image. The original cross-sectional image is not included in the reconstruction.

The reconstructed 3-dimensional geometry is stored in an equidistantly sampled volume. The rendering of the total volume on screen is efficient enough to be rotated interactively.

The reconstructed 3-dimensional geometry is used to indirectly determine parameters of a mathematical model of the electrochemistry of a lithium-ion cell.

The implementation is usable by an automatically generated graphical user interface. The user interface is generated at run-time by an analysis of given functions.

Keywords: Segmentation, Ellipse Fitting, Mixture Model, Reconstruction of Geometry, Parameter Estimation

Vorwort

Das Studium der Telematik an der Technischen Universität Graz[1] führte zu dieser Diplomarbeit. Vergleichbar mit dem Studium selbst, vereint diese Arbeit mehrere Fachbereiche. Sie stellt eine interdisziplinäre Herausforderung an Bereiche der Bildverarbeitung, der Stochastik, der Softwareentwicklung und weiteren Themengebieten dar.

Das Themengebiet der Elektrochemie ist nicht Teil der Universitätsausbildung der Telematik. Es musste im Selbststudium erarbeitet werden.

Ein großer Dank geht an das Kompetenzzentrum - Das virtuelle Fahrzeug (ViF)[2], speziell an Dipl.-Ing. Dr.techn. Martin Cifrain und Dipl.-Ing. Georg Scharrer, für die Existenz dieser Arbeit. Dipl.-Ing. Dr.techn. Martin Cifrain resümiert die elektrochemischen Vorgänge in einer Zelle hervorragend. Die mathematische Betreuung von Dipl.-Ing. Georg Scharrer war sehr inspirierend für die Verbesserung dieser Arbeit.

Ein besonderer Dank geht an Univ.-Prof. Dipl.-Ing. Dr.techn. Franz Aurenhammer für die Unterstützung und das außerordentliche Engagement.

Die Herstellung der Querschnitte der Elektroden und die Aufnahmen der Querschnitte mit dem Elektronenmikroskop sind dem Institut für Elektronenmikroskopie und Feinstrukturforschung (FELMI)[3] der TU Graz zu verdanken.

Dieses Projekt wird aus Mitteln des Klima- und Energiefonds[4] gefördert und im Rahmen des Programms „NEUE ENERGIEN 2020“ durchgeführt.

Graz, im Mai 2011 Gernot Walzl

[1] http://www.tugraz.at/
[2] http://www.vif.tugraz.at/
[3] http://www.felmi-zfe.tugraz.at/
[4] http://www.klimafonds.gv.at/

Inhaltsverzeichnis

1 Motivation

1.1 Grundlegendes zur Lithium-Ionen-Zelle

Eine Lithium-Ionen-Zelle besteht im wesentlichen aus zwei Elektroden, einem Separator und dem flüssigen Elektrolyt. Der Elektrolyt ist ein Ionenleiter, der die Hohlräume der porösen Elektroden und den Separator durchsetzt.

An der Elektrodenoberfläche laufen ständig elektrochemische Reaktionen ab. Es werden positiv geladene Ionen aus der Elektrode gelöst und Elektronen bleiben in der Elektrode zurück. Dadurch ergibt sich an jeder Elektrode ein bestimmtes elektrisches Potential. Die Zellspannung resultiert aus der Differenz der beiden Elektrodenpotentiale. Im unbelasteten, geladenen Zustand spricht man von der Nennspannung der Zelle. Der Separator dient als Abstandhalter, um einen elektrischen Kurzschluss zu vermeiden.

In der Elektrochemie ist die Anode jene Elektrode, an der eine Oxidationsreaktion stattfindet. Bei einer Oxidationsreaktion werden Elektronen frei, die über den elektrischen Anschluss abgeführt werden. Dies ist in Abb. 1.1 dargestellt.

Tiefere Einblicke in die elektrochemischen Zusammenhänge von Batterien sind in [Hamann05] und [Linden01] ausführlich erklärt.

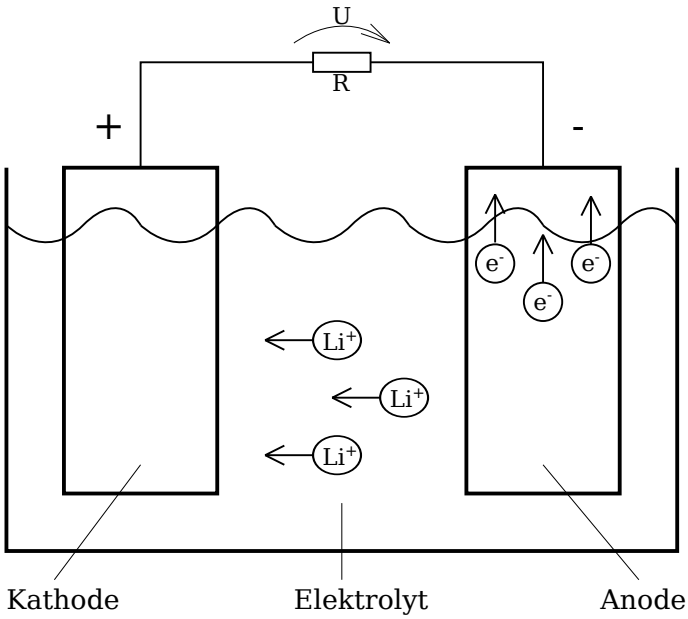


Abbildung 1.1: Schematische Darstellung des Entladevorganges einer Lithium-Ionen-Zelle

Mechanisch handelt es sich bei den Elektroden um beschichtete Kupfer/Aluminium-Folien mit einer Dicke von ca. 10 – 100 µm. Der Separator ist eine Folie mit ca. 10 – 20 µm Dicke. Diese Folien werden dicht aufgewickelt oder gestapelt. Bei Flachzellen werden die flachen Wicklungen oder der Stapel in prismatischen Gehäusen untergebracht (beispielsweise Handyakku). Rundzellen haben die klassisch zylindrische Form einer Batterie.

Als Anodenmaterial wird überwiegend Graphit C_6 eingesetzt. Graphit hat die Eigenschaft, Lithium einzulagern. Dies geschieht beim Laden der Zelle.

Für die Kathode hat sich noch kein bestimmtes Material durchgesetzt. Häufiger anzutreffen ist beispielsweise Lithium-Cobalt-Dioxid $LiCoO_2$, kurz LCO. Zwischen Graphit und LCO ergibt sich für eine Zelle eine Nennspannung von 3,6 V. Weiters wird Lithium-Eisen-Phosphat $LiFePO_4$, kurz LFP, häufig eingesetzt. Zwischen Graphit und LFP ergibt sich eine Nennspannung von 3,3 V. Lithium-Nickel-Cobalt-Aluminium-Oxid $Li(Ni,Co,Al)O_2$, kurz NCA, findet Verwendung bei Zellen, die für hohe Energiedichten ausgelegt sind.

Die Mikrostruktur der eingesetzten Materialien bestimmen die Eigenschaften der Lithium-Ionen-Zelle. Um diese zu untersuchen, wurde am Institut für Elektronenmikroskopie und Feinstrukturforschung (FELMI)[1] mit einem fokusierten Ionenstrahl (engl.: Focused Ion Beam) durch Beschuss mit Gallium-Ionen eine Art „Graben“ an der Oberfläche der Folien produziert. Die Schnittflächen wurden mit einem Elektronenmikroskop aufgenommen. Die Aufnahmen der unterschiedlichen Materialien sind in Abb. 1.2, Abb. 1.3 und Abb. 1.4 zu sehen.

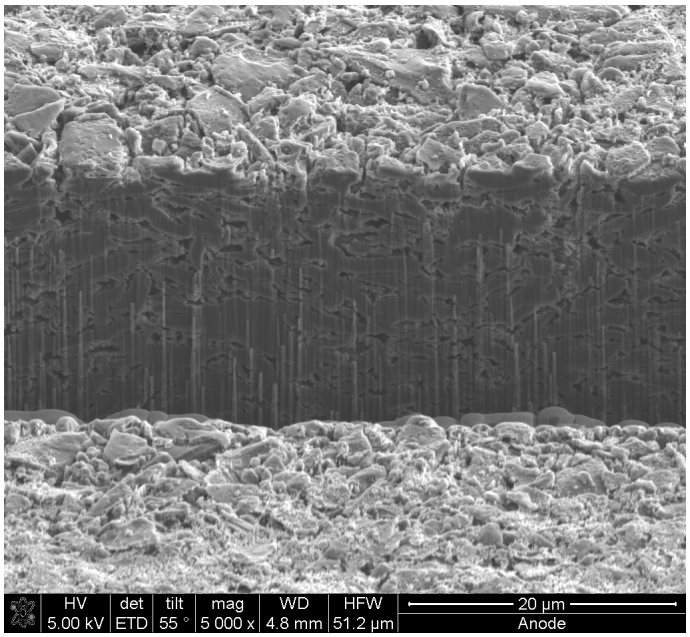


Abbildung 1.2: Grabenansicht einer Graphit-Anode

[1] `http://www.felmi-zfe.tugraz.at/`

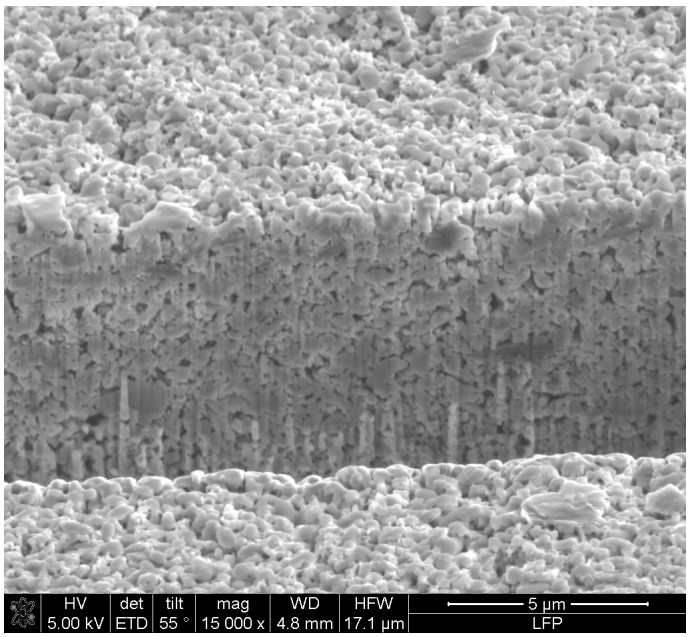


Abbildung 1.3: Grabenansicht einer LFP-Kathode

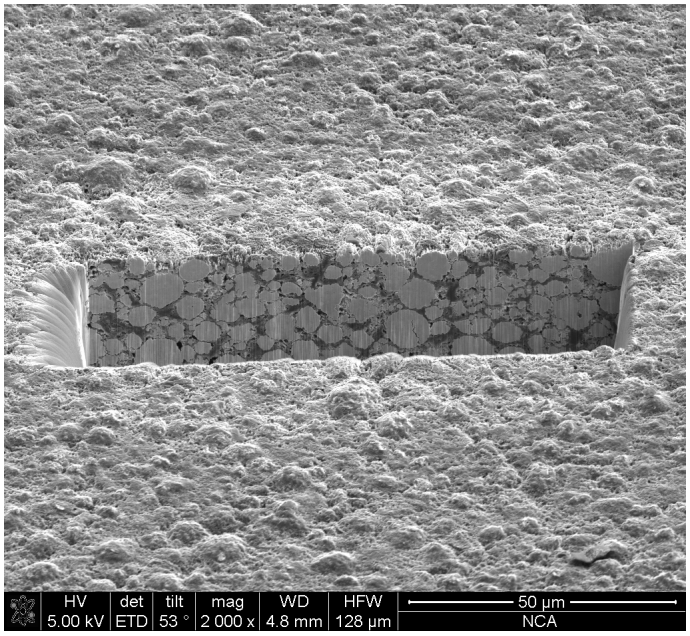


Abbildung 1.4: Grabenansicht einer NCA-Kathode

Im Querschnittsbild der NCA-Kathode (siehe Abb. 1.5) ist die Zusammensetzung des Kathodenmaterials besonders gut zu erkennen. Deshalb wird es für weiterführende Erklärungen herangezogen.

Die poröse Elektrode besteht aus elektrochemisch aktivem Material, inaktivem Material und Freiraum. Der Freiraum füllt sich bei einer assemblierten Zelle mit Elektrolyt. Das inaktive Material „verklebt“ einerseits die aktiven Teilchen miteinander, andererseits ist es als Elektronen-Leiter im Einsatz. Das aktive Material ist näherungsweise ellipsenförmig im Querschnittsbild (Abb. 1.5) zu sehen. Beim Entladevorgang bewegen sich die Lithium-Ionen aus dem Elektrolyt in das aktive Material. Die Elektronen vom elektrischen Anschluss bewegen sich über das inaktive Material zum aktiven Material.

Ausgehend von den Diffusionsvorgängen der Lithium-Ionen im Elektrolyt in porösen Materialien (Elektroden u. Separator), der Größe der Fläche zwischen Elektrolyt und aktivem Material (Austauschfläche), dem Stromfluss in den Elektroden und weiteren elektrochemischen Vorgängen wurde von Doyle et al. ein mathematisches Modell [Doyle93] erstellt. Dieses grundlegende Modell beschreibt die elektrochemischen Vorgänge beim Laden und Entladen einer Lithium-Ionen-Zelle. Es handelt sich hierbei um 7 gekoppelte, nicht-lineare, partielle Differentialgleichungen, parabolischen und elliptischen Typs. Die Gleichungen enthalten viele, mit dem derzeitigen Stand der Technik nicht direkt (messbar) zu bestimmende Parameter.

Durch Änderung der Parameter ändert sich naturgemäß der Verlauf der Lade- und Entladekurve der simulierten Zelle. Durch Messergebnisse an realen Zellen ergibt sich ein Soll-Kurvenverlauf. Anhand dessen können die Parameter optimiert werden, sodass die simulierte Kurve der gemessenen Kurve möglichst ähnelt. Es ergibt sich ein Parametersatz, der eine bestimmte Zelle in einem bestimmten Arbeitsbereich beschreibt.

Spätere Untersuchungen haben gezeigt, dass nicht alle elektrochemischen Effekte im Modell von Doyle et al. [Doyle93] abgebildet werden. Dies ist gut zu erkennen, wenn die Zelle mit unterschiedlichen, teils hohen Strömen geladen und entladen wird. Durch Parameteroptimierung der verschiedenen Kurvenverläufe ergeben sich unterschiedliche Parametersätze für die selbe Zelle.

Ausgehend vom grundlegenden Modell von Doyle et al. [Doyle93] gibt es inzwischen mehrere Arbeiten, die das elektrochemische Modell erweitern, um die Vorgänge detaillierter und dadurch wahrheitsgetreuer mathematisch abzubilden. Eine Zusammenfassung der vorherrschenden Modelle (inkl. Gleichungssätze) ist in [Santhanagopalan06] zu finden.
Eine Parameterabschätzung ist in [Santhanagopalan07] beschrieben.

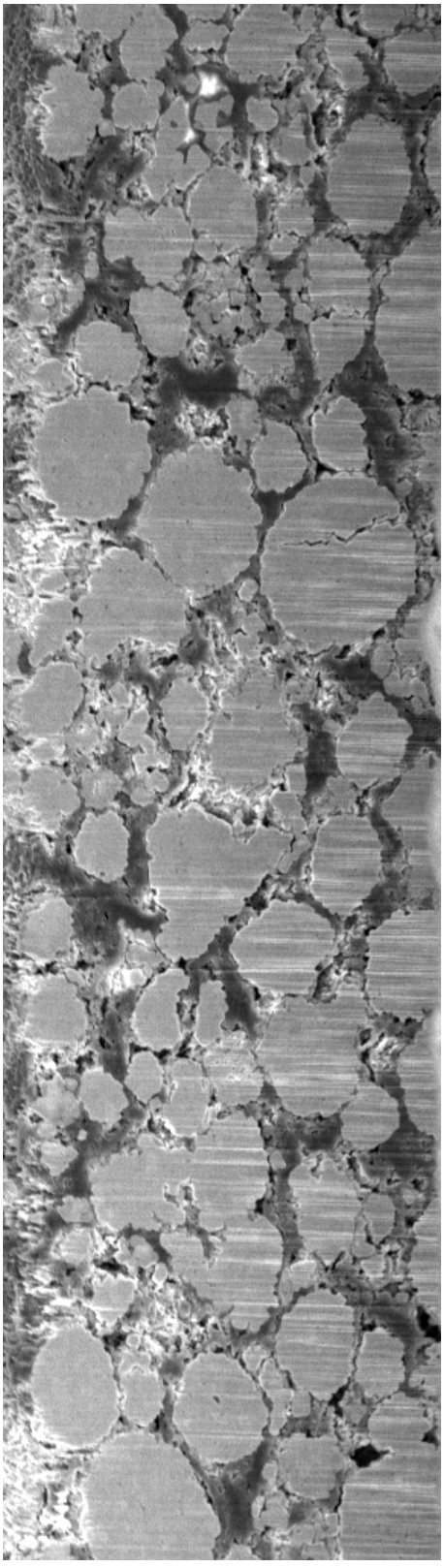


Abbildung 1.5: Schnittbild der NCA-Kathode

Ein Beispiel der Auswirkung unterschiedlicher Parameter ist der Vergleich zwischen Zellen mit hoher Energiedichte (High Energy Density) und Zellen mit hoher Leistungsdichte (High Power Density). Das Lithium wird vom aktiven Material in der Elektrode aufgenommen. Je mehr Lithium aufgenommen werden kann, desto mehr Energie kann in der Zelle gespeichert werden. Dies wird durch Vergrößerung des Volumens der Partikel des aktiven Materials erreicht. Jedoch verringert sich bei größerem Partikelvolumen die Summe der Oberflächen aller Partikel und somit die zur Verfügung stehende Austauschfläche zwischen aktivem Material und Elektrolyt. Es kann somit das Lithium langsamer aufgenommen bzw. abgegeben werden. Dadurch fließt weniger Strom, was die elektrische Leistung der Zelle verringert. Dieser Vergleich ist bei den Partikelgrößen der LFP-Kathode (Abb. 1.3) und der NCA-Kathode (Abb. 1.4) zu sehen. Die NCA-Kathode ist für hohe Energiedichte ausgelegt und hat größere Partikel. Die geometrische Struktur der LFP-Kathode erlaubt höhere Ströme und hat somit eine höhere Leistungsdichte.

Wie an diesem Beispiel zu erkennen ist, sind grundlegende Eigenschaften einer Zelle durch die Geometrie gegeben. Deshalb wird versucht, aus der Untersuchung der Mikrostruktur der Zelle, diverse Parameter für die elektrochemischen Modelle zu bestimmen.

In [Chen07] werden verschiedene Additive für die Wärmeabführung in der Kathode bewertet. Hierzu werden verschiedene geometrische Formen für das inaktive Material angenommen. Es handelt sich um einfache Formen wie flache Ellipsoide, Kugeln, etc. Die geometrischen Formen der Mikrostruktur sind notwendig, um die Wärmeausbreitung berechnen zu können.

In [Shearing10] wird das Anodenmaterial mit Computertomographie (nano-CT) untersucht. Es werden mehrere Röntgenbilder aus verschiedenen Winkeln aufgenommen. Die mathematischen Grundlagen von Johann Radon (Radon-Transformation) erlauben es, die 3-dimensionale Struktur zu errechnen.

In dieser Arbeit wird die 3-dimensionale Struktur anhand statistischer Zusammenhänge aus einem Querschnittsbild errechnet. Beginnend mit Bildverarbeitung werden Informationen aus dem Querschnittsbild extrahiert. Über die Wahrscheinlichkeitstheorie werden statistische Zusammenhänge analysiert und Häufigkeiten, der im gegebenen Bild vorkommenden Geometrien, berechnet. Mögliche 3-dimensionale Strukturen können aus den Ergebnissen der statistischen Analyse rekonstruiert werden. Die rekonstruierten Strukturen werden interaktiv am Bildschirm dargestellt. Die softwaretechnische Umsetzung ist mit Python realisiert.

Das Einsatzgebiet der rekonstruierten Mikrostruktur ist die indirekte Bestimmung von Parametern wie Austauschfläche, Porosität, etc. Ideen hierzu sind in Kap. 6 beschrieben. Ein weiteres mögliches Einsatzgebiet ist die Berechnung der Wärmeleitung einer Zelle.

1.2 Problemstellung

Gegeben ist ein 2-dimensionales Querschnittsbild wie in Abb. 1.5. Die einzelnen Pixel des Bildes müssen im 1. Schritt dem jeweiligen Material zugeordnet bzw. in Klassen eingeteilt werden. Hierbei handelt es sich um eine der folgenden Klasse:

- Aktives Material (rundliche Partikel)
- Inaktives Material (Leitruß, Binder, Additive)
- Freiraum bzw. Elektrolyt

Angenommen wird, dass die statistischen Gegebenheiten des Querschnittsbildes von der Position unabhängig sind. Des weiteren wird angenommen, dass eine Rotation um die x-Achse statistisch äquivalente Querschnittsbilder liefert. Das Achsensystem ist in Abb. 1.6 definiert.

Gesucht wird eine 3-dimensionale Rekonstruktion der Geometrie anhand statistischer Auswertung des gegebenen Querschnittsbildes. Ein Querschnittsbild aus der generierten Geometrie muss statistisch äquivalent zum gegebenen Querschnittsbild sein, d.h. Partikelgrößen, Volumenverhältnisse usw. sollen vergleichbar sein. Das gegebene Querschnittsbild muss in der rekonstrierten Geometrie selbst nicht vorkommen.

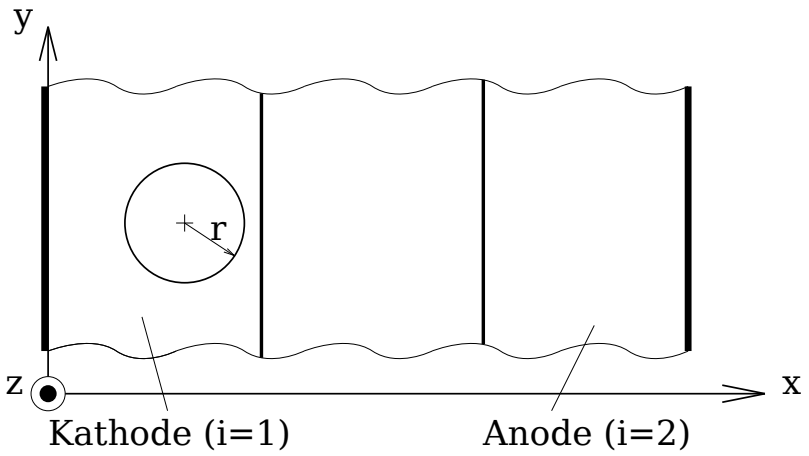


Abbildung 1.6: In der Elektrochemie gängiges Achsensystem

1.3 Lösungsansatz

Um das eigentliche Problem (von 2D auf 3D) zu lösen, wird in einem ersten Ansatz das Problem um eine Dimension reduziert betrachtet. Im Folgenden wird gezeigt, wie von einem geraden, 1-dimensionalen Schnitt eines 2-dimensionalen Bildes ein statistisch äquivalentes, 2-dimensionales Bild rekonstruiert werden kann. Es wird angenommen, dass der Schnitt durch das Bild richtungsunabhängig ist. Egal in welcher Richtung bzw. Winkel das Bild geschnitten wird, verändert sich, im statistischen Sinne, der 1-dimensionale Schnitt nicht.

Für die Herleitung wird weiters angenommen, dass es sich um Kreise handelt, die geschnitten werden. Später wird gezeigt, dass es nicht notwendig ist, die geometrische Form anzunehmen. Diese kann unter gewissen Voraussetzungen (eindeutige Trennbarkeit der Dichtefunktionen) errechnet werden.

Es können folgende Informationen aus dem 1-dimensionalen Schnitt entnommen werden:

- Mengenverhältnisse, wie Porosität $\varepsilon = \frac{Freiraum}{Gesamt}$
- Verteilung der Schnittlängen

Die Mengenverhältnisse errechnen sich im 1-dimensionalen Fall aus den Linienverhältnissen. Im 2- bzw. 3-dimensionalen Fall handelt es sich um Verhältnisse der Flächen bzw. der Volumina.

Zusammen mit den oben getroffenen Annahmen sind diese Informationen hinreichend, um eine statistische Rekonstruktion des Ursprungsbildes zu erreichen.

Lemma 1.1. *Die Mengenverhältnisse eines statistisch repräsentativen 1-dimensionalen Schnittes entsprechen den Mengenverhältnissen des 2-dimensionalen Ursprungsbildes.*

Beweis. Man betrachte ein diskretisiertes 2-dimensionales Bild. Das Bild besteht aus Zeilen. Jede dieser Zeilen wird aneinandergereiht. Es ergibt sich eine lange, eindimensionale Zeile. Bestimmt man die Mengenverhältnisse dieser zusammengesetzten Zeile, sind diese äquivalent zu den Mengenverhältnissen im 2-dimensionalen Bild. Bei einem statistisch repräsentativen Teil der zusammengesetzten Zeile bleiben die Mengenverhältnisse gleich. Somit entsprechen die Mengenverhältnisse eines statistisch repräsentativen Schnittes den Mengenverhältnissen des Ursprungsbildes. □

Lemma 1.1 kann für höhere Dimensionen ebenfalls bewiesen werden. Der Beweis verläuft analog zu dem hier gezeigten.

Die Verteilung der Schnittlängen wird dazu genutzt, die Zusammensetzung der im ursprünglichen 2-dimensionalem Bild vorhandenen Geometrien zu bestimmen. Für die Herleitung wird angenommen, dass es sich um Kreise mit unterschiedlichen Radien handelt.

Die analytische Darstellung eines Einheitskreises ($r = 1$) ist in Gl. 1.1 gegeben. Der zugehörige Funktionsgraph ist in Abb. 1.7 zu sehen.

$$y = \pm\sqrt{1 - x^2} \tag{1.1}$$

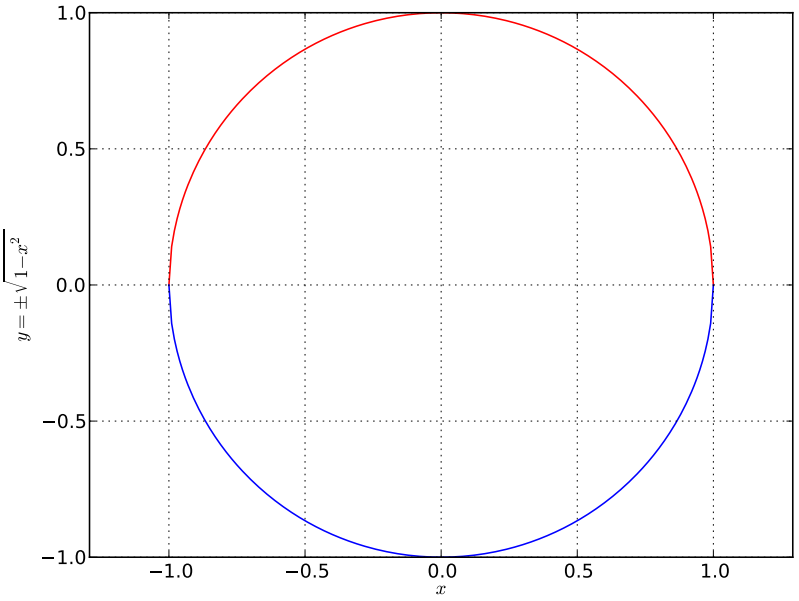


Abbildung 1.7: Einheitskreis

Die Wahrscheinlichkeit, dass der Kreis an einer Stelle x geschnitten wird, ist bei jedem x gleich groß. Die Länge des Schnittes kann beim Einheitskreis zwischen 0 und 2 liegen. Dies ist in Abb. 1.8 illustriert.

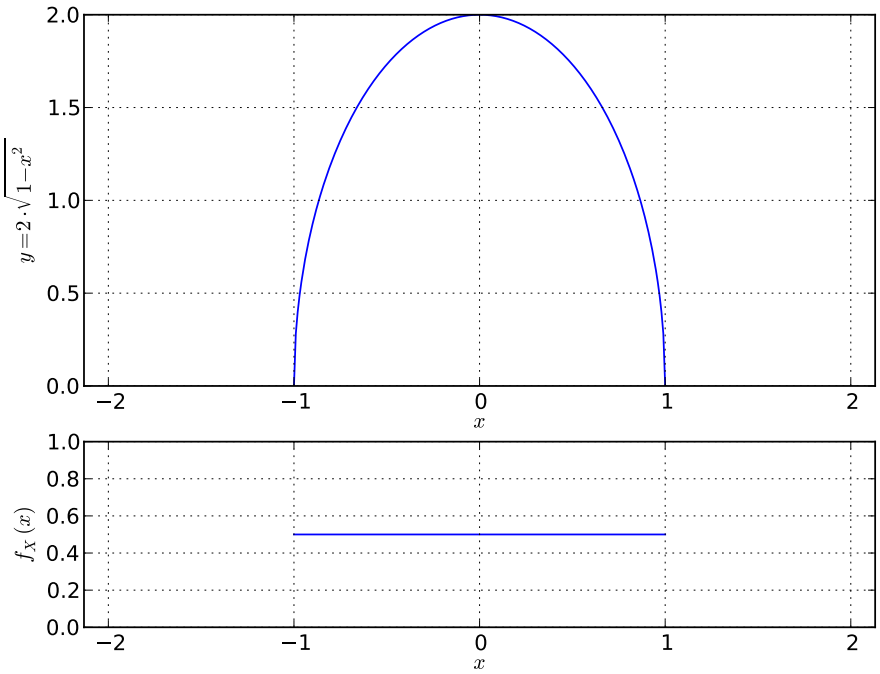


Abbildung 1.8: Mögliche Schittlängen eines Einheitskreises

Ausgehend von der Tatsache, dass jede Position eines Schnittes gleich wahrscheinlich ist, wird eine Schnittlängen-Verteilung berechnet. Diskret (vereinfacht) gesehen ist dies nichts anderes, als würde man den Kreis an konstanten Abschnitten schneiden und die erhaltenen Schnittlängen in Intervalle einteilen und abzählen. Dadurch würde man die Kreisdichte in diskreten Intervallen erhalten.

Um diesen Zusammenhang analytisch zu beschreiben, wird die gleichverteilte Zufallsvariable X über die Kreisgleichung zu einer, von der Form des Kreises abhängigen Zufallsvariable Y transformiert. Dies wird erreicht, indem die Gleichverteilungs-Dichte $f_X(x)$ über $y = 2 \cdot \sqrt{1 - x^2}$ zur Kreisverteilungsfunktion $F_Y(y)$ und weiters zur Kreisdichte $f_Y(y)$ berechnet wird.

Die Kreisverteilungsfunktion $F_Y(y)$ beschreibt die Wahrscheinlichkeit, dass der Kreis an einer Stelle mit einer Schnittlänge y oder kleiner geschnitten wird. Um die Berechnung nachvollziehen zu können, ist Abb. 1.8 hilfreich. Man stelle sich einen vorgegebenen y-Wert, also eine vorgegeben Schnittlänge vor. Dieser y-Wert hat bei der obigen Kurve 2 Schnittpunkte $(x_1, y), (x_2, y)$. Diese Schnittpunkte verfolgt man über die x-Achse zur Gleichverteilungs-Dichte $f_X(x)$. Der Wert für $F_Y(y)$ ergibt sich durch den Flächeninhalt von $f_X(x)$ zwischen -1 bis x_1 und x_2 bis 1. Dieser Flächeninhalt entspricht der Wahrscheinlichkeit, dass ein zufällig (gleichverteilt) gewählter Schnitt einer Stelle x eine Schnittlänge von y oder kleiner zur Folge hat.

Aus dem beschriebenen Zusammenhang $P_Y(Y \leq y) = \int_{-1}^{x_1} f_X(x)dx + \int_{x_2}^{1} f_X(x)dx$ ergibt sich die Kreisverteilungsfunktion in Gl. 1.2*c*.

$$\begin{aligned} F_Y(y) &= P_Y(Y \leq y) = 1 - P_Y(Y > y) && (1.2a) \\ &= \int_{-1}^{x_1} f_X(x)dx + \int_{x_2}^{1} f_X(x)dx = 1 - \int_{x_1}^{x_2} f_X(x)dx && (1.2b) \\ &= 1 - \sqrt{1 - \left(\frac{y}{2}\right)^2} && (1.2c) \end{aligned}$$

Die zugehörige Kreisdichte $f_Y(y)$ errechnet sich durch die Ableitung der Verteilungsfunktion nach y. Das Ergebnis ist in Gl. 1.3 zu sehen.

$$f_Y(y) = \frac{d}{dy} F_Y(y) = \frac{y}{4 \cdot \sqrt{1 - \left(\frac{y}{2}\right)^2}} \tag{1.3}$$

Die Funktionsgraphen der Verteilungsfunktion und der Dichte des Kreises sind in Abb. 1.9 dargestellt.

Gl. 1.4 beschreibt, dass Gl. 1.3 eine gültige Dichtefunktion darstellt.

$$\int_{-\infty}^{\infty} f_Y(y)dy = 1 \tag{1.4}$$

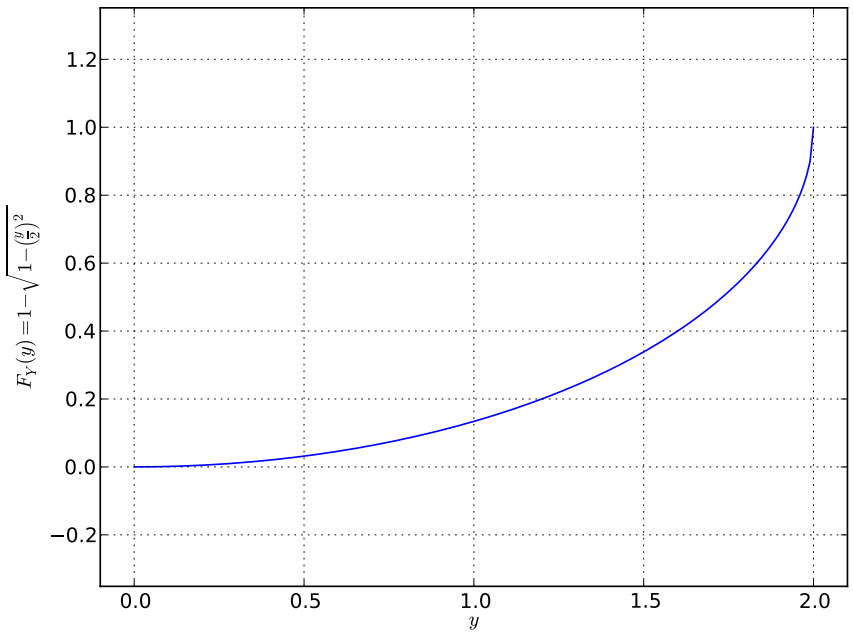
Der Erwartungswert für Y errechnet sich laut Gl. 1.5 für einen Einheitskreis zu $\frac{\pi}{2}$. Mit diskreten Überlegungen lässt sich die exakte analytische Beschreibung des Erwartungswerts verstehen. Man schneide den Einheitskreis mit äquidistanten Linien parallel zur y-Achse. Eine Schätzung des Erwartungswerts ergibt sich aus dem arithmetischen Mittelwert der Schnittlängen.

$$E(Y) = \frac{\int_{-1}^{1} 2 \cdot \sqrt{1 - x^2}dx}{2} = \frac{\pi}{2} = 1,5708 \tag{1.5}$$

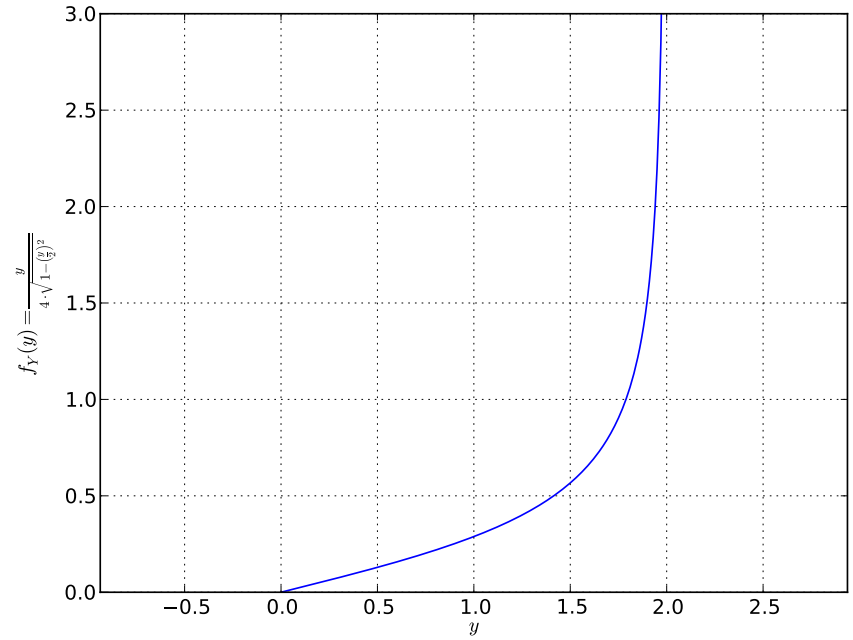
Die zuvor beschriebenen Gleichungen beziehen sich auf den Einheitskreis ($r = 1$). Für beliebige Radien verallgemeinert sich die Kreisverteilungsfunktion zu Gl. 1.6 und die Kreisdichte zu Gl. 1.7.

$$F_Y(y) = 1 - \sqrt{1 - \left(\frac{y}{2r}\right)^2} \tag{1.6}$$

$$f_Y(y) = \frac{y}{(2r)^2 \cdot \sqrt{1 - \left(\frac{y}{2r}\right)^2}} \tag{1.7}$$



(a) Kreisverteilungsfunktion



(b) Kreisdichte

Abbildung 1.9: Verteilungsfunktion und Dichte des Einheitskreises

Da die Verteilung der Schnittlängen nur diskret vorhanden ist, werden die Kreisdichtefunktionen ebenfalls diskretisiert generiert. Ein Beispiel für eine diskretisierte Kreisdichtefunktion für einen Kreis mit Radius $r = 8$ ist in Abb. 1.10 zu sehen.

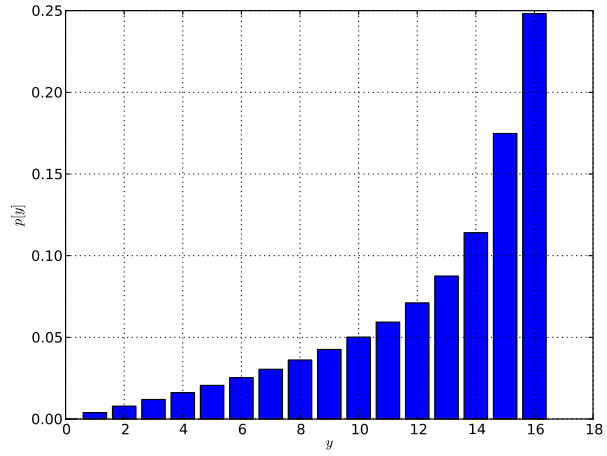


Abbildung 1.10: Diskrete Kreisdichte ($r = 8$)

Ein Beispiel für eine Verteilung der Schnittlängen ist in Abb. 1.11 dargestellt. Sie kann als Summe von gewichteten Kreisdichtefunktionen gesehen werden. Aus der Gewichtung der Kreisdichtefunktionen ergeben sich die relativen Häufigkeiten h_i der auftretenden Radien. Abb. 1.11 zeigt eine Verteilung, wo Kreise mit $r = 4$ doppelt so häufig vorkommen, wie Kreise mit $r = 8$.

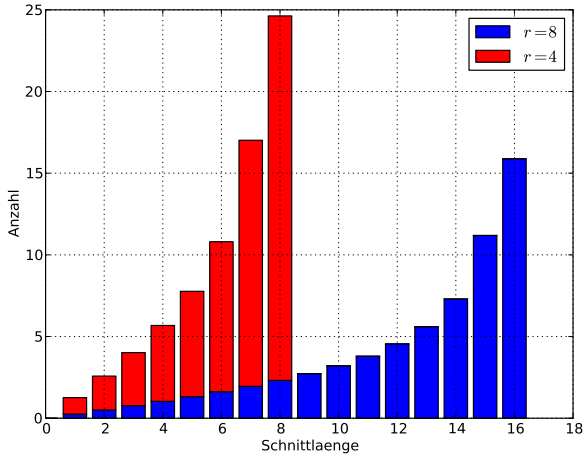


Abbildung 1.11: Optimal trennbare Verteilung der Schnittlängen als Summe von 2 Kreisdichten

Die relativen Häufigkeiten h_i beschreiben, wie häufig ein Kreis mit Radius r_i auftritt. Die Porosität $\varepsilon = \frac{Hohlraum}{Gesamt}$ drückt Mengenverhältnisse aus. Das komplementäre Mengenverhältnis der Porosität $\varepsilon' = (1 - \varepsilon)$ ergibt sich im 2-dimensionalen Fall aus der Summe aller nicht-leeren Flächen $A_{Befuellt}$ bezogen auf die Gesamtfläche A_{Gesamt}. Gl. 1.8c und Gl. 1.9 beschreiben, wie die relativen Häufigkeiten h_i in absolute Häufigkeiten H_i über Flächenverhältnisse berechnet werden.

$$(1 - \varepsilon) = \frac{A_{Befuellt}}{A_{Gesamt}} \tag{1.8a}$$

$$\varepsilon' = \frac{\sum_{i=1}^{n} k \cdot h_i \cdot (r_i^2 \cdot \pi)}{A_{Gesamt}} \tag{1.8b}$$

$$k = \frac{\varepsilon' \cdot A_{Gesamt}}{\sum_{i=1}^{n} h_i \cdot (r_i^2 \cdot \pi)} \tag{1.8c}$$

$$H_i = k \cdot h_i \tag{1.9}$$

Die absolute Häufigkeit H_i ist die konkrete Anzahl der Kreise mit Radius r_i. Durch das Wissen über die gewünschte Größe des zu generierenden Bildes, die Anzahlen und Größen der Kreise kann eine Geometrie rekonstruiert werden. Bei der Rekonstruktion ist die Nicht-Überlagerung der Kreise zu beachten. Wenn sich die Kreise überlagern, weicht die oben getroffene Berechnung von $A_{Befuellt}$ vom tatsächlichen Wert ab.

Um diese Idee zu validieren, wurde ein Testprogramm implementiert. Im ersten Schritt wird eine Geometrie mit nicht überlagernden Kreisen generiert. Dieses gegebene Bild wird an einer bestimmten Stelle geschnitten. Aus den Informationen des Schnittes wird mit der zuvor beschriebenen Methode eine diskrete Verteilung der Kreisradien berechnet. Mit dieser Verteilung wird erneut eine Geometrie mit nicht-überlagernden Kreisen generiert. Abschließend wird das generierte Bild mit dem gegebenen Bild verglichen.

Folgendes Beispiel zeigt Ergebnisse der hier vorgestellten Berechnung. Gegeben ist eine Porosität $\varepsilon = 0,5$. Es sollen Kreise von $r_1 = 8$ und $r_2 = 16$ im Bild enthalten sein. Die Kreise mit r_1 sollen doppelt so häufig vorkommen, wie Kreise mit r_2. Das aus diesen Angaben generierte Bild ist in Abb. 1.12a dargestellt.

Von diesem Bild werden Schnittlinien betrachtet und nach obiger Berechnung analysiert. Die Ergebnisse unterschiedlicher Schnittlinien sind in Abb. 1.12b und Abb. 1.12c zu sehen. Relative Häufigkeiten unter 10 % wurden ignoriert.

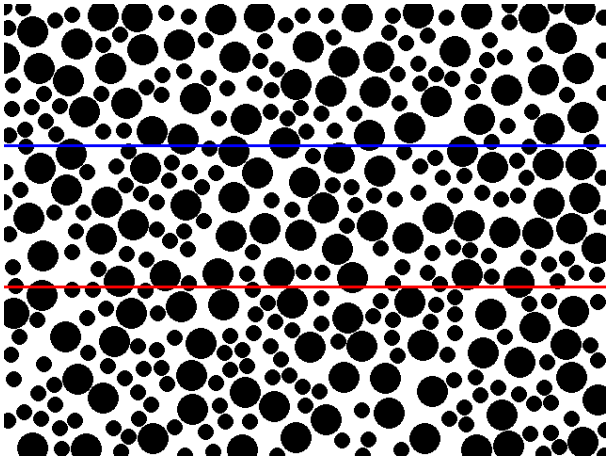
Die Zahlenwerte der Ergebnisse lauten wie folgt:

- Kreisradienverteilung aus Schnittlinie in Zeile 150 (Abb. 1.12b):
  $r_1 = 8, h_1 = 0.3128$
  $r_2 = 9, h_2 = 0.2122$
  $r_3 = 16, h_3 = 0.1742$
  $r_4 = 17, h_4 = 0.1113$

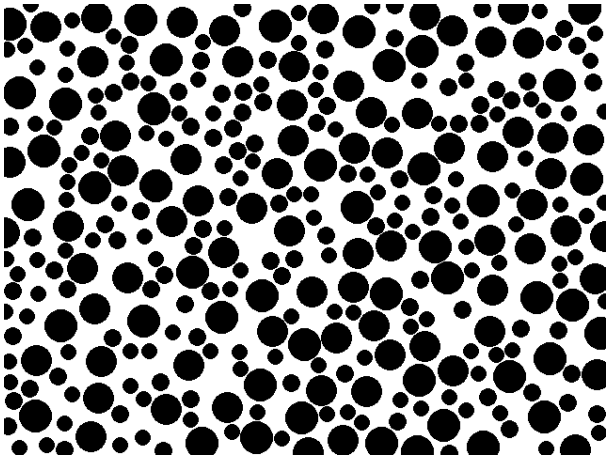
- Kreisradienverteilung aus Schnittlinie in Zeile 300 (Abb. 1.12c):
  $r_1 = 8, h_1 = 0.6037$
  $r_2 = 9, h_2 = 0.1300$
  $r_3 = 16, h_3 = 0.1423$

Auch bei kleinen Schnittlängen (Stichproben) können brauchbare Ergebnisse erzielt werden. Je länger die Schnittlänge, desto wahrscheinlicher werden berechnete Werte die Wirklichkeit widerspiegeln. Die daraus generierte Geometrie wird dadurch die Wirklichkeit besser annähern. Die hier gewählte Länge des Schnittes orientiert sich an der statistischen Repräsentativität der Größe des Querschnittsbildes der Elektrode. Die Größe des Querschnittsbildes ist für diese Arbeit vorgegeben.

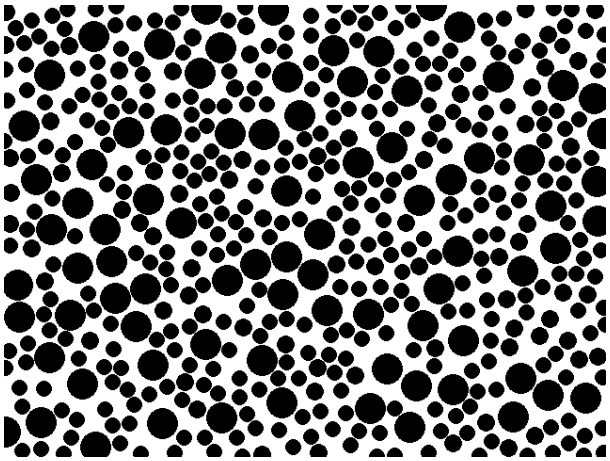
Je mehr Vorwissen in die Berechnung einfließen kann, desto besser ist dies für die Rekonstruktion. Hier wird angenommen, dass es sich um Kreise handelt, die geschnitten werden. Durch diese Annahme wird eine Verteilungsfunktion definiert. Dieses Verfahren kann in ähnlicher Weise für andere Formen (Dreiecke, Rechtecke, etc.) eingesetzt werden. Je nach geometrischer Form ergeben sich unterschiedliche Dichtefunktionen. Anders als beim Kreis sind die Schnittlängen und somit die Dichtefunktionen von der Rotation abhängig. Für jede geometrische Form kann eine Dichtefunktion aus dem Mittelwert aller möglichen Rotationswinkel erstellt werden. Treten mehrere unterschiedliche geometrische Formen auf, wird die Berechnung der Geometrien aus einem 1-dimensionalen Schnitt zu einer Suche, welche Zusammensetzung von Dichtefunktionen die Verteilung der Schnittlängen am besten trennt.



(a) Generiertes Bild



(b) Rekonstruktion aus oberen (blauen) Schnittlinie



(c) Rekonstruktion aus unteren (roten) Schnittlinie

Abbildung 1.12: Rekonstruierte Geometrie aus Kreisen

2 Bildverarbeitung

In diesem Kapitel werden alle für die Lösung der Problemstellung (Kap. 1.2) relevanten Themen der Bildverarbeitung (engl.: Computer Vision) behandelt. Zu Beginn wird das verwendete Achsensystem erklärt. In Kap. 2.1 wird über die Grundlagen der Segmentierung das eingesetzte Verfahren zur Segmentierung der Mikrostruktur des Querschnittsbildes der Elektrode erläutert. Die erhaltenen Segmente werden anschließend in Klassen eingeteilt. Da das aktive Material der Elektrode als kugelförmig bzw. ellipsoidförmig angenommen wird, handelt Kap. 2.2 von Ellipsenerkennung und morphologischen Operationen.

Abb. 2.1 zeigt das verwendete Achsensystem.

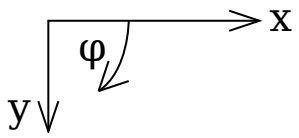


Abbildung 2.1: Verwendetes Achsensystem

Ein Bild mit einer Breite w und einer Höhe h wird in einer Matrix $A \in M(h \times w)$ gespeichert. Die Indizierung ist in Gl. 2.1 dargestellt.

$$A := \begin{pmatrix} a_{0,0} & \cdots & \cdots & \cdots & a_{0,w-1} \\ \vdots & \ddots & a_{y-1,x} & & \vdots \\ \vdots & a_{y,x-1} & a_{y,x} & a_{y,x+1} & \vdots \\ \vdots & & a_{y+1,x} & \ddots & \vdots \\ a_{h-1,0} & \cdots & \cdots & \cdots & a_{h-1,w-1} \end{pmatrix} \tag{2.1}$$

2.1 Segmentierung

Bei einer Segmentierung wird ein gegebenes Bild in Segmente geteilt. Im Idealfall umschließen die Grenzen der Segmente genau die Objekte, die von Interesse sind. Bei der gegebenen Problemstellung sind das kreisförmige Partikel des aktiven Materials und zusammenhängende Flächen des inaktiven Materials. Eine Segmentierung ist vollständig, wenn jedes Pixel des Bildes einem Segment zugeordnet ist. Sie ist überdeckungsfrei, wenn jedes Pixel maximal zu einem Segment gehört.

Ein zu untersuchendes Querschnittsbild einer Elektrode ist in Abb. 1.5 dargestellt. Die Problemstellung (Kap. 1.2) erfordert eine vollständige und überdeckungsfreie Segmentierung. Es muss jedes Pixel im Querschnittsbild in eine der folgenden Klassen eingeteilt werden:

- Aktives Material (rundliche Partikel)
- Inaktives Material (Leitruß, Binder, Additive)
- Freiraum bzw. Elektrolyt

Das aktive Material (Akt.Mat.) ist hauptsächlich kugelförmig in der Elektrode enthalten. Der Querschnitt durch eine kugelige Form zeigt einen Kreis bzw. eine Ellipse. Der Grauwert dieser Schnittfläche ist homogen. Dadurch bilden sich keine Grauwert-Gradienten direkt in der Schnittfläche aus. Die folgenden Grauwerte haben eine 8 Bit Auflösung und orientieren sich an der NCA-Kathode von Abb. 1.5. Der Grauwert des aktiven Materials liegt im oberen Bereich (ca. 180).

Das inaktive Material (Inakt.Mat.) besteht aus einem sehr hellen Teil (Grauwert über ca. 230) und einem sehr dunklen Teil (Grauwert unter ca. 60). Da wieder volles Material geschnitten wird, ist der Grauwert eines Segmentes homogen.

Zwischen den homogenen Schnittflächen sind die „Reflexionen der Oberflächen“ zu sehen. Dies ist der Freiraum, der sich bei einer assemblierten Zelle mit Elektrolyt füllt. Hier kann kein homogener Grauwert zugeordnet werden.

2.1.1 Threshold-Segmentierung

Der erste Versuch, das Bild zu segmentieren, ist über die zuvor beschriebenen Grenzen der Grauwerte. Man spricht hier von einer Threshold- oder Schwellwert-Segmentierung. Je nach Grauwert wird das Pixel zu einem Segment zugeordnet. Das Histogramm (Grauwertverteilung) des gegebenen Bildes (Abb. 1.5) ist in Abb. 2.2 dargestellt.

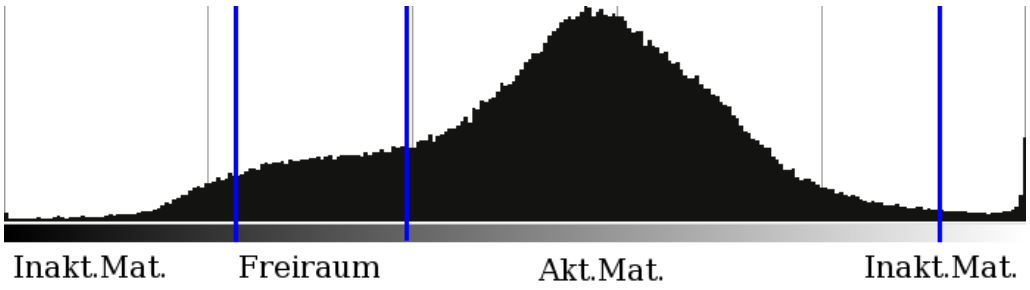


Abbildung 2.2: Histogramm der NCA-Kathode

Das Resultat der Threshold-Segmentierung ist in Abb. 2.3 zu sehen. Viele Grenzen der Segmente schmiegen sich passend zu den Kanten im Bild an. Das aktive Material ist hellgrau dargestellt. Das inaktive Material ist in dunkelgrau und weiß zu sehen. Der Freiraum ist schwarz.

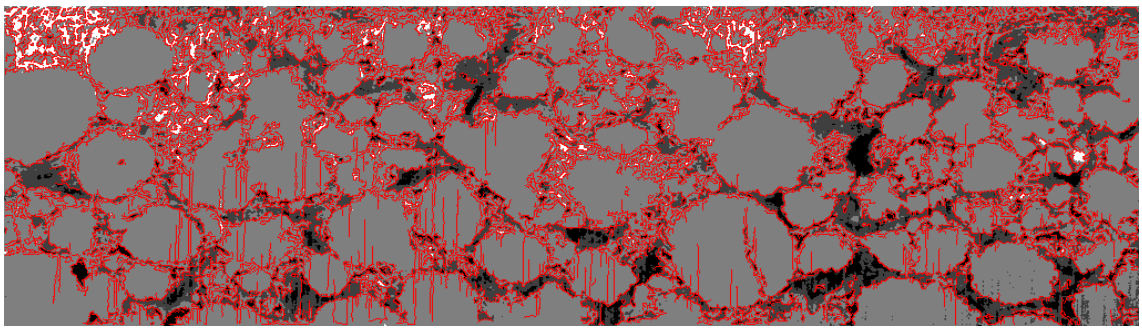


Abbildung 2.3: Threshold-Segmentierung und Canny Edge Detector

Ein großer Nachteil dieser Art der Segmentierung ist jedoch, dass die Freiräume vollständig falsch klassifiziert werden. Es vermischen sich Bereiche des inaktiven Materials sehr stark mit dem eigentlichen Freiraum. Aus diesem Grund muss ein anderes Segmentierungsverfahren verwendet werden.

2.1.2 Watershed-Segmentierung

Da die Kanten des Bildes als Evaluierungskriterium für die Segmentierung herangezogen werden, wird ein Verfahren gesucht, das Kanten zur Segmentierung verwendet. Viele, bereits implementierte, Segmentierungsalgorithmen sind in [ITKGuide05] beschrieben. Beim Watershed-Algorithmus fallen die Segmentierungsgrenzen mit den Kanten im Bild zusammen.

Die Berechnung der Watershed-Segmentierung beginnt mit dem Gradienten-Bild des gegebenen Bildes. Die Gradienten eines Bildes können mit einem einfachen Prewitt-, oder besser, einem Sobel-Filter berechnet werden. Dort, wo starke Unterschiede im Grauwert benachbarter Pixel sind, ist der Betrag des Gradienten hoch. Der Gradient ist Null, wo benachbarte Pixel einen gleichen Grauwert haben.

Das Gradienten-Bild wird vom Watershed-Algorithmus als Höhenwert-Karte interpretiert. Gedanklich kann man sich das langsame Füllen dieses, durch die Höhenwert-Karte definiertem „Gebirge“ mit Wasser vorstellen. Dort, wo das Wasser „zusammenschwappt“, wird eine Segmentierungsgrenze eingezeichnet. Dieses bildliche Modell erklärt den Namen des Verfahrens.

Parameter erlauben die Einstellung, wie groß der „Damm“ mindestens sein muss, um als eigenes Segment erkannt zu werden. Je nach Einstellung kann sich daraus eine zu hohe oder eine zu niedrige Anzahl an Segmenten ergeben. Man spricht von Über- bzw. Untersegmentierung. Da hier die erhaltenen Segmente nachträglich klassifiziert werden, ist leichte Übersegmentierung kein Nachteil für die Korrektheit des Ergebnisses.

2.1.3 Eingesetztes Segmentierungsverfahren

Sei *img* ein zu segmentierendes Querschnittsbild einer Elektrode.

`segment`(*img*)

1. *segments* = Watershed-Segmentierung von *img* (Kap. 2.1.2)
2. Für jedes Segment $seg \in segments$:
   a) *grad* = Mittelwert der Gradienten von *seg*
   b) Wenn $grad \geq$ Schwellwert:
      i. $seg \in$ **Freiraum**
   c) sonst:
      i. *grey* = Mittelwert der Grauwerte von *seg*
      ii. Durch Grauwertgrenzen von *grey* Einteilung von *seg* in **Aktives Material** $\oplus$ **Inaktives Material**

Algorithmus 2.1: Segmentierung des Querschnittsbildes

Korrektheit

Lemma 2.1. *`segment`(img) berechnet eine vollständige und überdeckungsfreie Segmentierung von img.*

Beweis. Der Watershed-Algorighmus von [ITKGuide05] ist als Filter implementiert. Der Rückgabewert ist ein Bild gleicher Abmessungen, wo jedes Pixel einem Segment zugeordnet ist. Die Segmente werden fortlaufend nummeriert. Die jeweilige Nummer des Segmentes wird als Wert des Pixels eingetragen. Da jedes Pixel genau einen Wert beinhalten kann, ist die Segmentierung überdeckungsfrei. Ein Wert für „undefiniert“ ist nicht vorgesehen. Deshalb ist die Segmentierung vollständig.

Im Schritt (2) werden alle Segmente klassifiziert. Es werden keine Segmente ignoriert oder hinzugefügt. Die Segmentierung bleibt vollständig und überdeckungsfrei. □

Laufzeit

Die Implementierung des Watershed-Algorithmus [ITKGuide05] baut eine hierarchischen Struktur der Segmente auf. Kleine Segmente werden zu größeren Segmenten in einer Baum-Datenstruktur zusammengefasst. Bildlich vorstellen kann man sich kleinere Höhenunterschiede, die zuerst vom Wasser überwunden werden. Größere Höhenunterschiede werden erst bei höherem Füllstand überwunden. Segmente mit kleineren Höhenunterschieden liegen innerhalb von Segmenten mit größeren Höhenunterschieden. Es baut sich ein hierarchisches Modell auf. Der Vorteil dieser Implementierung liegt darin, dass bei einer Änderung der Parameter die Segmentierung nicht erneut berechnet werden muss. Es werden nur jene Knoten im Baum als Segmente ausgegeben, die relevant sind.

Die Laufzeit hängt von der Komplexität des gegebenen Bildes ab. Darunter versteht sich die Anzahl der Segmente. Aus dieser Anzahl ergibt sich die Komplexität der hierarchischen Struktur und somit die Laufzeit. Eine konkrete obere Schranke der Laufzeit ist in [ITKGuide05] nicht angegeben.

Eine obere Schranke der Laufzeit für die Klassifizierung der Segmente in (2) lässt sich über die Anzahl der Pixel erläutern. Jedes Pixel ist genau einmal in einem Segment zu finden, da die Segmentierung vollständig und überdeckungsfrei ist. Hier ist zu beachten, dass die Segmentierung als Filter implementiert ist. Das Ergebnisbild des Filters muss erst auf eine Liste von Segmenten abgebildet werden. Jedes Segment wird in der Schleife genau einmal betrachtet. Dadurch wird jedes Pixel genau einmal einer Klasse zugeordnet. Es ergibt sich eine lineare Abhängigkeit der Laufzeit zu der Anzahl der Pixel im Bild.

Ergebnis

Wird der Algorithmus `segment` auf das Querschnittsbild der NCA-Kathode aus Abb. 1.5 angewandt, ergibt sich die in Abb. 2.4 dargestellte Segmentierung und Klassifizierung. Der Freiraum ist schwarz dargestellt. Inaktives Material ist dunkelgrau und weiß. Aktives Material ist grau zu sehen.

Besser als bei reiner Threshold-Segmentierung sind hier die Freiräume korrekt vom inaktiven Material getrennt.

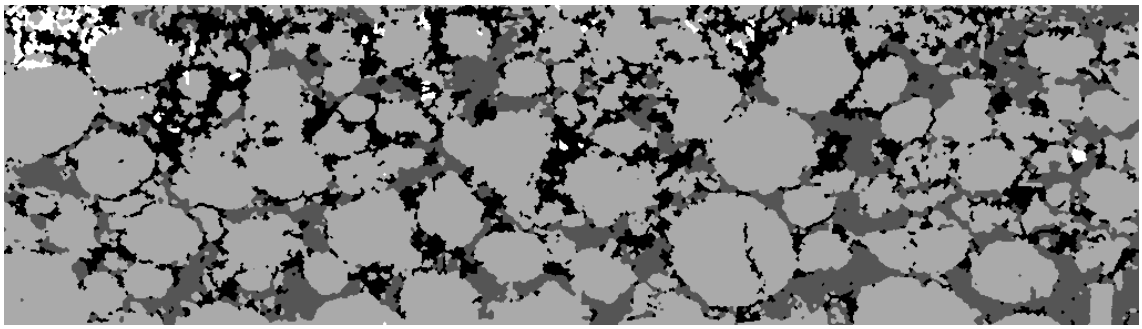


Abbildung 2.4: Segmentierte NCA-Kathode

Ohne die realen Mengenverhälntisse vorher gekannt zu haben, wurde eine Abweichung von ca. 5 % erreicht.

	Segmentierung	**Real**	**Abweichung**
Aktives Material	0,6428	0,589	5,4 %
Inaktives Material	0,1860	0,206	2,0 %
Freiraum	0,1712	0,205	3,4 %

Tabelle 2.1: Relative Mengenverhältnisse

Es ist anzumerken, dass die Mengenverhältnisse alleine nicht genügen, um das Ergebnis der Segmentierung bewerten zu können. Ein visueller Vergleich der Segmentierung mit dem gegebenen Bild bleibt nicht aus. Dabei vergleicht der Anwender, ob die Struktur der Segmentierung mit der Struktur der im Querschnittsbild vorkommenden Materialien übereinstimmt.

2.2 Ellipsenerkennung

Es wird angenommen, dass das aktive Material in der Elektrode kugel- bzw. ellipsoidförmig ist. Im Querschnittsbild ist daher das aktive Material kreis- bzw. ellipsenförmig zu sehen. Um eine analytische Beschreibung der Ellipsen zu erhalten, müssen diese als solche erkannt werden.

Beginnend mit Grundlagen der Geometrieerkennung wird in diesem Kapitel ein Verfahren entwickelt, das die zusammenhängenden Gebiete im Querschnittsbild erkennt und als Ellipsen analytisch beschreibt.

2.2.1 Hough-Transformation

Die Hough-Transformation ist ein einfaches Verfahren, um Geometrien in einem diskretisierten Bild zu erkennen. Durch die Variablen einer analytischen Beschreibung der zu erkennenden Geometrie wird der Hough-Raum aufgespannt. Jede Variable entspricht einer Koordinate. Die Werte im Hough-Raum geben an, wie wahrscheinlich eine Geometrie mit entsprechenden Variablen im Bild zu finden ist.

Anschaulicher wird dies am Beispiel einer Geraden. Die analytische Beschreibung einer Geraden ist in Gl. 2.2 dargestellt.

$$y = \tan(\alpha) \cdot x + d \tag{2.2}$$

Die Gerade wird durch einen Winkel α und einer Position d bestimmt. Daraus ergibt sich ein 2-dimensionaler Hough-Raum, der alle möglichen Werte für α und d abdeckt. Für alle Kombinationen der Werte wird Gl. 2.2 ausgewertet. Es wird überprüft, wie viele Pixel der angenommenen Gerade mit den Pixeln im Bild übereinstimmen. Dieser Wert wird an die jeweilige Stelle im Hough-Raum eingetragen. Die lokalen Maxima im Hough-Raum deuten auf die wahrscheinlichsten Geraden hin. Um Geraden in einem Bild zu erkennen, ergibt sich aufgrund der beiden Variablen eine quadratische Laufzeit. Der Vorteil dieses Verfahrens ist, dass es sehr einfach ist. Ein deutlicher Nachteil ist die hohe Laufzeit bei komplexen Geometrien.

Eine analytische Darstellung einer Ellipse in Hauptlage ist in Gl. 2.3 zu finden. Hauptlage bedeutet, dass der Mittelpunkt im Koordinatenursprung liegt und die Achsen der Ellipse parallel zu den Achsen des Koordinatensystems sind.

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1 \tag{2.3}$$

In allgemeiner Lage kann die Ellipse mit 5 Variablen beschrieben werden:
Haupthalbachse a, Nebenhalbachse b, Rotationswinkel α, Mittelpunkt (x_0, y_0).
Aus den 5 Variablen ergibt sich ein 5-dimensionaler Hough-Raum. Aufgrund der hohen Laufzeit ist dies, für die praktische Anwendung, nicht zu realisieren.

2.2.2 Effiziente Ellipsenerkennung nach Xie

Der hier beschriebene Algorithmus wurde von [Xie02] vorgestellt. Er nutzt die geometrischen Zusammenhänge der Ellipse aus, um die Laufzeit drastisch zu reduzieren.

Der Algorithmus beginnt mit einem diskretisierten Kantenbild, beispielsweise von einem Canny Edge Detector. Im Folgenden wird ein Iterationsschritt erläutert.

Aus der Menge der Kanten-Pixel wird ein Pixel p_1 gewählt. Zu p_1 wird ein weiterer Kanten-Pixel p_2 gewählt. Die beiden Pixel p_1 und p_2 definieren eine Hauptachse $2a$. Ein Voting-Array mit einer Länge der maximalen Größe von b wird mit Nullen initialisiert. Es wird versucht durch jeden weiteren Kanten-Pixel p eine Ellipse mit der Hauptachse $2a$ zu legen. Jeder Pixel p hat ein Stimmrecht für die Größe von b. Abb. 2.5 zeigt ein Beispiel dieser Situation.

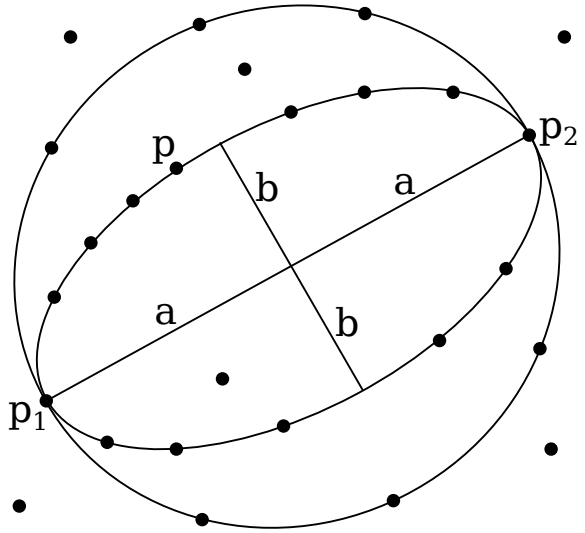


Abbildung 2.5: Beispiel der effizienten Ellipsen-Erkennungs-Methode

Auf der kleineren Nebenachse b liegen 12 Kanten-Pixel. Diese 12 Kanten-Pixel stimmen deshalb für die kleinere Nebenachse b. Weitere 6 Kanten-Pixel würden eine Ellipse mit einer größeren Nebenachse definieren. Da aber die Anzahl der Stimmen geringer ausfällt, wird diese nicht ausgewählt. Es wird die Ellipse mit den meisten Stimmen ausgewählt. Ist die Anzahl der Stimmen über einem Grenzwert, wird diese Ellipse dedektiert und alle Kanten-Pixel, die zu dieser Ellipse gehören, aus dem Kanten-Bild entfernt.

Dieser Vorgang wird für alle Kombinationen zweier Kanten-Pixel p_1 und p_2 wiederholt.

Laufzeit

Sei n die Anzahl der Kanten-Pixel des gegebenen Bildes.

Lemma 2.2. *Eine obere Schranke der Laufzeit, für den in [Xie02] vorgestellten Algorithmus ist* $O(n^3)$.

Beweis. Alle möglichen Kombinationen aller Kanten-Pixel ergeben eine quadratische Anzahl von Iterationsschritten. In einem Iterationsschritt werden wieder alle Kanten-Pixel betrachtet, um ihre Stimme abzugeben. Daraus resultiert eine kubische Laufzeit bezogen auf die Anzahl der Kanten-Pixel. □

Ergebnisse

Der Algorithmus wurde im Rahmen dieser Arbeit in C implementiert und evaluiert. Für synthetisch generierte Ellipsen funktioniert er sehr gut. Jedoch für Objekte, die keine perfekten Konturen einer Ellipse aufweisen, liefert dieser Algorithmus keine zufriedenstellenden Ergebnisse.

Speziell bei dem segmentierten Querschnittsbild aus Abb. 2.4 sind beim aktiven Material keine perfekten Konturen einer Ellipse zu erkennen. Aus diesem Grund scheiden alle Hough-basierten Ansätze zur Ellipsenerkennung aus. Ein Verfahren zum Einpassen (engl.: fitting) der Ellipsen ist erforderlich.

2.2.3 Least Squares Fitting

Die Methode der kleinsten Fehlerquadrate (engl.: Least Squares Fitting) wird verwendet, um Kurvenverläufe von überbestimmten Gleichungen möglichst nahe an alle Datenpunkte heranzuführen. Dazu wird die Summe der Fehlerquadrate minimiert. Die Abweichungen werden als Residuen oder Fehler bezeichnet.

Im Folgenden wird diese Methode mit einem einfachen Beispiel, einer Geraden, erläutert. Die lineare Funktion einer Geraden ist in Gl. 2.4 gegeben.

$$f(x) := \alpha_0 + \alpha_1 x \tag{2.4}$$

Um die Unbekannten α_0 und α_1 zu bestimmen, würden 2 Datenpunkte (x_i, y_i) genügen. Überbestimmt ist dieses System durch eine größere Anzahl $n > 2$ von Datenpunkten. Es muss ein Optimum gefunden werden. Abb. 2.6 zeigt ein Beispiel.

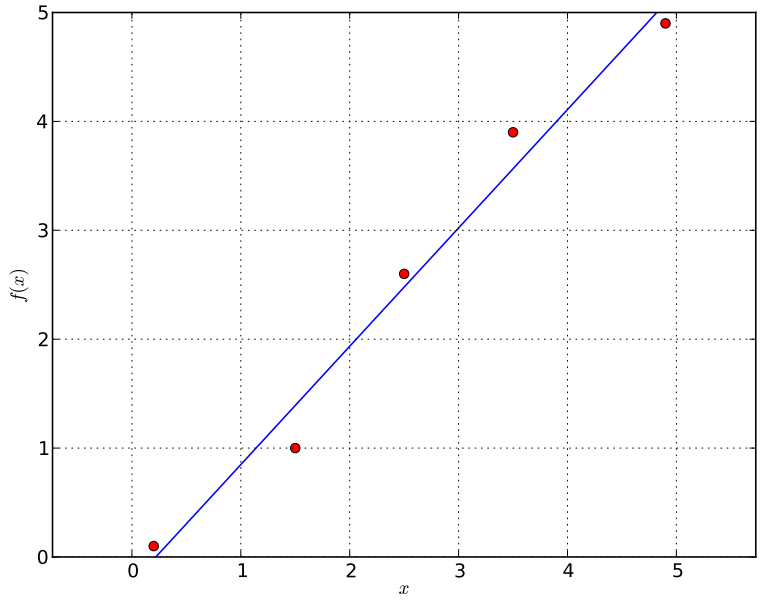


Abbildung 2.6: Beispiel zur Methode der kleinsten Fehlerquadrate anhand einer Geraden

Die Residuen r_i errechnen sich aus den Differenzen zwischen Soll- und Ist-Werten der Funktion. Gl. 2.5 formuliert diesen Zusammenhang.

$$r_i = \alpha_0 + \alpha_1 x_i - y_i \qquad i \in \{1, ..., n\} \tag{2.5}$$

Bei einer einfachen Summierung der Residuen könnten sich die positiven und die negativen Residuen gegenseitig aufheben. Eine Summierung der Beträge bringt Probleme mit der Differenzierbarkeit mit sich. Die Residuen zu quadrieren und anschließend zu summieren umgeht diese Probleme. Die Abweichungen werden in Gl. 2.6

zusammengefasst und minimiert.

$$\min_{\alpha_0,\alpha_1} \sum_{i=1}^{n} r_i^2 \tag{2.6}$$

Ein Optimum findet sich, wenn die erste Ableitung auf 0 gesetzt wird. Im mehrdimensionalen Fall wird der Gradient auf $\vec{0}$ gesetzt.

$$\nabla \sum_{i=1}^{n} r_i^2 \stackrel{!}{=} \vec{0} \tag{2.7}$$

Gl. 2.7 kann für einfache und lineare Gleichungen direkt gelöst werden. Handelt es sich um nichtlineare Gleichungen, kann die Lösung über ein iteratives Verfahren errechnet werden. Eine Standard-Verfahren hierfür ist die Methode nach Newton-Raphson.

Die Lösung für das Minimum aller Fehlerquadrate der linearen Funktion von Gl. 2.4 ist in Gl. 2.8a und Gl. 2.8b dargestellt. $\bar{x}$ beschreibt den arithmetischen Mittelwert aller x_i: $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$. Das gleiche gilt für $\bar{y}$.

$$\alpha_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2} \tag{2.8a}$$

$$\alpha_0 = \bar{y} - \alpha_1 \bar{x} \tag{2.8b}$$

[Gander94] beschreibt, wie Kreise und Ellipsen optimal in eine Menge von gegebenen Datenpunkten eingepasst werden können. Es wird zwischen der „algebraischen“ und „geometrischen“ Distanz der Residuen unterschieden. Die algebraische Distanz der Residuen ist die Differenz zwischen gegebenen Datenpunkten und Funktion. Dies entspricht dem Abstand auf der y-Achse. Die geometrische Distanz der Residuen ist jene, mit der minimalen euklidischen Distanz zwischen Funktion und Datenpunkten. Das Minimum der Fehlerquadrate wird in [Gander94] iterativ berechnet.

Eine Verbesserung zur iterativen Suche des Optimums ist in [Halir98] gezeigt. Für Ellipsen in allgemeiner Lage kann das Optimum direkt bestimmt werden.

Keines dieser Verfahren wurde implementiert. Es wurde die freie Implementierung von OpenCV[1] verwendet.

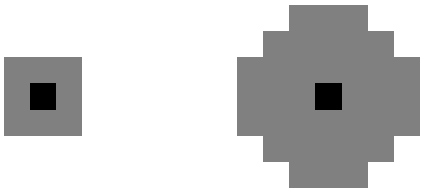
Um dieses Verfahren für das gegebene Problem anwenden zu können, muss im Vorhinein bekannt sein, welche Datenpunkte bzw. Kanten-Pixel zu welcher Ellipse gehören. Die Lösung dieses Problems wird in den folgenden Kapiteln präsentiert.

[1] `http://opencv.willowgarage.com/`

2.2.4 Morphologische Operationen

Morphologische Bildverarbeitung wird verwendet, um Objekte voneinander zu trennen oder zusammenzufügen. Bestimmte Bereiche im Bild, beispielsweise Bereiche mit einem vorgegebenen Pixel-Wert, werden hierzu betrachtet.

Um Objekte im Bild zu „schrumpfen“ wird die **Erosion** eingesetzt. Die **Dilatation** „bläht“ Objekte auf. Wie die Objekte verändert werden, wird durch ein Strukturelement festgelegt. Das Strukturelement wird an jeden Pixel der Ränder der Objekte gelegt. Bei einer Erosion werden alle, durch das Strukturelement vorgegebenen Pixel gelöscht. Bei einer Dilatation werden alle, durch das Strukturelement vorgegebenen Pixel befüllt. Beispiele verschiedener Stukturelemente sind in Abb. 2.7 zu sehen. Der schwarze Pixel in der Mitte markiert jenen Pixel, der direkt auf den Rand des Objektes gesetzt wird.

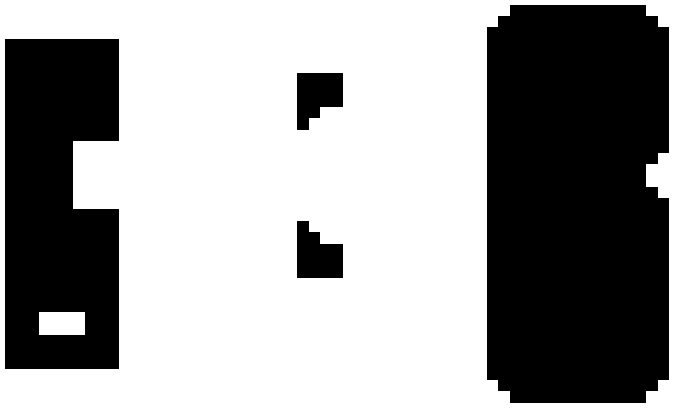


(a) Rechteckig

(b) Kreisrund (Näherung aus Diskretisierung)

Abbildung 2.7: Beispiele von Strukturelementen

Abb. 2.8 zeigt eine Erosion und eine Dilatation der gegebenen Pixel mit dem in Abb. 2.7b definiertem Strukturelement.



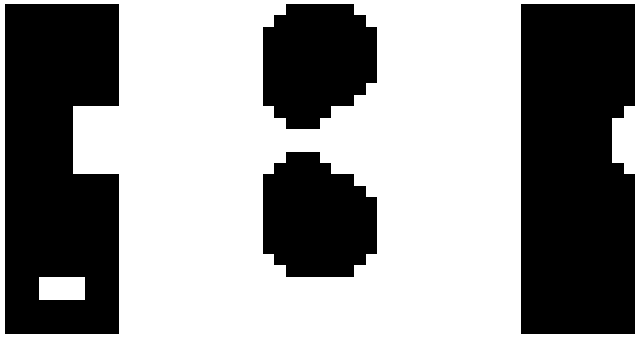
(a) Gegeben **(b)** Erosion **(c)** Dilatation

Abbildung 2.8: Beispiel von Erosion und Dilatation mit kreisrundem Strukturelement

Um Objekte voneinander zu trennen, wird die **Opening**-Operation verwendet. Beim Opening wird eine Erosion und anschließend eine Dilatation angewandt.

Einschlüsse in Objekten werden mit einer **Closing**-Operation entfernt. Beim Closing wird nach der Dilatation eine Erosion ausgeführt.

Abb. 2.9 zeigt eine Opening- und eine Closing-Operation mit dem in Abb. 2.7b gezeigtem Strukturelement.



(a) Gegeben **(b)** Opening **(c)** Closing

Abbildung 2.9: Beispiel von Opening und Closing mit kreisrundem Strukturelement

Bei der gegebenen Problemstellung erlaubt die Opening-Operation das Trennen von leicht verbundenen Ellipsen. Einschlüsse in Ellipsen können durch die Closing-Operation entfernt werden. Dadurch werden innere Konturen verhindert. Innere Konturen wirken sich bei der Methode der kleinsten Fehlerquadrate als zusätzliche Datenpunkte aus und beeinflussen somit das Ergebnis. Da sich die analytischen Beschreibungen der Ellipsen nur an die äußeren Konturen anschmiegen sollen, werden Einschlüsse durch die Closing-Operation entfernt.

2.2.5 Distanztransformation

Die Distanztransformation berechnet auf effizente Weise, für jeden Pixel, eine Annäherung der kürzesten Distanz zum nähesten Null-Wert-Pixel. Die Distanzfunktion wird durch einen Kernel definiert. Gl. 2.9 zeigt einen Kernel $k_{Euklid} \in M(3 \times 3)$ für die näherungsweise Berechnung der euklidischen Distanz. Je größer der Kernel ist, desto genauer wird die Approximation. Jedoch wirkt sich die Größe des Kernels direkt auf die Rechenzeit aus.

$$k_{Euklid} := \begin{pmatrix} \sqrt{2} & 1 & \sqrt{2} \\ 1 & x & 1 \\ \sqrt{2} & 1 & \sqrt{2} \end{pmatrix} \tag{2.9}$$

Um die Distanztransformation anschaulich zu erklären, ist in Gl. 2.10 ein Binärbild in einer Matrix img gegeben.

$$img := \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.10}$$

Die Vorwärtsabtastung ist der erste Schritt der Distanztransformation. Dabei wird ein Kernel zeilenweise von oben nach unten über jeden Pixel des Bildes geschoben. In der Zeile bewegt sich der Kernel von links nach rechts. Wenn der Mittelpunkt x des Kernels k_{Euklid} auf einem Nicht-Null-Wert-Pixel steht, wird die minimale Distanz zu einem Null-Wert-Pixel in Richtung nach oben bzw. nach links berechnet. Hierzu wird das Minimum aus der Summe der Distanz im Kernel und der schon zuvor bestimmten, darunterliegenden Distanz im Bild berechnet. Das Ergebnis wird an die aktuelle Position des Mittelpunktes des Kernels eingetragen.

Das Ergebnis der Vorwärtsabtastung von Bild img mit dem Kernel k_{Euklid} ist in Gl. 2.11 dargestellt.

$$img_{forward} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 2 & 2 & 2 & 0 \\ 0 & 1 & 2 & 3 & 0 & 0 \\ 0 & 1 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.11}$$

Das Resultat der Distanztransformation ergibt sich aus der Rückwärtsabtastung des Ergebnisses der Vorwärtsabtastung. Die Richtung, in der sich der Kernel über jedes Pixel des Bildes bewegt, wird umgedreht. Die minimale Distanz zu einem Null-Wert-Pixel in Richtung nach unten bzw. nach rechts berechnet sich analog der Vorwärtsabtastung. Ist die minimale Distanz kleiner als die bereits zuvor errechnete, wird diese im Bild aktualisiert.

Das Ergebnis der Distanztransformation von Bild img mit dem Kernel k_{Euklid} ist in Gl. 2.12 zu sehen.

$$img_{dist} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1+\sqrt{2} & \sqrt{2} & 1 & 0 \\ 0 & 1 & \sqrt{2} & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.12}$$

Laufzeit

Da der Kernel zweimal über jedes Pixel geschoben wird, ergibt sich eine obere Schranke der Laufzeit linear zur der Anzahl der im Bild vorkommenden Pixel. Die Größe des Kernels ist bei der Berechnung konstant. Aus diesem Grund geht er nicht in die Komplexität der Laufzeit ein.

Ein Nachteil eines größeren Kernels ist die Verlängerung der Rechenzeit. Ein Vorteil ist eine genauere Näherung der minimalen Distanz.

2.2.6 Eingesetztes Ellipsenerkennungsverfahren

Sei *img* ein bereits segmentiertes, diskretisiertes Bild mit ellipsenähnlichen Formen.

`detect_ellipses`(*img*)

1. *opened* = Opening-Operation(*img*) (Kap. 2.2.4)
2. *dist* = Distanztransformation(*opened*) (Kap. 2.2.5)
3. Solange max(*dist*) $\geq$ Schwellwert:
   a) *window* = Ausschnitt mit Zentrum bei max(*dist*)
   b) *interrested* = Zusammenhängende Fläche vom Zentrum des Ausschnitts *window*
   c) *closed* = Closing-Operation(*interrested*) (Kap. 2.2.4)
   d) *edge_pixels* = Kanten von *closed*
   e) *ellipse* = fit_ellipse(*edge_pixels*) (Kap. 2.2.3)
   f) Setze alle Pixel von *interrested* in *dist* auf 0

Algorithmus 2.2: Ellipsenerkennung

Der Algorithmus `detect_ellipses` erkennt Ellipsen in einem bereits segmentierten Querschnittsbild einer Elektrode. Zur Nachvollziehbarkeit wird dieser Algorithmus schrittweise mit einem Beispiel erläutert. Als Eingabe-Bild dient das Ergebnis-Bild der Segmentierung aus Abb. 2.4.

Aus dem gegebenen Bild wird die Klasse des Aktivmaterials extrahiert. Die Opening-Operation im ersten Schritt trennt leicht aneinander grenzende Flächen. Das Resultat der Opening-Operation ist in Abb. 2.10 dargestellt. Kleine Fraktale des Aktivmaterials werden dadurch ebenfalls entfernt.

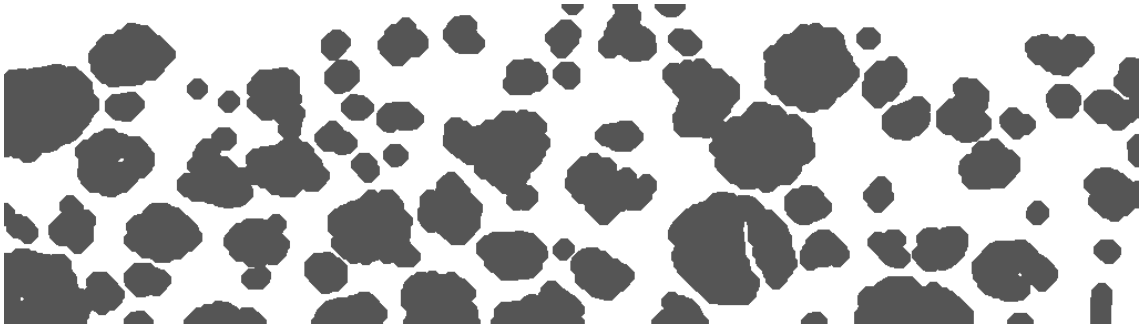


Abbildung 2.10: Binärbild des Aktivmaterials nach Opening-Operation

Das erhaltene Binärbild wird der Distanztransformation zugeführt. Im Anschluss wird das globale Maximum des Distanzbildes von Abb. 2.11 gesucht. Der Schwellwert

des Maximums legt die minimale Größe der erkennbaren Ellipsen fest.

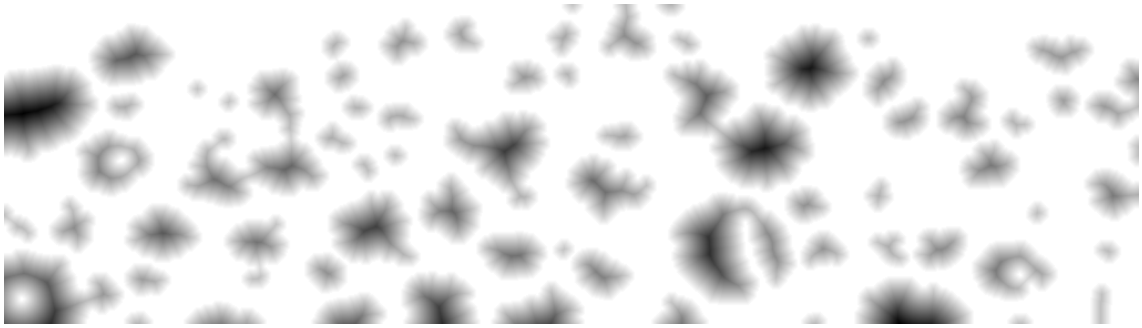


Abbildung 2.11: Distanztransformation des Binärbildes

Das globale Maximum der Distanz dient zur Positionierung eines Ausschnittsfensters. Die Größe des Ausschnittsfensters legt die maximale Größe der erkennbaren Ellipsen fest. Im Ausschnittsfenster wird, ausgehend vom Maximum der Distanz, eine zusammenhängende Fläche gesucht. Auf die Pixel dieser Fläche wird eine Closing-Operation angewandt. Diese entfernt eventuelle Einschlüsse und erhält die äußere Kontur der Ellipse. Durch die Methode der kleinsten Fehlerquadrate lässt sich eine Ellipse der Kontur anschmiegen. Dieser Vorgang ist in Abb. 2.12 gezeigt.

(a) Ausschnittsfenster **(b)** Closing **(c)** Ellipse Fitting

Abbildung 2.12: Operationen im Auschnittsfenster

Abschließend werden alle Pixel der zuvor bestimmten zusammenhängenden Fläche im Distanzbild auf Null gesetzt. Das Resultat eines Iterationsschrittes ist die analytische Beschreibung einer Ellipse. Die Iteration wird so oft wiederholt, bis das globale Maximum der verbleibenden Distanzen unter einen vorgegebenen Schwellwert fällt.

Ergebnis

Das Ergebnis von `detect_ellipses` ist in Abb. 2.13 zu sehen. Zu erkennen ist, dass die zusammenhängenden Bereiche aus Abb. 2.10 verwendet werden können, um Ellipsen optimal einzupassen und analytisch zu beschreiben.

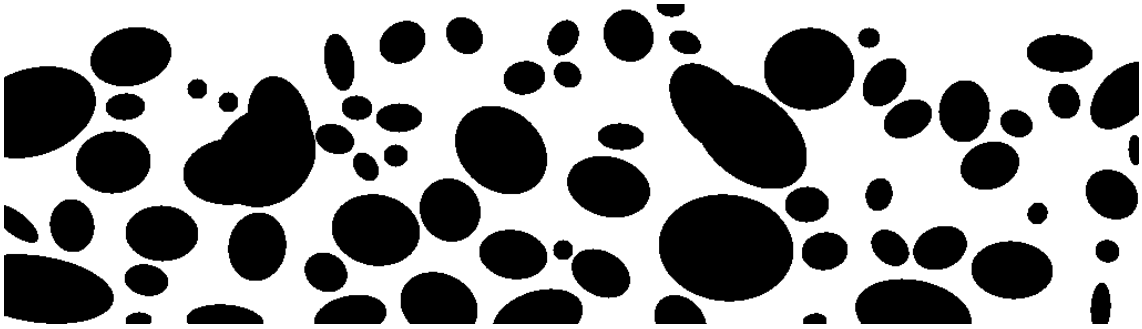


Abbildung 2.13: Aktivmaterial als Ellipsen

Wenn die Konturen der Ellipsen durch die Opening-Operation nicht voneinander getrennt werden können, verzerren sich die erhaltenen Ellipsen in Richtung der Überlagerung. Dies ist in Abb. 2.14 dargestellt.

(a) Ausschnittsfenster **(b)** Zusammenhängender Bereich **(c)** Ellipse Fitting

Abbildung 2.14: Verzerrung bei starker Überlagerung

Durch eine Verkleinerung des Ausschnittsfensters verringert sich die Verzerrung. Die untere Grenze für die Größe des Ausschnittsfensters ist durch die größte im Bild vorkommende Ellipse gegeben. Erkennbar ist die Verzerrung bei starken Überlagerungen, indem überprüft wird, ob die Kontur des zusammenhängenden Bereiches an den Rand des Ausschnittsfensters stößt. Ist dies der Fall, ist entweder das Ausschnittsfenster zu klein gewählt oder die Ellipsen werden bei starken Überlagerungen verzerrt erkannt.

Das Ergebnis der Ellipsenerkennung wird im folgenden Kapitel für eine statistische Analyse verwendet. Die selten auftretenden, geringfügigen Verzerrungen haben dort eine geringe Auswirkung.

3 Stochastik

In diesem Kapitel werden für die Lösung der Problemstellung (Kap. 1.2) relevanten Bereiche aus der Wahrscheinlichkeitstheorie und Statistik betrachtet. Vergleichbar mit dem Lösungsansatz aus Kap. 1.3 soll eine Summe von Dichtefunktionen in Komponenten zerlegt werden. Anders als beim Lösungsansatz wird hier von 2-dimensionalen Querschnittsinformationen auf die wahrscheinlichsten 3-dimensionalen Geometrien gerechnet. Bei der gegebenen Problemstellung handelt es sich um Ellipsoide, die als Ellipsen im Querschnittsbild zu sehen sind.

3.1 Expectation-Maximization-Algorithmus

Der Expectation-Maximization-Algorithmus, kurz EM-Algorithmus, ist das Standardverfahren zur Dekomposition von „Mixture Models“ [Bishop07, Kap. 9]. Unter „Mixture Models“ wird eine Mixtur von Wahrscheinlichkeitsmodellen verstanden. Bester Vertreter hierfür ist das „Mixture of Gaussians“-Modell. Bei diesem Modell wird angenommen, dass die Stichprobenelemente aus einer Grundgesamtheit von überlagernden Normalverteilungen entnommen werden. Wenn die Anzahl der Normalverteilungen bekannt ist, kann der EM-Algorithmus jedes Stichprobenelement der jeweiligen Normalverteilung zuordnen. Man spricht von Einteilung der Stichprobenelemente in Cluster, oder kurz Clustering.

Die Wahrscheinlichkeitsdichte einer d-dimensionalen, normalverteilten Zufallsvariable X ist in Gl. 3.1 gegeben. Dies ist die multivariate Darstellung der Dichte der Gauß-Verteilung. Sie ist bestimmt durch den Erwartungswertvektor μ und der Kovarianzmatrix Σ. Die Determinante der Kovarianzmatrix ist durch $|\Sigma|$ dargestellt.

$$f_X(x) = \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right) \tag{3.1}$$

Sei n die Anzahl der d-dimensionalen Stichprobenelemente $x_{.,j}$. Eine Schätzung für den Erwartungswertvektor μ ergibt sich aus den Mittelwerten $\bar{x_i}$. Dieser Zusammenhang ist in Gl. 3.2 dargestellt. Gl. 3.3 zeigt, wie für jede Dimension $i \in \{1, ..., d\}$ ein Mittelwert $\bar{x_i}$ berechnet wird.

$$\mu \approx \bar{x} = \begin{pmatrix} \bar{x_1} \\ \vdots \\ \bar{x_d} \end{pmatrix} \tag{3.2}$$

$$\bar{x_i} = \frac{1}{n} \sum_{j=1}^{n} x_{i,j} \qquad i \in \{1, ..., d\} \tag{3.3}$$

Eine Approximation der Kovarianzmatrix Σ ist in Gl. 3.4 beschreiben.

$$\Sigma \approx \begin{pmatrix} \mathrm{Cov}(x_1, x_1) & \dots & \mathrm{Cov}(x_1, x_d) \\ \vdots & \ddots & \vdots \\ \mathrm{Cov}(x_d, x_1) & \dots & \mathrm{Cov}(x_d, x_d) \end{pmatrix} \tag{3.4}$$

Jedes Element der Kovarianzmatrix Σ wird nach Gl.3.5 berechnet.

$$\mathrm{Cov}(x_k, x_l) = \frac{1}{n-1} \sum_{j=1}^{n} (x_{k,j} - \bar{x_k})(x_{l,j} - \bar{x_l}) \tag{3.5}$$

Der EM-Algorithmus ist ein iteratives Verfahren mit einer Initialisierung und 2 Schritten. Jeder Cluster ist bestimmt durch die Dichtefunktion mit dem Erwartungswertvektor μ und der Kovarianzmatrix Σ. Jedes Stichprobenelement x wird genau einem Cluster zugeordnet.

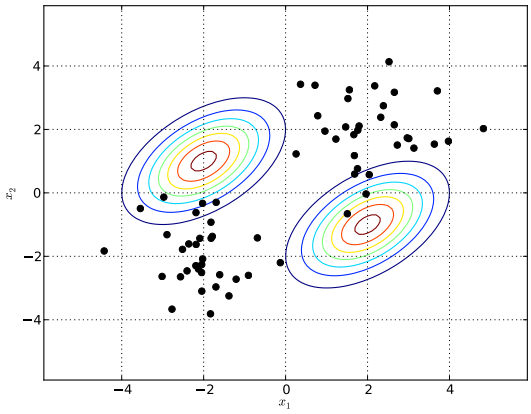
Bei der Initialisierung wird μ und Σ für jeden Cluster auf unterschiedlich vorgegebene Werte gesetzt.

Im 1. Schritt werden alle Dichtefunktionen (Gl. 3.1) für alle Stichprobenelemente ausgewertet. Für jedes Stichprobenelement wird das Maximum aus den Ergebnissen der Dichtefunktionen gesucht. Die Stichprobenelemente werden durch das Maximum dem jeweiligen Cluster zugeordnet. Das Zuordnen eines Stichprobenelementes zum wahrscheinlichsten Cluster wird **Maximum-Likelihood**-Methode genannt.

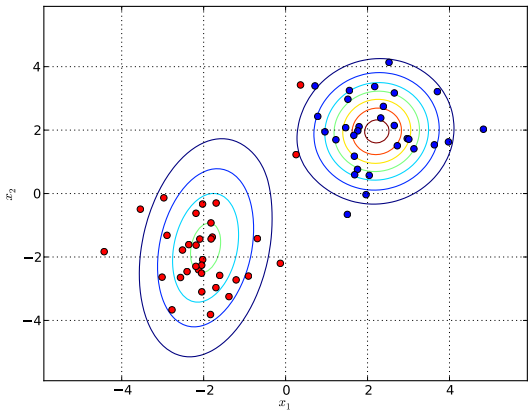
Im 2. Schritt werden μ und Σ aktualisiert. Die einem Cluster zugeordneten Stichprobenelemente dienen als Eingabewerte zur erneuten Bestimmung des Erwartungswertvektors μ (Gl. 3.2) und der Kovarianzmatrix Σ (Gl. 3.4).

Diese beiden Schritte werden so lange wiederholt, bis das Verfahren konvergiert. Das Ergebnis ist eine Zuordnung jedes Stichprobenelementes zu einem Cluster. Die Erwartungswertvektoren und die Kovarianzmatrizen der Dichtefunktionen der Cluster werden ebenfalls berechnet.

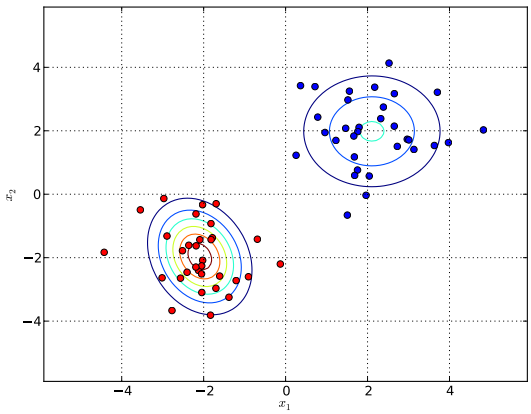
Ein Beispiel des Ablaufs ist in Abb. 3.1 zu sehen. Die Punkte zeigen die Werte der Stichprobenelemente. Die Dichten der Gauß-Verteilungen sind als „Höhenlinien“ dargestellt.



(a) Initialisierung



(b) Nach 4 Iterationen



(c) Ergebnis

Abbildung 3.1: Ablauf des EM-Algorithmus

Das Ergebnis aus Kap. 2.2.6 ist eine vollständige Beschreibung aller im Querschnittsbild vorkommenden Ellipsen. Eine Ellipse ist durch 5 Variablen beschrieben. Dazu zählt die Haupt- und Nebenhalbachse (a, b), ein Rotationswinkel α und die Position des Mittelpunkts (x_0, y_0).

Mit Hilfe dieser Informationen soll errechnet werden, welche Ellipsoide geschnitten wurden. Gl. 3.6 zeigt eine analytische Darstellung eines Ellipsoids in Hauptlage. Hauptlage bedeutet, dass der Mittelpunkt in den Koordinatenursprung fällt und dass die Achsen parallel zu den Achsen des Koordinatensystems verlaufen.

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} = 1 \tag{3.6}$$

Es wird angenommen, dass das geschnittene Material an jeder Stelle die gleichen Eigenschaften aufweist. Aus diesem Grund ist die Position der Mittelpunkte irrelevant für eine statistische Auswertung. Je komplexer die Zusammenhänge der Variablen beschrieben werden, desto größer muss die Anzahl der Stichprobenelemente sein, um eine repräsentative Auswertung zu erhalten. Da für die gegebene Problemstellung ein relativ kleiner Ausschnitt vorhanden ist und deshalb die Anzahl der Stichprobenelemente sehr eingeschränkt ist, wird der Rotationswinkel α als eine von der Größe unabhängige Variable angenommen. Damit ergeben sich 2-dimensionale Stichprobenelemente mit den Einträgen (a, b).

Im Folgenden wird versucht eine Beschreibung einer Dichtefunktion eines Ellipsoids über vereinfachte diskrete Überlegungen zu erhalten. Der Ellipsoid wird in diskreten Abständen in der Tiefe bzw. der z-Achse geschnitten. Daraus erhält man mehrere Ellipsen mit den jeweiligen Haupt- und Nebenhalbachsen (a, b). Die erhaltenen Werte für a und b können wieder (vergl. Kap. 1.3) nach ihrer Größe in diskrete Positionen in eine Tabelle bzw. einer Matrix eingeordnet und gezählt werden. Analytisch betrachtet ergibt sich eine Kurve in $\mathbb{R}^3$. Die Kurve steigt bei größeren Werten für a und b an. Jedoch handelt es sich nur um eine Kurve, ohne Ausdehnung in die Breite. Da die Wahrscheinlichkeit für ein beliebiges Stichprobenelement (a, b) immer 0 ist, außer (a, b) liegt genau auf dieser Kurve, ist die auf diese Weise definierte Dichtefunktion für die Maximum-Likelihood-Methode ungeeignet. Einfacher nachzuvollziehen ist das anhand eines Beispiels. Ein Ellipsoid mit den Halbachsen $a = 3, b = 2, c = 3$ hat bei allen möglichen Querschnitten keine Ellipse mit beispielsweise $a = 6, b = 5$. Die Auswertung der Dichtefunktion ergibt für diese Werte 0.

Aus diesem Grund funktioniert ein einfacher EM-Algorithmus für diese Problemstellung nicht. Es ist nicht möglich, die Stichprobenelemente mit der Maximum-Likelihood-Methode einem Cluster zuzuordnen.

Eine mögliche Lösung ist, die Dichtefunktion als Fläche anstatt als Kurve zu modellieren. Dies erfordert einen zusätzlichen Parameter, der den Abfall der Wahrscheinlichkeit außerhalb der Kurve beschreibt. Dadurch wird allen (a, b) eine Wahrscheinlichkeit zugeordnet. Die Wahrscheinlichkeiten entsprechen außerhalb der Kurve nicht den Tatsachen, ermöglichen es jedoch, den EM-Algorithmus für dieses Problem einzusetzen.

Eine Möglichkeit die tatsächliche Dichte-Kurve als Dichtefunktion zu verwenden wäre ein Hybrid-Verfahren aus Least-Squares-Fitting (Kap. 2.2.3) und Expectation-Maximization-Algorithmus. Dieser relativ aufwändige Ansatz, den EM-Algorithmus für diese Problemstellung einzusetzen, wird hier nicht weiter verfolgt, da durch den Lösungsansatz von Kap. 1.3 bereits ein einfacherer Ansatz existiert.

3.2 Verallgemeinerung des Lösungsansatzes

Der Lösungsansatz (Kap. 1.3) geht von einem statistisch repräsentativen 1-dimensionalen Querschnitt aus und konstruiert mit den Informationen aus dem 1-dimensionalen Querschnitt ein statistisch vergleichbares 2-dimensionales Bild. Die Informationen, die verwendet werden, sind die Mengenverhältnisse und die Längenverteilung.

Die Mengenverhältnisse gelten laut Lemma 1.1 direkt für höhere Dimensionen. Diese Erkenntnis trifft in gleicher Weise auf Mengenverhältnisse eines statistisch repräsentativen 2-dimensionalen Querschnitts einer 3-dimensionalen Geometrie zu.

Um die Längenverteilung zu bestimmen, werden die geschnittenen Bereiche nach ihren Längen eingeteilt und gezählt. Es ist nicht möglich, diesen Ansatz direkt bei höheren Dimensionen anzuwenden. Eine mögliche 2-dimensionale Interpretation der zuvor verwendeten Einteilung der Längen wäre, die Größe der Flächen einzuteilen und zu zählen. Jedoch gehen dadurch alle Informationen der geometrischen Form der Fläche bzw. der Geometrie verloren. Daher ist diese Möglichkeit nicht zielführend.

Die nächste Idee ist, die Ausdehnungen der Flächen in x- und y-Richtung zu betrachten. Um wieder zu vermeiden, dass verschiedene geometrische Formen der Fläche eine ähnliche Verteilung der Längen bzw. der Größen zur Folge haben, wird die Ausdehnung in x- und y-Richtung an jeder Position bzw. an jedem Pixel einmal bestimmt. Es ergibt sich somit eine 2-dimensionale „Längenverteilung“. Für jeden Pixel wird genau ein Wert in der Matrix der 2-dimensionalen „Längenverteilung“ inkrementiert. Die Position des Wertes wird durch die Ausdehnung in x- und y-Richtung bestimmt.

Da diese Idee eine Verallgemeinerung des Lösungsansatzes darstellt, muss dieses Verfahren auch für den Lösungsansatz selbst funktionieren. Wird die Ausdehnung der Länge bei jedem Pixel bestimmt, eingeteilt und gezählt, ergibt sich eine andere Längenverteilung als beim Lösungsansatz selbst. Eine Schnittlänge mit beispielsweise 3 Pixel wird nicht 1-mal, sondern 3-mal gezählt. Um dies zu korrigieren, kann man die bestimmte Anzahl der jeweiligen Längen durch die Länge selbst dividieren, oder man definiert die Dichtefunktion passend. Eine passend definierte Dichtefunktion hat den Vorteil, dass die Divisionen eingespart werden.

Sei $2d\_shapes$ eine Liste von 2-dimensionalen, analytisch beschriebenen geometrischen Formen (beispielsweise Ellipsen). Der Algorithmus `create_mixture`($2d\_shapes$) generiert aus einer Liste von 2-dimensionalen geometrischen Formen ein, in eine Matrix M diskretisiertes, Mixture-Model.

`create_mixture`($2d\_shapes$)

1. Initialisiere die Mixture-Model-Matrix M
2. Für jede geometrische Form $s \in 2d\_shapes$:
   a) Rotiere s in Hauptlage
   b) Generiere äquidistant diskretisiertes Bild img von s.
   c) Für alle Pixel $pix \in img$ im Inneren von s:
      i. $pval$ = Wert des Pixels pix
      ii. x_{range} = Länge des Bereichs mit gleichem Wert wie $pval$ in x-Richtung, ausgehend von pix
      iii. y_{range} = Länge des Bereichs mit gleichem Wert wie $pval$ in y-Richtung, ausgehend von pix
      iv. Inkrementiere Zähler $m[y_{range}, x_{range}]$ der Mixture-Model-Matrix M

Algorithmus 3.1: Generierung der Mixture-Model-Matrix aus 2-dimensionalen geometrischen Formen

Vergleichbar mit dem Lösungsansatz (Kap. 1.3) wird dieses Mixture-Model als Summe von Dichtefunktionen gesehen. Das Mixture-Model wird aus dem 2-dimensionalen Querschnittsbild generiert. Die Dichtefunktionen stammen von den 3-dimensionalen geometrischen Formen ab.

Sei $3d\_shape$ eine 3-dimensionale, analytisch beschriebene, geometrische Form (beispielsweise ein Ellipsoid). Der Algorithmus `density`($3d\_shape$) berechnet für beliebige, 3-dimensionale, geometrische Formen die Dichtefunktion diskret. Durch die Diskretisierung ist das Resultat eine Dichte-Matrix P. Die Mixture-Model-Matrix M wird im Anschluss als Summe von Dichte-Matrizen gesehen.

`density`($3d\_shape$)

1. Initialisiere Dichte-Matrix P
2. (Optional: Rotiere $3d\_shape$ in Hauptlage)
3. Generiere äquidistant diskretisiertes Volumen vol aus $3d\_shape$
4. Für jede Ebene bzw. jedes Schnittbild $img \in vol$ in z-Richtung:
   a) Für jeden Pixel $pix \in img$ im Inneren von $3d\_shape$:
      i. $pval$ = Wert des Pixels pix
      ii. x_{range} = Länge des Bereichs mit gleichem Wert wie $pval$ in x-Richtung, ausgehend von pix
      iii. y_{range} = Länge des Bereichs mit gleichem Wert wie $pval$ in y-Richtung, ausgehend von pix
      iv. Inkrementiere Zähler $p[y_{range}, x_{range}]$ in der Dichte-Matrix P
5. Normiere Dichte-Matrix P auf 1

Algorithmus 3.2: Erstellung einer Dichte-Matrix einer 3-dimensionalen geometrischen Form

Hier ist zu beachten, dass die Rotation der 3-dimensionalen geometrischen Form im Raum eine Auswirkung auf das Ergebnis der Berechnung hat. Die Rotation eines Körpers im Raum kann durch 2 Winkel beschrieben werden. Es darf keine Rotation um die z-Achse vorhanden sein, da bei der Generierung der Mixture-Model-Matrix von
`create_mixture`($2d\_shapes$) alle 2-dimensionalen Geometrien in Hauptlage rotiert werden. Der zweite Winkel der 3-dimensionalen Geometrie wird aufgrund Vereinfachungen ebenfalls in Hauptlage rotiert. Wenn die Rotation zusätzlich bestimmt werden soll, ist dies ein Parameter der 3-dimensionalen Geometrie mehr und erfordert eine dementsprechend größere Anzahl an Stichprobenelementen.

Nach der statistischen Auswertung der Stichproben werden die berechneten 3-dimensionalen geometrischen Formen rekonstruiert. Bei der Rekonstruktion werden die 3-dimensionalen Geometrien nach der Verteilung der Rotationswinkel der 2-dimensionalen Geometrien um die z-Achse rotiert.

Eine positive Eigenschaft hat die durch `density` definierte Dichte-Matrix P eines Ellipsoids. Das Maximum der Dichte-Matrix P eines Ellipsoids liegt in der Gegend von $(2a, 2b)$. Durch die Diskretisierung kann es zu geringfügigen Abweichungen der Position des Maximums kommen, die durch eine erneute Annahme der Parameter des Ellipsoids korrigiert werden. Das Maximum am Rande von P vereinfacht die Dekomposition der Mixture-Model-Matrix M erheblich. Der Algorithmus `decompose_ellipsoids` veranschaulicht den Ablauf.

`decompose_ellipsoids`(M)

1. (num_rows, num_cols) = Größe der Mixture-Model-Matrix M
2. Für r = von $num\_rows - 1$ bis 0,
   und c = von $num\_cols - 1$ bis 0:
   a) Wenn $m_{r,c} \geq$ Schwellwert:
      i. Suche Dichte-Matrix P mit `density`$(2d\_shape)$
      mit Maximum bei $p_{r,c}$
      durch Anpassung der Parameter eines Ellipsoids e
      ii. Relative Häufigkeit $h = \frac{m_{r,c}}{p_{r,c} \cdot \text{Volume}(e)}$
      iii. Entferne erkannte Dichte $M = M - h \cdot P$
      iv. Füge (e, h) dem Ergebnis $result$ hinzu
3. Normiere relative Häufigkeiten in $result$ auf 1

Algorithmus 3.3: Trennung der Mixture-Model-Matrix

Eine analytische Beschreibung eines Ellipsoids in Hauptlage ist in Gl. 3.7 dargestellt.

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} = 1 \tag{3.7}$$

Die Variablen a und b werden variiert, um ein Maximum an der gewünschten Position der Dichte-Matrix P zu erhalten. Der Parameter c wird auf a gesetzt. Dadurch ergibt sich die Form eines Diskus. Ellipsoide in Form eines Reiskorns sind durch diese Annahme nicht möglich.

Wenn die Anzahl an Stichprobenelemente ausreichend groß ist, kann jeder Parameter, wie beispielsweise c oder die Rotation bzw. Neigung des Diskus, bestimmt werden. Das Trennen der Mixture-Model-Matrix M wird zu einer Suche, welche Dichte-Matrizen das Mixture-Model optimal trennen. Durch eine Suche des Optimums ist es zusätzlich möglich, die Form der Geometrie zu bestimmen. Bei einer eingeschränkten Anzahl von Stichprobenelementen ist dies nicht durchführbar. Das Ergebnis ist dann statistisch nicht repräsentativ.

3.3 Von der Wahrscheinlichkeit zur Anzahl

Die Wahrscheinlichkeiten bzw. die relativen Häufigkeiten h_i sind in Kapitel 3.2 bestimmt worden. Der Index i beschreibt hier die i-te Größe (a_i, b_i, c_i) eines Ellipsoids. Die relativen Volumenverhältnisse ergeben sich laut Lemma 1.1 direkt. Das Volumenverhältnis des Volumens des aktiven Materials V_{AktMat} bezogen auf das Gesamtvolumen V_{Gesamt} wird durch $\varepsilon' = \frac{V_{AktMat}}{V_{Gesamt}}$ repräsentiert. Das Volumen des aktiven Materials ist eine Summe aller Volumen der Ellipsoide.

Aus den Volumenverhältnissen von Gl. 3.8b wird in Gl. 3.8c, durch Umformung, ein Häufigkeitsparameter k ermittelt.

$$\varepsilon' = \frac{V_{AktMat}}{V_{Gesamt}} \tag{3.8a}$$

$$\varepsilon' = \frac{\sum_{i=1}^{n} k \cdot h_i \cdot V_i}{V_{Gesamt}} \tag{3.8b}$$

$$k = \frac{\varepsilon' \cdot V_{Gesamt}}{\sum_{i=1}^{n} h_i \cdot V_i} \tag{3.8c}$$

Das Volumen V des Ellipsoids ergibt sich aus den Halbachsen a, b, c wie in Gl. 3.9 dargestellt.

$$V = \frac{4}{3}\pi abc \tag{3.9}$$

Die absolute Häufigkeit H_i beschreibt, wie viele Ellipsoide der i-ten Größe im Gesamtvolumen V_{Gesamt} zu finden sein werden. Sie ergibt sich, laut Gl. 3.10, aus der relativen Häufigkeit h_i.

$$H_i = k \cdot h_i \tag{3.10}$$

Bei dieser Berechnung ist die Nicht-Überlagerung der Ellipsoide zu beachten. Wenn sich die Ellipsoide überlagern, weicht die oben getroffene Berechnung von $V_{AktMat} = \sum_{i=1}^{n} k \cdot h_i \cdot V_i$ vom tatsächlichen Wert ab.

Aus diesem Grund werden die Ellipsoide bei der Rekonstruktion mit möglichst wenig Überlagerungen im Raum angeordnet.

Nachdem die Anzahl und Größe der Ellipsoide bestimmt wurden, wird im Kap. 4 beschrieben, wie sie in ein gegebenes Volumen eingefügt werden.

4 Geometriegenerierung

In diesem Kapitel wird beschrieben, wie eine statistisch plausible Rekonstruktion der 3-dimensionalen Geometrie durchgeführt wird. Im ersten Schritt werden alle in Kap. 3.3 berechneten Ellipsoide eingefügt (Kap. 4.1). Anschließend werden die Ellipsoide so angeordnet, dass möglichst wenige Überlagerungen auftreten (Kap. 4.2).

4.1 Einfügen der Ellipsoide

Sei V das Gesamtvolumen und L eine Liste mit n einzufügenden Ellipsoiden.

`insert_ellipsoids`(V, L)

1. Sortiere L absteigend nach benötigtem Volumen.
2. Für jeden Ellipsoid $e \in L$:
   a) Bestimme randomisiert ein freies Volumenelement v aus V.
   b) Setze Mittelpunkt c von e auf v.
   c) Fülle Ellipsoidvolumen ausgehend von c.

Algorithmus 4.1: Einfügen von Ellipsoiden

Korrektheit

Lemma 4.1. *`insert_ellipsoids`(V, L) fügt alle Ellipsen $\in L$ in das Volumen V ein.*

Beweis. Laut (2a) können Elemente eingefügt werden, solange es freie Stellen im Volumen gibt. Wenn alle einzufügenden Elemente in das Volumen passen, können diese auch eingefügt werden. □

Laufzeit

Lemma 4.2. *`insert_ellipsoids`(V, L) benötigt $O(n \log n)$ Zeit.*

Beweis. Das Sortieren in (1) benötigt $O(n \log n)$ Zeit. Die Schleife über alle Elemente in (2) benötigt $O(n)$ Zeit. Insgesamt ergibt sich somit eine Laufzeit von $O(n \log n)$. Diese Laufzeit kann nur erreicht werden, wenn die Implementierung in (2a) ein freies Volumenelement in konstanter Zeit ($O(1)$) liefern kann. □

4.2 Anordnung der Ellipsoide

Dieses Kapitel beginnt mit einer Herleitung der benötigten euklidischen Distanz zwischen den Mittelpunkten zweier Ellipsoide in allgemeiner Lage im Raum. Ausgehend davon werden Bewegungsvektoren errechnet, welche die Ellipsoide verschieben, um möglichst wenig Überlagerungen zu erhalten.

Eine analytische Beschreibung eines Ellipsoids mit den Halbachsen a, b, c und einer Ausrichtung in Hauptlage ist in Gl. 4.1 zu sehen. Hauptlage bedeutet, dass der Mittelpunkt im Koordinatenursprung liegt und dass die Halbachsen parallel zu den Koordinatenachsen verlaufen.

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} = 1 \tag{4.1}$$

Eine Gerade im Raum ($\mathbb{R}^3$) kann durch Multiplikation eines Skalars λ mit einem Richtungsvektor $\vec{u} = (u_x, u_y, u_z)^T$ beschrieben werden. Die Darstellung in Gl. 4.2 geht davon aus, dass die Gerade durch den Ursprung $\vec{0}$ verläuft.

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \lambda \cdot \begin{pmatrix} u_x \\ u_y \\ u_z \end{pmatrix} \tag{4.2}$$

Die zwei Schnittpunkte $\vec{p}_1, \vec{p}_2 \in \mathbb{R}^3$ der Geraden mit dem Ellipsoid errechnen sich durch Kombination ihrer Gleichungen. Das Ergebnis ist in Gl. 4.4a und Gl. 4.4b zu sehen.

$$\frac{\lambda^2 u_x^2}{a^2} + \frac{\lambda^2 u_y^2}{b^2} + \frac{\lambda^2 u_z^2}{c^2} = 1 \tag{4.3a}$$

$$\lambda^2 = \frac{1}{\frac{u_x^2}{a^2} + \frac{u_y^2}{b^2} + \frac{u_z^2}{c^2}} \tag{4.3b}$$

$$\vec{p}_1 = \lambda_1 \cdot \vec{u} \tag{4.4a}$$

$$\vec{p}_2 = \lambda_2 \cdot \vec{u} \tag{4.4b}$$

Da sich der Ellipsoid in Hauptlage befindet und die Gerade durch den Ursprung $\vec{0}$ verläuft, ist die euklidische Distanz d beider Punkte $\vec{p}_1, \vec{p}_2$ gleich.

$$d(\vec{0}, \vec{p}_1) = d(\vec{0}, \vec{p}_2) = \sqrt{p_x^2 + p_y^2 + p_z^2} \tag{4.5}$$

Die beschriebenen Zusammenhänge ermöglichen es, die benötigte Distanz eines Ellipsoids (a, b, c), ausgehend vom Mittelpunkt des Ellipsoids in eine gegebene Richtung $\vec{u}$ zu errechnen.

Da die vorliegenden Ellipsoide nicht in Hauptlage sind, wird eine Methode benötigt, um Ausdehnungen von rotierten Ellipsoiden zu berechnen. Dazu wird angenommen, dass sich der Ellipsoid in Hauptlage befindet. Statt Rotation des Ellipsoids wird der Richtungsvektor $\vec{u}$ entgegengesetzt rotiert. Ein Vektor $\vec{v} \in \mathbb{R}^3$ kann durch Transformationsmatrizen im Raum skaliert, rotiert u. dgl. werden.

Ein Vektor $\vec{v} \in \mathbb{R}^3$ kann um eine Ursprungsgerade in Richtung des Vektors $\vec{n} \in \mathbb{R}^3$ um den Winkel α mit der Transformationsmatrix in Gl. 4.7 rotiert werden. Hierbei ist zu beachten, dass der Vektor $\vec{n}$ normiert $\|\vec{n}\| = 1$ vorliegen muss. Die Transformationsmatrix der Rotation ist aus [Redbook07, Appendix G] entnommen.

$$c := \cos(\alpha) \tag{4.6a}$$
$$s := \sin(\alpha) \tag{4.6b}$$

$$\vec{v}_{rotated} = \begin{pmatrix} n_x^2(1-c)+c & n_x n_y(1-c)-n_z s & n_x n_z(1-c)+n_y s \\ n_x n_y(1-c)+n_z s & n_y^2(1-c)+c & n_y n_z(1-c)-n_x s \\ n_x n_z(1-c)-n_y s & n_y n_z(1-c)+n_x s & n_z^2(1-c)+c \end{pmatrix} \cdot \vec{v} \tag{4.7}$$

Mit den hier beschriebenen Gleichungen ist es möglich, die euklidische Distanz zwischen dem Mittelpunkt und der Oberfläche eines Ellipsoids in allgemeiner Lage, in Richtung $\vec{r}$, zu bestimmen. Zu Beginn muss der Richtungsvektor $\vec{r}$ gegen die Rotation des Ellipsoids nach Gl. 4.7 rotiert werden. Der Schnittpunkt der aufgespannten Geraden des Richtungsvektors lässt sich nach Gl. 4.3b und Gl. 4.4a errechnen. Abschließend ist die Distanz nach Gl. 4.5 zu bestimmen.

Sei L eine Liste mit n Ellipsoiden in allgemeiner Lage $\in \mathbb{R}^3$.

`rearrange_content`(L)

1. Erstelle Liste M der Länge n mit Bewegungsvektoren $\in \mathbb{R}^3$.
2. Für jedes Element $e_1 \in L$:
   a) Setze c_1 auf Mittelpunkt von e_1.
   b) Für jedes Element $e_2 \in L : index_L(e_2) > index_L(e_1)$:
      i. Setze c_2 auf Mittelpunkt von e_2.
      ii. Gegebene Distanz $dist = d(c_1, c_2)$
      iii. Richtungsvektor $\vec{r} = c_2 - c_1$
      iv. Bestimme benötigte Distanz d_1 von e_1 in Richtung $\vec{r}$.
      v. Bestimme benötigte Distanz d_2 von e_2 in Richtung $-\vec{r}$.
      vi. Distanz zwischen Mittelpunkten $dist\_req = d_1 + d_2$
      vii. Wenn $dist < dist\_req$:
         A. $a = dist\_req - dist$
         B. Teile a indirekt proportional zu Volumen von e_1 und e_2 in a_1 und a_2 auf.
         C. Skaliere $\vec{r}$ zu Länge von a_1 und addiere Vektor zu Bewegungsvektor $\in M$ von e_1.
         D. Skaliere $-\vec{r}$ zu Länge von a_2 und addiere Vektor zu Bewegungsvektor $\in M$ von e_2.
3. Für jedes Element $\vec{m} \in M$:
   a) Bewege zugehöriges Element $e \in L$ um $\vec{m}$.

Algorithmus 4.2: Anordnung der Ellipsoide

Je öfter dieser Algorithmus angewendet wird, desto weniger Überlagerungen existieren. Dadurch nähert sich die Porosität ε schrittweise von oben an den gewünschten Wert heran.

Korrektheit

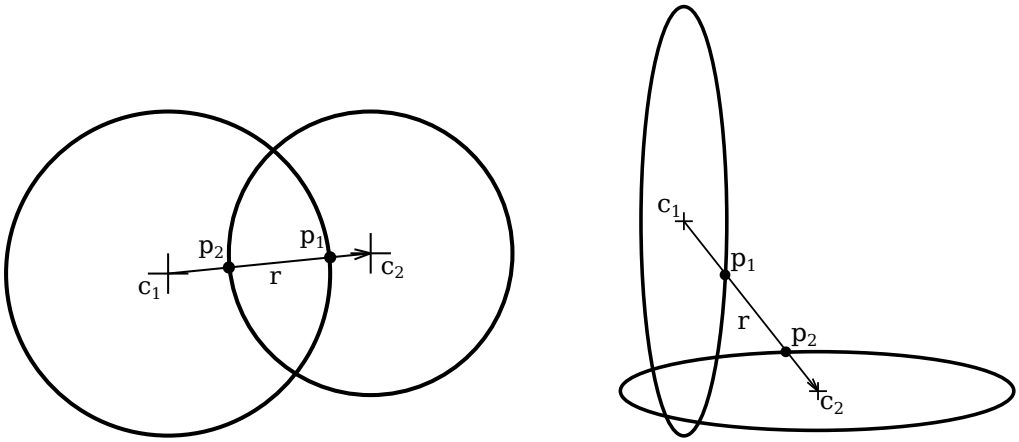
Lemma 4.3. *Bei zufällig angeordneten Ellipsoiden verringert der Algorithmus* `rearrange_content` *das überlagernde Volumen.*

Beweis. Durch (2b vii) werden nur Elemente verschoben, die sich überlagern. Tritt eine Überlagerung auf, wird ein Bewegungsvektor errechnet, der beide Elemente voneinander weg bewegt. Dieser Bewegungsvektor wird zur Summe von Bewegungsvektoren für jedes Element hinzu addiert. Die Elemente werden in (3) in Richtung

der Summe ihrer Bewegungsvektoren verschoben. An dicht besetzten Stellen bewegen sich dadurch äußere Ellipsoide langsam auseinander.

Wenn aufgrund der geometrischen Form der Elemente nicht alle überlagerungsfrei in das Gesamtvolumen passen, ist es nicht möglich, Überlagerungen weiter zu reduzieren. Jedoch bei einem nicht zu dicht besetzten Volumen bewegen sich überlagernde Elemente von einander weg. Dadurch verringert sich bei jedem Aufruf von `rearrange_content` tendenziell das überlagernde Volumen. □

Wenn sich flache Ellipsoide ungünstig überlagern, ist der Algorithmus `rearrange_content` nicht in der Lage, diese Überlagerungen zu trennen. Abb. 4.1 zeigt eine ungünstige Überlagerung.



(a) trennbar **(b)** ungünstig

Abbildung 4.1: Überlagerungen von Ellipsoiden

Laufzeit

Lemma 4.4. *Ein Aufruf von* `rearrange_content` *benötigt* $O(n^2)$ *Zeit.*

Beweis. In den Schleifen (2) und (2b) wird die Kombination von jedem Element mit jedem anderen Element genau einmal betrachtet. Daraus ergibt sich über die Gaußsche Summenformel eine obere Schranke von $O(n^2)$.

$$T(n) \approx \sum_{k=1}^{n-1} k = \frac{(n-1) \cdot n}{2} = O(n^2) \tag{4.8}$$

□

4.3 Füllen des Freiraumes

Bei der Herstellung der Elektrode wird das aktive Material mit dem inaktiven Material (Leitruß, Binder, Additive) und einem Füllmaterial vermischt. Das Gemisch wird auf die gewünschte Dicke der Folie gewalzt. Die Partikel des aktiven Materials sind härter und verformen sich beim Walzen nicht. Sie behalten die Kugel- bzw. Ellipsoidform bei. Das inaktive Material und das Füllmaterial verformt sich und fügt sich zwischen den Partikeln ein.

Nach dem Walzen wird das Füllmaterial entfernt. Sehr oft wird es ausgebrannt. Dadurch entsteht die poröse Struktur der Elektrode. Der durch diesen Herstellungsprozess erhaltene Freiraum füllt sich bei der assemblierten Zelle mit Elektrolyt.

Da sich das inaktive Material und der Freiraum plastisch verformt, ist es nicht möglich, die geometrische Form für statistische Auswertungen zu verwenden.

Es gibt jedoch Zusammenhänge, die bei der Rekonstruktion der 3-dimensionalen Geometrie hilfreich sind.

Jede Fläche A hängt von zwei, hier unbekannten Variablen x_1, x_2 ab. Die geometrische Form findet sich im Koeffizienten k_A. Zur Vereinfachung werden die unbekannten Variablen als gleich angenommen $x = x_1 = x_2$. Es ergibt sich die Fläche zu $A = k_A \cdot x^2$.

Jedes Volumen V hängt von drei, hier unbekannten Variablen x_1, x_2, x_3 ab. Die geometrische Form spiegelt sich in einem linearen Koeffizienten k_V wider. Die unbekannten Variablen werden zur Vereinfachung wieder als gleich angenommen $x = x_1 = x_2 = x_3$. Es ergibt sich das Volumen zu $V = k_V \cdot x^3$.

Werden diese beiden Zusammenhänge kombiniert, erhält man Gl. 4.9.

$$V = k_V \cdot \left(\sqrt{\frac{A}{k_A}} \right)^3 = k \cdot A^{\frac{3}{2}} \tag{4.9}$$

Die mittlere Größe der Flächen und die Standardabweichung wird aus dem segmentierten Bild (Abb. 2.4) bestimmt. Die Mengenverhältnisse stehen laut Lemma 1.1 ebenfalls fest.

Um die vorgegebenen Mengenverhältnisse zu erreichen, kann die Größe des Volumens mit dem Parameter k variiert werden, oder die Anzahl der einzufügenden Volumen kann angepasst werden.

Bei der Implementierung wurde $k = 1$ angenommen. Es werden über die Größenverteilung der Flächen einzufügende Volumen bestimmt. Die Volumen werden solange eingefügt, bis die vorgegebenen Mengenverhältnisse passen. Jedes Volumen wird als beliebiger, zusammenhängender Bereich eingefügt.

5 Computergrafik

In der Computergrafik werden Objekte im Raum üblicherweise durch Darstellung der Oberflächen angezeigt. Die Oberflächen der Objekte werden zumeist in Dreiecke aufgelöst. Die Dreiecke im Raum werden über eine Projektionsmatrix auf eine 2-dimensionale Abbildung reduziert und somit auf den Bildschirm gebracht [Redbook07, Kap. 3].

Die lokalen Koordinatensysteme der Dreiecke der Objekte werden über Model-View-Matrizen zum globalen Koordinatensystem übergeführt. Jedes Objekt wird somit in die globale Szenerie eingebettet. Die Objekte selbst können aus Teilobjekten zusammengesetzt sein. Jedes Teilobjekt hat wiederum eine eigene Model-View-Matrix im Objekt.

Die Model-View-Matrix ist eine Transformationsmatrix. Je nach Einträge erlaubt sie Translation, Rotation und Skalierung der Objekte.

Bei der gegebenen Problemstellung (Kap. 1.2) ist die innere geometrische Struktur des gesamten Volumens von Interesse. Die Oberflächen alleine sind nicht ausreichend, um das Innere des Volumens zu beschreiben. Aus diesem Grund wird das Volumen diskretisiert und in Matrizen abgespeichert. Vergleichbar mit der Diskretisierung von 2-dimensionalen Bildern in Pixel, werden 3-dimensionale Volumina in Voxel aufgelöst. Dieser Ansatz findet Verwendung, wenn es um die innere geometrische Struktur von Volumina geht. Dies ist der Fall in der Computertomographie, der Geologie, der Archäologie [Bezzi08], etc..

In diesem Kapitel wird erläutert, wie aus den analytischen Beschreibungen der 3-dimensionalen geometrischen Formen ein diskretisiertes Volumen generiert wird. Anschließend wird gezeigt, wie das generierte Volumen effizient genug auf den Bildschirm gebracht wird, um dieses interaktiv zu betrachten.

5.1 Flood-Fill

Im 2-dimensionalen Fall werden die Pixel im Inneren einer 2-dimensionalen Geometrie gezeichnet, indem die analytische Beschreibung der Geometrie an vorgegebenen Stellen ausgewertet wird. Die erhaltenen Punkte werden mit Linien verbunden. Die daraus resultierende Kontur wird anschließend mit einem Algorithmus, wie Flood-Fill oder Scanline-Fill, befüllt. Dies ist das Standardverfahren um beispielsweise gefüllte Ellipsen zu zeichnen.

Diese Methode basiert darauf, dass alle Pixel des Randes gesetzt sind. Die Methode auf diese Weise für den 3-dimensionalen Fall zu verwenden ist nicht trivial. Es muss jeder Voxel am Rand des Objektes gesetzt werden, bevor ein Algorithmus zum Füllen aufgerufen werden kann. Dies erfordert eine Beschreibung der Oberfläche des zu füllenden Objektes. Das Problem kann umgangen werden, indem beim Algorithmus des Füllens eine `is_inside`-Methode aufgerufen wird. Die Methode `is_inside` gibt wahr oder falsch zurück, je nachdem ob die gegebenen Koordinaten innerhalb oder außerhalb der Geometrie liegen. Die Funktionsweise der `is_inside`-Methode wird im folgenden Kapitel 5.2 erklärt.

Listing 5.1 zeigt einen modifizierten Flood-Fill-Algorithmus. Diese Implementierung benötigt aufgrund der `is_inside`-Methode in Zeile 17 keine gesetzten Pixel bzw. Voxel am Rand des zu füllenden Objektes. Anstatt eine Fläche mit Pixel zu füllen, wird hier durch die Modifikationen in den Zeilen 19–25 ein Volumen mit Voxel befüllt.

```
1  def fill(self):
2      """
3      Flood fill algorithm based on a queue
4      """
5      if (not self.is_inside(self.start_inside)):
6          raise Exception("start_inside is not inside volume.")
7      queue = deque()
8      queue.append(self.start_inside);
9      while True:
10         try:
11             current = queue.popleft()
12         except IndexError:
13             break
14         pval = self.get_voxel(current)
15         if pval != -1 and
16                 pval != self.value and
17                 self.is_inside(current):
18             self.set_voxel(current, self.value)
19             (x, y, z) = current
20             queue.append((x-1, y, z))
21             queue.append((x+1, y, z))
22             queue.append((x, y-1, z))
23             queue.append((x, y+1, z))
24             queue.append((x, y, z-1))
25             queue.append((x, y, z+1))
```

Listing 5.1: *Flood-Fill*

Anzumerken ist, dass der Flood-Fill-Algorithmus ein sehr einfach zu implementierender, jedoch recht ineffizienter Algorithmus ist. Die Ineffizienz liegt nicht in der Komplexität der Laufzeit, sondern in der Tatsache, dass nahezu alle Pixel bzw. Voxel mehrmals ausgelesen und überprüft werden. In der Implementierung von Listing 5.1 werden nahezu alle Voxel 6-mal ausgelesen und überprüft.

Ein effizienter Algorithmus, um Bereiche in einem 2-dimensionalen Bild zu befüllen, ist der Scanline-Fill-Algorithmus. Wie der Name bereits vermuten lässt, wird eine horizontale Linie, die Scanline, von y_{min} bis y_{max} über ein Polygon bewegt. Verfolgt man die Scanline von x_{min} bis x_{max}, ist jeder Schnittpunkt der Scanline mit den Kanten des zu füllenden Polygons ein Indikator für einen Übergang von außen nach innen oder innen nach außen. Befindet man sich beim Verfolgen der Scanline im Inneren des Polygons, werden die Pixel gesetzt. Der Vorteil dieses Algorithmus ist, dass jeder Pixel genau einmal betrachtet wird. Jedoch wurde der Scanline-Fill-Algorithmus hier nicht für die Verwendung im 3-dimensionalen Raum erweitert.

5.2 Inverse Model-View-Matrix

Wie in der Einleitung dieses Kapitels beschrieben, wird die Model-View-Matrix verwendet, um Objekte in die globale Szenerie einzufügen. Bei der häufig verwendeten Darstellung der Oberflächen beginnt die Darstellungsberechnung bei den Objekten selbst. Die darzustellenden Objekte werden mit den Model-View-Matrizen transformiert. Die Projektionsmatrix dient zur Abbildung der Objekte aus der Perspektive des Betrachters. Der z-Buffer wird verwendet, um Überdeckungen im Bild zu entfernen.

Wird die Berechnungsrichtung umgekehrt, d.h. nicht vom Objekt zum Bildschirm, sondern vom Bildschirm zum Objekt, muss die Model-View-Matrix invertiert werden. Hier findet dies Anwendung bei der `is_inside`-Methode. Die gegebenen Koordinaten im Raum werden mit der inversen Model-View-Matrix M^{-1} auf Objekt-Koordinaten transformiert. Im Falle von Ellipsoiden ist das Objekt immer eine Einheitskugel. Die Einheitskugel wird mit der inversen Model-View-Matrix skaliert, rotiert und positioniert. Somit kann jeder Ellipsoid in allgemeiner Lage auf eine Einheitskugel in Hauptlage zurückgeführt werden. Die Auswertung, ob sich die transformierten Koordinaten innerhalb oder außerhalb der Einheitskugel liegen ist trivial. Ein Aufruf der `is_inside`-Methode reduziert sich auf eine Matrizenmultiplikation und eine Berechnung der euklidischen Distanz. Bei der Distanzberechnung kann sogar auf die Berechnung der Wurzel verzichtet werden.

Sei *coord_global* ein 3-dimensionaler Vektor mit Koordinaten im Raum. Die Funktion `is_inside`(*coord_global*) bestimmt, ob sich die gegebenen Koordinaten innerhalb oder außerhalb eines Ellipsoids in allgemeiner Lage befinden.

`is_inside`(*coord_global*)

1. $$(x, y, z)^T = coord\_global$$
2. $$coord\_obj = M^{-1} \cdot (x, y, z, 1)^T$$
3. $$\texttt{return} \left(\sum_{i=1}^{3} coord\_obj_i^2 \right) \leq 1$$

Algorithmus 5.1: Bestimmung, ob ein gegebener Punkt innen oder außen liegt

Die invertierte Model-View-Matrix ist eine (4×4)-Matrix, die in Skalierung S^{-1}, Rotation R^{-1} und Translation T^{-1} zerlegt werden kann. Beim Beispiel in Gl. 5.1 wird das Objekt zuerst skaliert, anschließend rotiert und abschließend an die gewünschte Position verschoben. Die Reihenfolge der Einzeloperationen spielt eine bedeutende Rolle.

$$M^{-1} = S^{-1} \cdot R^{-1} \cdot T^{-1} \tag{5.1}$$

Der Aufbau der einzelnen Matrizen ist aus [Redbook07, Appendix G] entnommen.

Die Inverse der Translationsmatrix T ergibt sich, indem man die Richtung $t = (t_x, t_y, t_z)$, in die das Objekt verschoben wird, umkehrt. Die inverse Translationsmatrix T^{-1} ist in Gl. 5.2 dargestellt.

$$T^{-1} := \begin{pmatrix} 1 & 0 & 0 & -t_x \\ 0 & 1 & 0 & -t_y \\ 0 & 0 & 1 & -t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{5.2}$$

Die Inverse der Skalierungsmatrix S ergibt sich, durch Invertierung der Skalierungsfaktoren $s = (s_x, s_y, s_z)$. Die inverse Skalierungsmatrix S^{-1} ist in Gl. 5.3 zu sehen.

$$S^{-1} := \begin{pmatrix} \frac{1}{s_x} & 0 & 0 & 0 \\ 0 & \frac{1}{s_y} & 0 & 0 \\ 0 & 0 & \frac{1}{s_z} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{5.3}$$

Die Rotationsmatrix R rotiert ein Objekt um eine Ursprungsgerade in Richtung des Vektors $n = (n_x, n_y, n_z)$ um den Winkel α. Hier ist zu beachten, dass der Vektor n normiert $\|n\| = 1$ vorliegen muss. Die Rotationsmatrix R wird invertiert, indem der Winkel α umgedreht wird. Die inverse Rotationsmatrix R^{-1} ist in Gl. 5.5 dargestellt.

$$s := \sin(-\alpha) \tag{5.4a}$$

$$c := \cos(-\alpha) \tag{5.4b}$$

$$R^{-1} := \begin{pmatrix} n_x^2(1-c)+c & n_x n_y(1-c)-n_z s & n_x n_z(1-c)+n_y s & 0 \\ n_x n_y(1-c)+n_z s & n_y^2(1-c)+c & n_y n_z(1-c)-n_x s & 0 \\ n_x n_z(1-c)-n_y s & n_y n_z(1-c)+n_x s & n_z^2(1-c)+c & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{5.5}$$

5.3 Volumenvisualisierung

Bei der Volumenvisualisierung wird das innere Volumen von 3-dimensionalen Objekten mit vorgegebenen Transparenzwerten zur Anzeige gebracht. Hierzu wird das Volumen in kleine Volumenelemente, sogenannte Voxel, diskretisiert.

5.3.1 Raycasting

Raycasting dient zur Volumenvisualisierung. Es ist ein einfaches Raytracing-Verfahren. Ausgehend vom Auge des Betrachters wird ein Strahl durch jeden Pixel des Bildschirms in das zu rendernde Volumen verfolgt. Der Farbwert des Pixels am Bildschirm ergibt sich aus den Farb- und Transparenzwerten der getroffenen Voxel im Volumen. Beim Raycasting werden keine Reflexionen berücksichtigt.

Derzeitige Implementierungen vom Raycasting-Verfahren verwenden oft keine Hardwarebeschleunigung oder die Hardwarebeschleunigung ist vom Grafikkartenhersteller abhängig.

5.3.2 Texture Mapping

Die Open Graphics Library (OpenGL) [Redbook07] ist ausgelegt, um Oberflächen von 3-dimensionalen Objekten hardwarebeschleunigt am Bildschirm anzuzeigen. Seit Version 1.2 (1998) bietet die Spezifikation von OpenGL die Möglichkeit, 3-dimensionale Texturen, auch Volume Textures genannt, zu verwenden. Mittels Texture Mapping wird die Textur auf die Oberfläche des darzustellenden Objektes gebracht. Texture Mapping ist eine Abbildung der Koordinaten der Oberfläche des Objektes auf die Koordinaten der Textur. Die Textur selbst kann Transparenzwerte enthalten.

Ein Volumen kann mit OpenGL durch Texture Mapping einer 3-dimensionalen Textur auf parallele, äquidistante Ebenen im Raum dargestellt werden. Durch Transparenzwerte der Textur werden dahinterliegende Ebenen nur teilweise überdeckt. Um das Volumen korrekt anzuzeigen, ist es notwendig, dass die Ebenen parallel zum Bildschirm ausgerichtet sind.

Eine existierende, hardwarebeschleunigte Implementierung dieses Verfahrens wird in [VTKGuide10] beschrieben. Auf aktuellen Rechnern ist es problemlos möglich, ein Volumen mit $200 \times 200 \times 200 = 8 \cdot 10^6$ Voxel interaktiv darzustellen. Volumina mit einer größeren Auflösung werden mittels Downsampling auf eine problemlos darzustellende Größe gebracht.

Abb. 5.1 zeigt das Ergebnis der Volumenvisualisierung mittels Texture Mapping. Das inaktive Material ist blau und transparent zu sehen. Die Ellipsoide des aktiven Materials sind rot dargestellt.

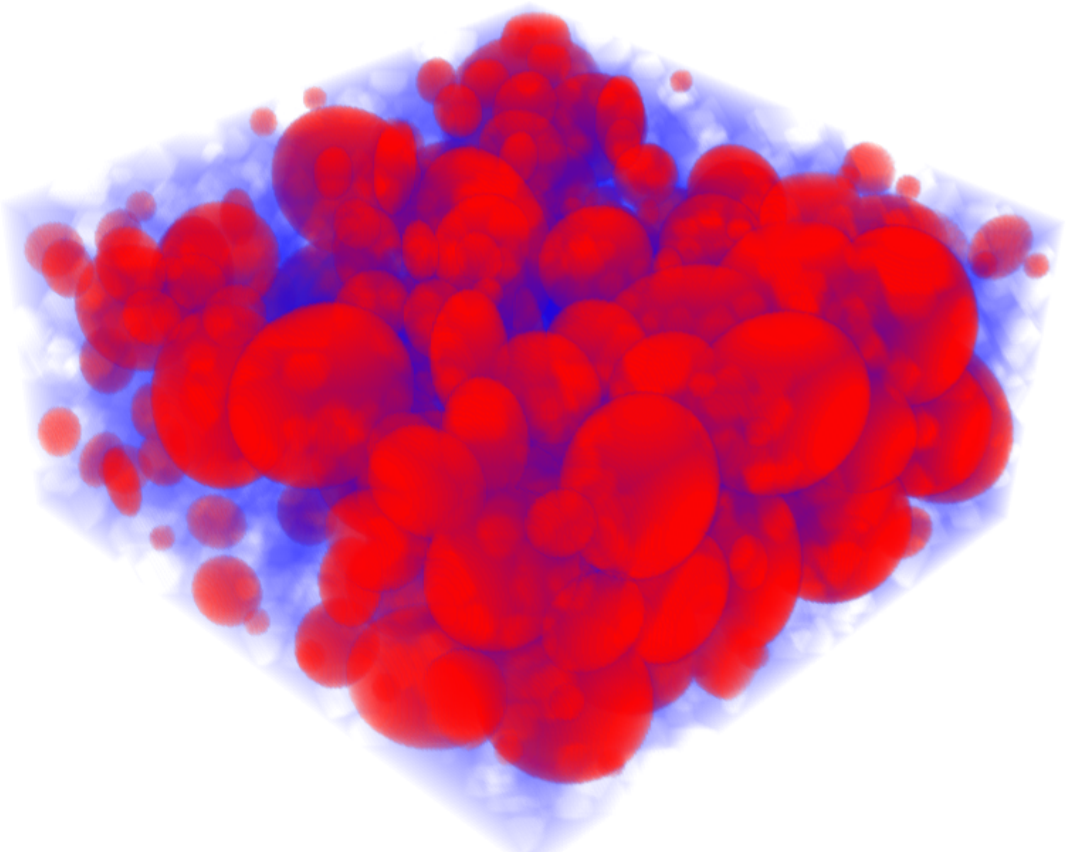


Abbildung 5.1: Ergebnis der Volumenvisualisierung

5.3.3 Stand der Technik

Der aktuelle Stand der Technik erlaubt es, eine große Menge an Voxel mit effizienten Datenstrukturen und Algorithmen ohne spezieller Hardware in Echtzeit darzustellen. Hierbei werden die Voxel in Octrees gespeichert und auf effiziente Weise auf den Bildschirm gebracht. Der Algorithmus ist in [Laine10] erläutert.

6 Indirekte Bestimmung der Parameter

Die generierte 3-dimensionale Geometrie wird verwendet, um Parameter für das mathematische Modell der Elektrochemie einer Lithium-Ionen-Zelle [Doyle93] zu finden. Um die Parameter wie die Austauschfläche oder den effektive Diffusionskoeffizient bestimmen zu können, wird eine 3-dimensionale Beschreibung der Geometrie benötigt.

6.1 Porosität

Die Porosität ε (manchmal Φ) beschreibt das Mengenverhältnis von freiem Raum bezogen auf den gesamten Raum. Im 3-dimensionalen Fall ist dies das Volumenverhältnis von der Summe an freien Volumina $V_{Freiraum}$ bezogen auf das Gesamtvolumen V_{Gesamt}. Im 2-dimensionalen Fall ist das Mengenverhältnis ein Verhältnis von Flächen. Bei nur einer Dimension ist es ein Verhältnis von Längen. Gl. 6.1 zeigt die Berechnung der Porosität für Volumina.

$$\varepsilon = \frac{V_{Freiraum}}{V_{Gesamt}} \tag{6.1}$$

Lemma 1.1 beweist, dass die Mengenverhältnisse eines statistisch repräsentativen 1-dimensionalen Schnittes den Mengenverhältnissen des 2-dimensionalen Ursprungsbildes entsprechen. Der Beweis ist analog auf höhere Dimensionen anwendbar.

Lemma 6.1. *Die Mengenverhältnisse eines statistisch repräsentativen 2-dimensionalen Querschnittsbildes entsprechen den Mengenverhältnissen des 3-dimensionalen Ursprungsvolumens.*

Beweis. Durch äquidistante Diskretisierung des Volumens wird die Annahme nachvollziehbar. Man betrachte jede Ebene in die Tiefe des Volumens separat. Diese Ebenen werden an den Seiten zu einer großen Ebene bzw. einer großen Fläche zusammengefügt. Jedes Mengenverhältnis dieser konstruierten Ebene ist äquivalent zu den Mengenverhältnissen im Ursprungsvolumen. Die Mengenverhältnisse eines statistisch repräsentativer Teil der konstruierten Ebene entsprechen den Mengenverhältnissen des 3-dimensionalen Ursprungsvolumens. □

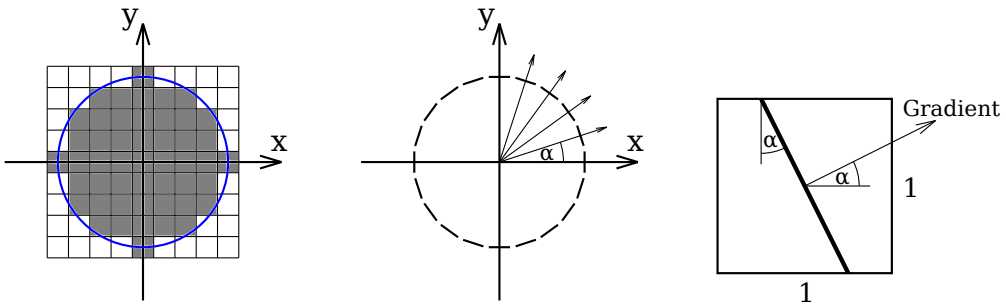
6.2 Austauschfläche

Die Austauschfläche ist jene Oberfläche des aktiven Materials, die mit dem Elektrolyt in Verbindung steht. Die Fläche zwischen dem aktiven Material und dem inaktiven Material ist nicht Teil der Austauschfläche.

Um die Austauschfläche zu bestimmen, ist eine Beschreibung der Oberfläche erforderlich. Im 3-dimensionalen Fall kann die Oberfläche durch Triangulierung der Oberfläche des Volumens angenähert werden. Einfacher ist es, eine Annäherung der Austauschfläche über eine Gewichtung jedes Kanten-Voxels anhand des jeweiligen Gradienten zu berechnen. Im Folgenden wird ein Lösungsansatz präsentiert, der das Problem um eine Dimension reduziert betrachtet.

6.2.1 Lösungsansatz

Bei einem äquidistant diskretisierten Kreis soll der Umfang bestimmt werden. Zur Bestimmung des Umfangs steht nur die diskretisierte Darstellung des Kreises zur Verfügung. In diesem Beispiel wird die diskrete Darstellung eines Kreises mit Radius $r = 4$ nach der im Kap. 5 beschriebenen Methode generiert. Abb. 6.1a zeigt die gesetzten Pixel im Inneren des Kreises.



(a) Diskretisierter Kreis mit $r = 4$ **(b)** Gradienten der Kanten-Pixel **(c)** Gewichtung eines Kanten-Pixels

Abbildung 6.1: Lösungsansatz zur Annäherung des Umfangs

Der exakte Umfang U eines Kreises mit Radius $r = 4$ errechnet sich nach Gl. 6.2 zu $25,1327$.

$$U = 2\pi \cdot r = 2\pi \cdot 4 = 25,1327 \tag{6.2}$$

Bei der Annäherung wird jeder Kanten-Pixel anhand seiner Ausrichtung gewichtet. Das Ergebnis der Annäherung ist in Gl. 6.3 gezeigt.

$$U \simeq 4 \cdot \left(1 + \frac{1}{\cos(18°)} + \frac{1}{\cos(36°)} + \frac{1}{\sin(54°)} + \frac{1}{\sin(72°)}\right) = 22,3 \tag{6.3}$$

Analog zur Annäherung des Umfangs durch eine Gewichtung der Kanten-Pixel kann die Oberfläche durch eine Gewichtung der Voxel des Randes angenähert werden. Zur Gewichtung werden die Gradienten an den Voxeln des Randes bestimmt.

6.3 Effektiver Diffusionskoeffizient

Der Diffusionskoeffizient wird von den Fick'schen Gesetzen verwendet, um die Ausbreitungsgeschwindigkeit von zufällig bewegten Teilchen in einem Medium zu beschreiben. Die zufällige Bewegung der Teilchen ist thermisch bedingt. In einer Zelle handelt es sich bei den Teilchen um ladungstragende Lithium-Ionen, die sich im Elektrolyt zufällig bewegen.

Das 2. Fick'sche Gesetz ist die Diffusionsgleichung. Sie ist in Gl. 6.4 zu sehen. Der Diffusionskoeffizienten D ist in $\frac{\mathrm{m}^2}{\mathrm{s}}$ gegeben. Die Konzentration c beschreibt, wie viele Teilchen in einem Volumen zu finden sind. Sie ist in $\frac{\mathrm{mol}}{\mathrm{m}^3}$ gegeben.

$$\frac{\partial c}{\partial t} = \nabla \cdot (D \nabla c) \tag{6.4}$$

Der Diffusionskoeffizient der Lithium-Ionen im Elektrolyt kann experimentell bestimmt werden. Dieser verändert sich jedoch, wenn der Elektrolyt die poröse Elektrode durchtränkt. Hier spricht man vom effektiven Diffusionskoeffizient poröser Medien. Die Diffusion findet nur in den Poren der Elektrode statt. Dies verlängert die Diffusionswege und hat somit Auswirkungen auf den Diffusionskoeffizienten.

Nach [Doyle93] und [Doyle95, Gl. 2-25] wird der längere Diffusionsweg mit der Bruggeman-Korrektur berücksichtigt. Die Bestimmung des effektiven Diffusionskoeffizienten D_{eff} ist in Gl. 6.5 dargestellt. Er hängt von der Porosität ε ab. Die Tortuosität („Gewundenheit“) ist im Bruggeman-Exponenten $brugg$ enthalten.

$$D_{eff} = D \cdot \varepsilon^{brugg} \tag{6.5}$$

Durch die Rekonstruktion der Geometrie kann die Verlängerung des Diffusionsweges bestimmt werden. Abb. 6.2 zeigt dies schematisch.

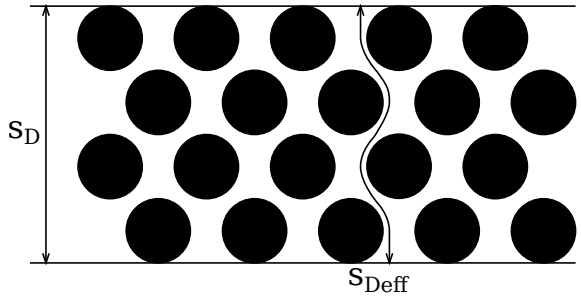


Abbildung 6.2: Verlängerung des Diffusionsweges

Der Diffusionskoeffizient wird durch die Verlängerung des Diffusionsweges linear skaliert. Somit kann der effektive Diffusionskoeffizient über die Geometrie bestimmt werden.

Weiters kann dadurch der Bruggeman-Exponent *brugg* bestimmt werden. Gl. 6.6 zeigt die Zusammenhänge.

$$\varepsilon^{brugg} = \frac{s_D}{s_{D_{eff}}} \tag{6.6}$$

Durch den Bruggeman-Exponenten kann neben dem effektiven Diffusionskoeffizienten D_{eff} die effektive ionische Leitfähigkeit des Elektrolyts $\kappa_{eff}\left[\frac{\mathrm{S}}{\mathrm{m}}\right]$ und die effektive elektrische Leitfähigkeit $\sigma_{eff}\left[\frac{\mathrm{S}}{\mathrm{m}}\right]$ bestimmt werden.

7 Softwareentwicklung

In diesem Kapitel wird ein Überblick über die Softwarearchitektur der Implementierung, über die Implementierung selbst und über die verwendeten Softwarebibliotheken gegeben. Die Softwarearchitektur baut auf Konzepten der Objekt-orientierten Programmierung und verwendet Methoden des Extreme Programming (XP) [Beck04] um die Qualität zu sichern.

7.1 Softwarearchitektur

Die einzelnen Module sind in Pakete (engl.: Packages) organisiert. Ein Modul ist eine Quelltext-Datei. Ein Paket wird durch einen Ordner realisiert. Tabelle 7.1 zeigt, wo die Implementierungen der einzelnen Kapiteln dieser Arbeit zu finden sind.

Package	**Thema**	**Kap.**
`statgeo.vis`	Bildverarbeitung	2
`statgeo.analysis.stat`	Stochastik	3
`statgeo.geo.threed`	Geometriegenerierung	4
`statgeo.geo.threed` und `.ui.vtk`	Computergrafik	5
`statgeo.analysis.geo`	Parameterbestimmung	6
`statgeo.test`	Qualitätssicherung	7.4
`statgeo.ui.tk`	Benutzeroberfläche	7.5

Tabelle 7.1: Organisation des Quelltextes

Ein signifikanter Teil der Objekt-orientierten Struktur ist im Klassendiagramm in Abb. 7.1 gezeigt.

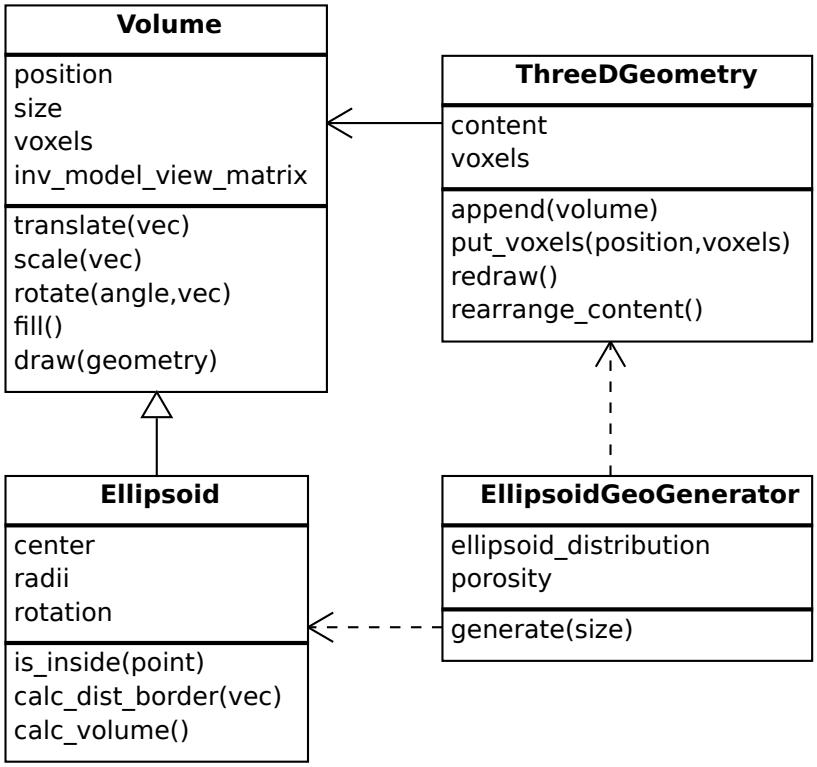


Abbildung 7.1: UML-Klassendiagramm von `statgeo.geo.threed`

Eine Geometrie (`ThreeDGeometry`) hat eine Menge von Volumina. Eine konkrete Realisierung eines Volumens (`Volume`) ist ein Ellipsoid (`Ellipsoid`). Die Realisierung der Klasse `Volume` muss nur eine Implementierung der Methode `is_inside(point)` zur Verfügung stellen. Das Füllen der Voxel im Inneren des Volumens übernimmt die Klasse `Volume` selbst. Ebenso sind Methoden, die die inverse Model-View-Matrix anpassen, in der Klasse `Volume` implementiert. Die Klasse `EllipsoidGeoGenerator` ist zuständig für das Generieren von Ellipsoiden in einer `ThreeDGeometry`. Durch die Methode `rearrange_content()` werden Überlagerungen der Ellipsoiden reduziert.

7.2 Implementierung in Python

Für die Implementierung wurden mehrere Programmiersprachen evaluiert. Es wurde Matlab, Python, Java und C/C++ für eine mögliche Implementierung herangezogen. Kriterien für die Auswahl waren der Implementierungsaufwand, die Ausführungsgeschwindigkeit und die zur Verfügung stehenden Software-Bibliotheken.

Da es sich bei Matlab und Python um interpretierende Sprachen handelt, wurde die Ausführungsgeschwindigkeit untersucht. Hierzu wurde eine simple Matrizenmultiplikation mit 3 ineinander verschachtelten Schleifen implementiert. Tabelle 7.2 zeigt die benötigte Ausführungszeit bei der Multiplikation zweier Matrizen der Größe 255×255. Die Elemente der Matrizen wurden mit zufälligen Werten initialisiert. Als CPU war ein Intel Core i5 M540 im Einsatz.

Sprache	**Zeit [sec]**
Python & NumPy	0,031
Python 2.6.4 (mit Schleifen)	7,337
Matlab 2010a	0,020
Matlab 2010a (mit Schleifen)	33,925
Java 6u20	0,076
GCC 4.4.4 (-O2)	0,040

Tabelle 7.2: Geschwindigkeitsvergleich durch Matrizen-Multiplikation

Deutlich zu erkennen ist der große Geschwindigkeitsunterschied in Matlab zwischen der mit Schleifen implementierten Matrizenmultiplikation und des in der Sprache enthaltenen Multiplikationsoperators.

Des Weiteren ist der Vergleich zwischen C (GCC) und Matlab oder Python & NumPy zu beachten. Sei n die Anzahl der Zeilen und Spalten einer quadratischen Matrix. Eine simple Matrizenmultiplikation mit 3 ineinander verschachtelten Schleifen hat eine Laufzeit von $O(n^3)$. Eine Verbesserung für praktische Anwendungen ist der Algorithmus nach Volker Strassen [Strassen69]. Strassen's Algorithmus hat eine asymptotische Laufzeit von $O(n^{2,8})$. Der derzeit schnellste, bekannte Algorithmus zur Multiplikation von quadratischen Matrizen ist der Algorithmus nach Coppersmith-Winograd [Coppersmith90]. Dieser hat eine asymptotische Laufzeit von $O(n^{2,38})$.

Interpretierende Sprachen sind aufgrund des hohen Abstraktionslevels einfach zu bedienen. Der Implementierungsaufwand ist dadurch geringer. Ein Nachteil ist jedoch, dass nicht alle möglichen Algorithmen als eingebaute Funktionen zur Verfügung stehen können. Eine Implementierung rechenintensiver Algorithmen in interpretierenden Sprachen führt zu einer erhöhten Ausführungszeit.

Eine Erweiterung des Interpreters selbst vereint die Vorteile der Ausführungsgeschwindigkeit von kompilierten Sprachen (C/C++) und den geringen Implementierungsaufwand bei interpretierenden Sprachen.

Durch eine Erweiterung des Interpreters erweitern sich gleichzeitig die zur Verfügung stehenden Software-Bibliotheken. Python hat durch NumPy[1] eine umfassende Software-Bibliothek für Lineare Algebra. Die meisten Software-Bibliotheken der Bildverarbeitung sind in C/C++ implementiert.

7.2.1 Python-Interpreter erweitern

Listing 7.1 zeigt ein Minimalbeispiel um eine in C implementierte Funktion in Python aufzurufen. Durch diese Implementierung in C steht in Python ein Modul namens `anything` mit der Funktion `do_anything` zur Verfügung.

```
1  #include "Python.h"
2  int _do_anything(int n)
3  { /* ... */ }
4  static PyObject* do_anything(PyObject* self, PyObject* args)
5  {
6    int n;
7    if (!PyArg_ParseTuple(args, "i", &n))
8      return NULL;
9    return Py_BuildValue("i", _do_anything(n));
10 }
11 static PyMethodDef MyMethods[] = {
12   {"do_anything", do_anything, METH_VARARGS, "doc string"},
13   {NULL, NULL, 0, NULL}
14 };
15 PyMODINIT_FUNC initanything(void)
16 {
17   (void) Py_InitModule("anything", MyMethods);
18 }
```

Listing 7.1: Python-Interpreter erweitern

In den Zeilen 4–10 ist der eigentliche Wrapper implementiert. Hier wird in Zeile 7 ein Pyhton-Integer auf einen C-Integer umgewandelt. Die in C implementierte Funktion nimmt diesen Wert als Eingangsparameter. Der Rückgabewert ist laut Zeile 2 ebenfalls ein C-Integer, der in Zeile 9 zu einem Python-Integer konvertiert wird. Die Zeilen 11–14 definiert die Funktionen des Python-Moduls, die in den Zeilen 15–18 initialisiert werden müssen.

[1] `http://numpy.scipy.org/`

Der Datentyp einer Matrix wird von Python selbst nicht definiert. Erst durch NumPy wird eine Matrix und diverse Operationen der Linearen Algebra in Python definiert.

Um eine NumPy-Matrix in C beschreiben zu können, muss das jeweilige Header-File eingebunden und die Funktion `import_array()` bei der Initialisierung aufgerufen werden. Listing 7.2 zeigt die notwendigen Modifikationen im Vergleich zum Minimalbeispiel aus Listing 7.1.

```
1 #include "Python.h"
2 #include "numpy/arrayobject.h"
3 /* ... */
4 PyMODINIT_FUNC initnumpymod(void) {
5   (void) Py_InitModule("numpymod", MyMethods);
6   import_array();
7 }
```

Listing 7.2: NumPy erweitern

7.3 Verwendete Bibliotheken

Verschiedenste Software-Bibliotheken von den Bereichen der Linearen Algebra, der Bildverarbeitung und Computergrafik, etc. sind im Einsatz. In diesem Kapitel wird auf die Software-Bibliotheken eingegangen, die mit speziellen Konfigurationsoptionen kompiliert werden.

Insight Toolkit

Das Insight Segmentation and Registration Toolkit (ITK)[2] ist eine umfassende Software-Bibliothek, die Algorithmen der Bildverarbeitung, im speziellen der Segmentierung, zur Verfügung stellt. Diese Bibliothek ist in C++ implementiert. Es existieren bereits implementierte Wrapper für Python, Java und Tcl. Diese Wrapper sind jedoch als experimentell gekennzeichnet und müssen deshalb mit speziellen Konfigurationsoptionen gebaut werden.

Zu Beginn müssen alle Dateien von `CableSwig-3.20.0` nach `InsightToolkit-3.20.0/Utilities/CableSwig` kopiert werden.
Die verwendeten Konfigurationsoptionen sind in Listing 7.3 dargestellt.

```
1  cd InsightToolkit-3.20.0
2  cmake -D BUILD_SHARED_LIBS="ON" \
3        -D ITK_CSWIG_JAVA="OFF" \
4        -D ITK_CSWIG_PYTHON="OFF" \
5        -D ITK_CSWIG_TCL="OFF" \
6        -D ITK_USE_REVIEW="ON" \
7        -D USE_WRAP_ITK="ON" \
8        -D WRAP_ITK_JAVA="OFF" \
9        -D WRAP_ITK_PYTHON="ON" \
10       -D WRAP_ITK_TCL="OFF" .
```

Listing 7.3: InsightToolkit Flags

Der Python-Wrapper von WrapITK erlaubt den Zugriff auf die Funktionalität des Insight-Toolkits, jedoch fehlt ein effizienter Zugriff auf die Bildinformationen im Hauptspeicher. Um die Bilder direkt in eine NumPy-Matrix konvertieren zu können, muss ein externes Projekt namens PyBuffer kompiliert werden. Es befindet sich im Quelltext-Archiv vom Insight Toolkit. Bevor dieses Projekt kompiliert werden kann, muss NumPy und WrapITK bereits im Binärformat (kompiliert) vorliegen. Listing 7.4 zeigt die Konfiguration der Informationen zum Bauen von PyBuffer.

```
1  cd Wrapping/WrapITK/ExternalProjects/PyBuffer
2  cmake -D PYTHON_NUMARRAY_INCLUDE_DIR="${NUMPY_INCLUDE_DIR}" \
3        -D WrapITK_DIR="${WRAP_ITK_DIR}" .
```

Listing 7.4: PyBuffer Flags

[2] `http://www.itk.org/`

VTK

Das Visualization Toolkit (VTK)[3] wird zur Visualisierung der erstellten Geometrie verwendet. Es ist in C++ implementiert und bietet offizielle Wrapper für Python, Java und Tcl. Das Bauen der Wrapper muss bei der Konfiguration aktiviert werden. Listing 7.5 zeigt, wie das Bauen des Wrappers für Python aktiviert wird.

```
1 cd vtk-5.6.1
2 cmake -D BUILD_SHARED_LIBS="ON" \
3       -D VTK_WRAP_JAVA="OFF" \
4       -D VTK_WRAP_PYTHON="ON" \
5       -D VTK_WRAP_TCL="OFF" .
```

Listing 7.5: VTK Flags

7.4 Qualitätssicherung

Unit Tests sind eine bewährte Methode, um die Qualität von Software sicherzustellen. Es handelt sich um kleine Programme, die die implementierten Funktionen automatisch testen. Beim Testen wird überprüft, ob eine implementierte Funktion bei definierten Eingabeparametern einen definierten Rückgabewert liefern.

Unit Tests helfen bei der Entwicklung. Es ist sinnvoller, einen Unit Test zu einer Funktion zu schreiben, anstatt die Funktion im Interpreter aufzurufen, da alle Unit Tests zu einem späteren Zeitpunkt vollautomatisch überprüft werden können. Negative Auswirkungen von späteren Änderungen werden somit automatisch erkannt.

Neben der Qualitätssicherung dokumentieren Unit Tests die Verwendung der implementierten Funktionen.

[3] http://www.vtk.org/

7.5 GUI

Das Programm startet mit einer grafischen Benutzeroberfläche (engl.: graphical user interface). Nach dem Start ist das Hauptfenster von Abb. 7.2 am Bildschirm zu sehen. Es zeigt eine Auswahl an Funktionen.

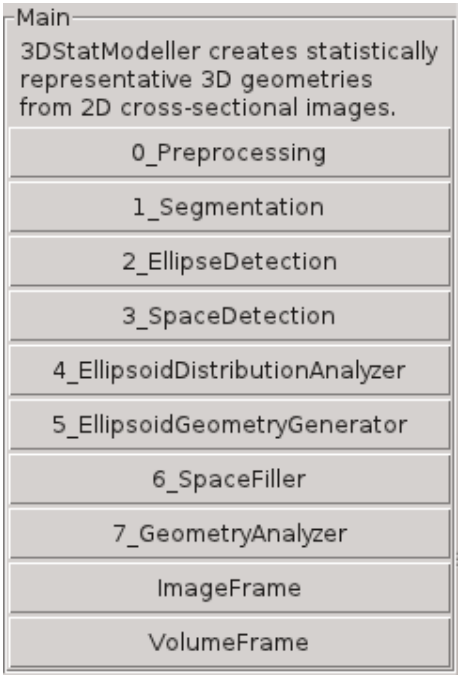


Abbildung 7.2: Hauptfenster

Nach der Auswahl einer Funktion, wird der jeweilige Parameter-Dialog geöffnet. Dieser beinhaltet eine Beschreibung der Funktion und Eingabefelder der benötigten Parameter.

Der Parameter-Dialog wird zur Laufzeit vollautomatisch aus dem Quelltext der gegebenen Funktion generiert. Listing 7.6 zeigt ein Beispiel einer gegebenen Funktion.

```
1 def do_anything(s_first, f_second, i_third, b_fourth, fileopen_fifth):
2     """
3     This is a simple test.
4     """
5     # ...
```

Listing 7.6: Funktionsdefinition

Die Namen der Eingabeparameter legen das zu erstellende Eingabefeld und die Validierung der Eingabe fest. Abb 7.3 zeigt den aus Listing 7.6 generierten Parameter-Dialog.

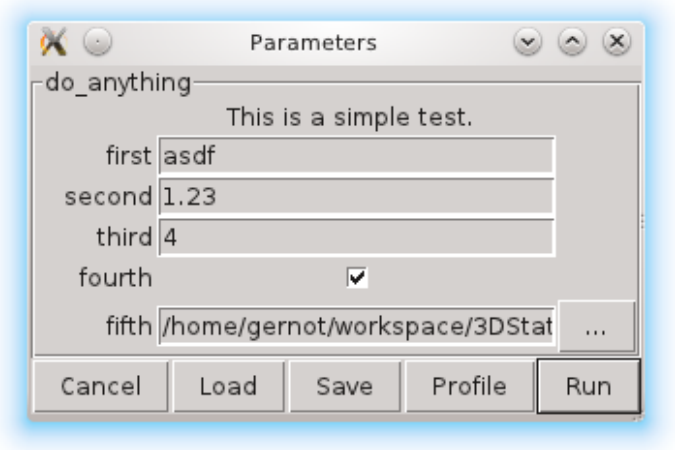


Abbildung 7.3: Parameter-Dialog

Der Parameter-Dialog bietet folgenden Funktionsumfang:

- Anzeigen der `pydoc`-Dokumentation
- Validierung der Eingabe
- Konfigurierbare Standardwerte
- Konfigurationsdateien laden/speichern
- Aufrufen des Profilers
- Verlauf der aufgerufenen Funktionen

Durch Präfixe der Namen der Eingabeparameter werden die gewünschten Eingabefelder im Parameter-Dialog erzeugt. Alle unterstützten Präfixe sind in Tab. 7.3 aufgelistet.

Präfix	**Type**
`b_`	bool
`i_`	int
`f_`	float
`s_`	str
`l_`	list
`d_`	dict
`o_`	object
`fileopen_`	str (Datei-Öffnen-Dialog)
`filesave_`	str (Datei-Speichern-Dialog)

Tabelle 7.3: Präfixe von Variablennamen

8 Zusammenfassung und Ausblick

Zu Beginn dieser Arbeit wird darauf hingewiesen, dass viele Eigenschaften einer Lithium-Ionen-Zelle von der Geometrie der Elektroden-Mikrostruktur abhängen. Ein Lösungsansatz zur statistischen Rekonstruktion der Geometrie aus Querschnittsinformationen wird präsentiert. Im Lösungsansatz werden Geometrien aus Kreisen mit Linien geschnitten. Mittels statistischer Analyse der Informationen der Schnittlinien werden wahrscheinliche Kreisradien und zugehörige Häufigkeiten bestimmt. Eine randomisierte Rekonstruktion der Geometrie mit den zuvor bestimmten Kreisen zeigt eine zur ursprünglichen Geometrie statistisch vergleichbare Rekonstruktion. Je nach geometrischer Form der geschnittenen Geometrien und der statistischen Repräsentativität des Querschnitts sind bei kleineren Querschnitten gute Rekonstruktionen möglich.

In dieser Arbeit wird eine Methode präsentiert, die aus statistisch repräsentativen 2-dimensionalen Querschnittsbildern einer 3-dimensionalen Geometrie eine statistisch vergleichbare 3-dimensionale Geometrie rekonstruiert. Je öfter die vorkommenden geometrischen Formen im Querschnittsbild geschnitten sind, desto statistisch repräsentativer ist das Querschnittsbild und desto besser ist die daraus resultierende Rekonstruktion. Aufbauend auf den im 2-dimensionalen Querschnittsbild vorhandenen 2-dimensionalen geometrischen Formen wird eine statistische Analyse verwendet, um geschnittene 3-dimensionale geometrische Formen zu erkennen. Bei der gegebenen Problemstellung handelt es sich um Ellipsoide, die im Querschnittsbild als Ellipsen zu sehen sind.

Das Querschnittsbild ist eine Aufnahme von einem Elektronenmikroskop. Um im Querschnittsbild Ellipsen zu erkennen, müssen die im Querschnittsbild vorhandenen Materialien segmentiert werden. Der in dieser Arbeit eingesetzte Algorithmus zur Segmentierung und Klassifizierung hat relative Abweichungen der Mengenverhältnisse von ca. 5 %. Da die verwendete Ellipsenerkennung auf der Methode der kleinsten Fehlerquadrate aufbaut, funktioniert sie ebenfalls bei starken Abweichungen von ellipsenähnlichen Formen.

Für die Rekonstruktion der Geometrie werden die bestimmten 3-dimensionalen geometrischen Formen randomisiert in ein Volumen eingefügt. Es ist zu beachten, dass Überlagerungen nicht erwünscht sind. Der hier beschrieben Algorithmus verschiebt die geometrischen Formen auf eine Weise, dass möglichst wenig Überlagerungen auftreten. Hierbei werden gute Ergebnisse erzielt, wenn alle Ellipsoide gleich ausgerichtet (rotiert) oder möglichst kugelförmig sind.

Die rekonstruierte 3-dimensionale Geometrie wird in einem äquidistant diskretisierten Volumen gespeichert. Die verwendete Volumenvisualisierung ist effizient genug, um das gesamte Volumen interaktiv am Bildschirm auszugeben.

In dieser Arbeit werden mehrere Methoden zur indirekten Bestimmung der Parameter des elektrochemischen Modells von Lithium-Ionen-Zellen vorgestellt. Die Methoden zeigen Einsatzgebiete der rekonstruierten Mikrostruktur von Lithium-Ionen-Zellen und sind hier nicht evaluiert.

Die Implementierung verwendet das Paradigma der Objekt-orientierten Programmierung (OOP). Die Qualität der Implementierung wird mit Methoden des Extreme Programmings (XP) [Beck04] sichergestellt. Bedienbar wird die Implementierung durch eine automatisch generierte graphische Benutzeroberfläche (GUI). Dadurch steht eine umfangreiche graphische Benutzeroberfläche mit geringem Programmieraufwand zur Verfügung.

8.1 Weiterführende Arbeit

Aufbauend auf dem derzeitigen Stand dieser Arbeit ergibt sich eine Vielzahl an Einsatzzwecke der gewonnenen Ergebnisse. Verbesserungsmöglichkeiten sind im Folgenden ebenfalls angeführt.

- Export der rekonstruierten Geometrie zu anderen Formaten

  OpenFOAM[1] ist eine „Computational Fluid Dynamics (CFD)“-Software. Neben der Berechnung von Strömungen ermöglicht eine CFD-Software Wärmeausbreitungen zu berechnen. Thermische Vorgänge in der Zelle können dadurch modelliert werden.

- Oberflächenunebenheiten modellieren

  Bei der aktuellen Rekonstruktion des aktiven Materials handelt es sich um perfekte Ellipsoide. Dies entspricht nicht der Realität. Die daraus bestimmten Parameter, speziell die Austauschfläche, hat dadurch Abweichungen.

- Ideen zur indirekten Bestimmung der Parameter evaluieren

- Bessere Ellipsenerkennung

  Grenzen die ellipsenähnlichen Formen im Querschnittsbild direkt aneinander und werden sie durch die Opening-Operation nicht voneinander getrennt, dann werden die Ellipsen ineinander überlagernd erkannt. Die Ellipsen werden dadurch größer und weiter ineinander überlagernd erkannt, als es im Querschnittsbild real der Fall ist.

- Voxel in Octree-Datenstruktur speichern

  Es ist effizienter, das Volumen nur an notwendigen Stellen hochauflösend zu diskretisieren. Dadurch wird Speicherplatz bei großen homogenen Teilen des Volumens eingespart.

- Effizienterer Flood-Fill

  Der einfache Flood-Fill-Algorithmus betrachtet jeden Pixel bzw. Voxel mehrmals. Im 2-dimensionalen Fall ist der Scanline-Fill-Algorithmus eine effizientere Möglichkeit um das Innere eines Polygons zu füllen.

[1] `http://www.openfoam.com/`

Verwendete Software

- Python 2.6

  Diese Arbeit wurde in Python implementiert.
  `http://www.python.org/`

- NumPy 1.5

  bietet Funktionen der Linearen Algebra.
  `http://numpy.scipy.org/`

- PIL 1.1

  dient zur Codierung der Bilder im „Portable Network Graphics“-Format.
  `http://www.pythonware.com/products/pil/`

- InsightToolkit 3.20 (& WrapITK)

  stellt eine Implementierung der Watershed-Segmentierung zur Verfügung.
  `http://www.itk.org/`

- OpenCV 2.2

  bietet Implementierungen von Algorithmen der Bildverarbeitung.
  Es wurden Morphologische Operationen, die Distanztransformation und das Einpassen von Ellipsen verwendet.
  `http://opencv.willowgarage.com/`

- VTK 5.6

  dient zur Visualisierung des Volumens.
  `http://www.vtk.org/`

Literaturverzeichnis

[Hamann05] Carl H. Hamann, Wolf Vielstich; 2005: *Elektrochemie*, Wiley-VCH, ISBN 978-3527310685

[Linden01] David Linden; 2001: *Handbook of Batteries*, McGraw-Hill Professional, ISBN 978-0071359788

[Doyle93] C.M. Doyle, J. Newman, T. Fuller; 1993: *Modeling of galvanostatic charge and discharge of the lithium/polymer/insertion cell*, Journal of the Electrochemical Society, 140 (6) 1526–1533

[Doyle95] C.M. Doyle, J. Newman; 1995: *Design and Simulation of Lithium Rechargeable Batteries*

[Santhanagopalan06] S. Santhanagopalan, Q. Guo, P. Ramadass, R.E. White; 2006: *Review of models for predicting the cycling performance of lithium ion batteries*, Journal of Power Sources 156 (2006) 620–628

[Santhanagopalan07] S. Santhanagopalan, Q. Guo, R.E. White; 2007: *Parameter Estimation and Model Discrimination for a Lithium-Ion Cell*, Journal of The Electrochemical Society, 154 (3) A198–A206

[Chen07] Y.-H. Chen, C.-W. Wang, G. Liu, X.-Y. Song, V.S. Battaglia, A.M. Sastry; 2007: *Selection of Conductive Additives in Li-Ion Battery Cathodes*, Journal of The Electrochemical Society, 154 (10) A978–A986

[Shearing10] P.R. Shearing, L.E. Howard, P.S. Jørgensen, N.P. Brandon, S.J. Harris; 2010: *Characterization of the 3-dimensional microstructure of a graphite negative electrode from a Li-ion battery*, Electrochemistry Communications 12 (2010) 374–377

[ITKGuide05] L. Ibanez, W. Schroeder, L. Ng, J. Cates; 2005: *The ITK Software Guide*, Kitware, ISBN 978-1930934108, `http://prdownloads.sourceforge.net/itk/ItkSoftwareGuide-2.4.0.pdf`

[Xie02] Yonghong Xie, Qiang Ji; 2002: *A New Efficient Ellipse Detection Method*, 16th International Conference on Pattern Recognition 2002, `http://www.ecse.rpi.edu/homepages/qji/Papers/ellipse_det_icpr02.pdf`

[Gander94] W. Gander, G.H. Golub, R. Strebel; 1994: *Least-Squares Fitting of Circles and Ellipses*, BIT Numerical Mathematics, Volume 34, Number 4, 558–578, `ftp://ftp.inf.ethz.ch/pub/publications/papers/wr/strebel/bit-ellipse.ps.gz`

[Halir98] R. Halir, J. Flusser; 1998: *Numerically Stable Direct Least Squares Fitting of Ellipses*, The Sixth International Conference in Central Europe on Computer Graphics and Visualization, `http://autotrace.sourceforge.net/WSCG98.pdf`

[Bishop07] Christopher M. Bishop; 2007: *Pattern Recognition and Machine Learning*, Springer, ISBN 978-0387310732, `http://research.microsoft.com/en-us/um/people/cmbishop/PRML/`

[Redbook07] OpenGL Architecture Review Board, D. Shreiner, M. Woo, J. Neider, T. Davis; 2007: *OpenGL Programming Guide: The Official Guide to Learning OpenGL*, Addison-Wesley Professional, ISBN 978-0321481009, `http://www.opengl.org/documentation/red_book/`

[Bezzi08] A. Bezzi, L. Bezzi, D. Francisci, R. Gietl; 2008: *Die Anwendung von Voxel in der Archäologie*, Italian Grass-User Meeting 2006, `http://www.arc-team.com/`

[VTKGuide10] Kitware; 2010: *VTK User's Guide*, Kitware, ISBN 978-1930934238

[Laine10] S. Laine, T. Karras; 2010: *Efficient Sparse Voxel Octrees*, I3D '10 Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games, `http://code.google.com/p/efficient-sparse-voxel-octrees/`

[Strassen69] V. Strassen; 1969: *Gaussian Elimination is Not Optimal*, Numerische Mathematik 13, 354–356

[Coppersmith90] D. Coppersmith, S. Winograd; 1990: *Matrix multiplication via arithmetic progressions*, Journal of Symbolic Computation 9 (3): 251–280, `http://www.cs.umd.edu/~gasarch/ramsey/matrixmult.pdf`

[Beck04] Kent Beck, Cynthia Andres; 2004: *Extreme Programming Explained: Embrace Change*, Addison-Wesley Professional, ISBN 978-0321278654