Direct Volume Rendering

Maarten Bussler[1]
Seminar: Data Visualization
maarten.bussler@tum.de
Supervisor: Prof.Dr. Rüdiger Westermann
[1] Technische Universität München

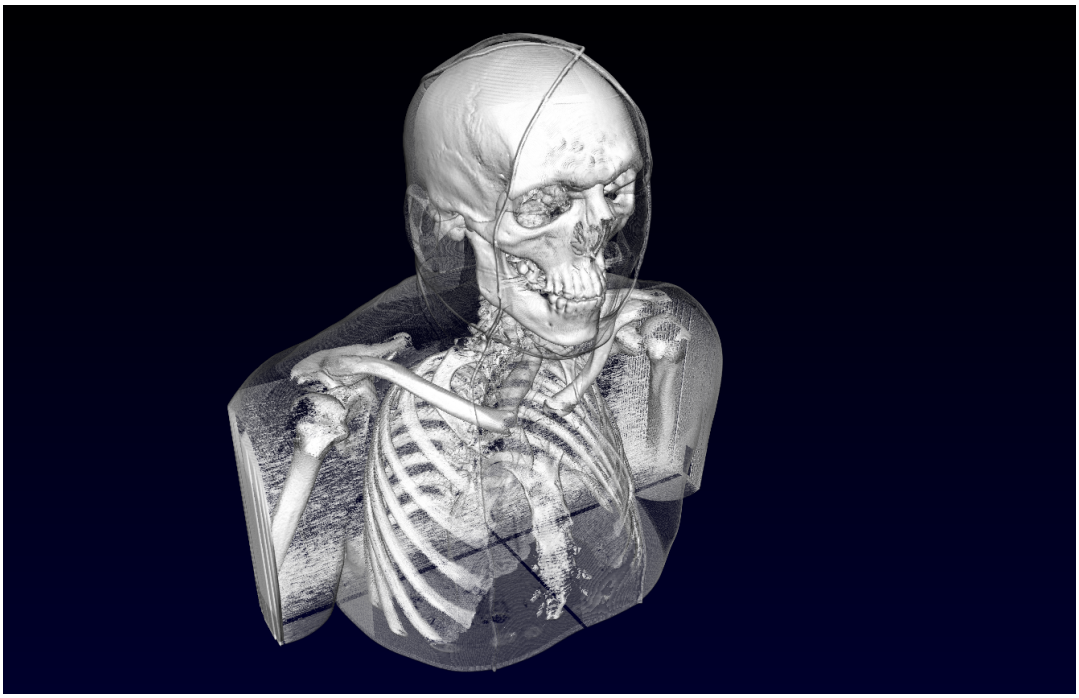

Figure 1: *Volumetrische Abbildung eines menschlichen Oberkörpers [FK10]*

Abstract

Direct-Volume-Rendering hat sich als eine effiziente Methode zur Darstellung und Analyse von volumetrischen Daten etabliert. Anwendungen in der Medizin (MRI, CT), Biotechnologie oder Flüssigkeitssimulation greifen auf Volumengrafiken zurück. Ausgiebige Forschung in diesem Bereich führte zur Entwicklung unterschiedlicher Renderingalgorithmen. Frühe Renderingmethoden von Volumendaten umfassten approximative Darstellungen des Volumens durch geometrische Primitive. Moderne Volumendaten sind jedoch zu umfangreich für diese Renderingmethode. Zudem wird häufig eine möglichst detailgetreue Abbildung des Volumens benötigt. Infolgedessen wurde die Renderingtechnik des Direct-Volume-Rendering entwickelt, die ein zweidimensionales Bild direkt aus den gesamten gegebenen dreidimensionalen Daten des Volumens extrahiert. Zusätzliche Datenstrukturen als Zwischenglied werden bei dieser Methode nicht mehr benötigt. Hohe Bildraten und ein flüssiges visuelles Feedback beim Rendern sind für das Verständnis der Volumendaten essentiell, doch eine umfangreiche Darstellung des Volumens bringt auch einen hohen Rechenaufwand mit sich. Entsprechend wurden speziell unter Einbeziehung der GPU eine Vielzahl an unterschiedlichen Optimierungsmöglichkeiten für den Renderingvorgang entwickelt.

1. Einführung

Ziel der Volumengrafik ist es, transparente und verschwommene Objekte (z.B. Gase oder Gewebeschichten) originaltreu bildlich darzustellen. Als eine weit verbreitete Visualisierungsmethode zur Abbildung von Volumendaten gilt die Abtastung des Volumens anhand von Raycasting. Abtastpunkte entlang des Rays werden mittels Interpolation ausgewertet, zugehörige Farbwerte ausgelesen, schattiert und anschließend zum resultierenden Pixelwert verrechnet. Diese Arbeit bemüht sich um eine zusammenfassende Darstellung des Renderingvorgangs einer Volumengrafik mit Hilfe von *Direct-Volume-Rendering*. Abschnitt 2 beschreibt das Rendern von Volumendaten mittels Raycasts, beginnend mit der Herleitung der zugrundeliegenden Strahlengleichung in Unterabschnitt 2.1. Unterabschnitt 2.2 stellt verschiedene Interpolationsmethoden zum Auslesen der Volumenwerte vor. Es folgt unter 2.3 eine Erläuterung über die Zuweisung von optischen Eigenschaften, wie Farbe oder Deckkraft, zu den spezifischen Volumendaten. Unterabschnitt 2.4 beleuchtet Schattierungsmöglichkeiten und die Wichtigkeit des Gradienten für den Renderingvorgang. Anschließend werden in Abschnitt 3 verschiedene Algorithmen zur Beschleunigung des Renderingvorgangs präsentiert.

2. Renderingvorgang

Ziel des *Direct-Volume-Rendering* ist es, ein 2D Bild direkt aus 3D Volumendaten zu berechnen. Die in dieser Arbeit vorgestellten Algorithmen arbeiten also ohne die Hilfe von grafischen Primitiven direkt mit den abgetasteten Werten. Unterschiedliche Methoden und Modelle zum Auslesen von Volumendaten existieren. Beispielsweise können Isosurfaces aus den Volumendaten direkt ausgegeben werden, indem der erste Voxel mit gesuchtem Schwellwert abgebildet wird. Weitere Möglichkeiten stellen die bei Röntgenaufnahmen oder CT-Scans produzierten Bilder dar, bei denen jeweils nur der Mittelwert oder der Maximalwert der untersuchten Volumenbereiche dargestellt wird. Diese Arbeit fokussiert sich auf das *Full-Volume-Rendering-Modell*. Dabei wird der Transport eines Lichtstrahls durch ein Volumen simuliert und so auch transparente Schichten sichtbar gemacht.

2.1. Renderingintegral

Full-Volume-Rendering liegt das Emission-Absorption-Modell zu Grunde. Dieses nimmt an, dass sich Lichtstrahlen nur einmal zerstreuen können, bevor sie das Volumen verlassen. Das Volumen an sich wird als eine Menge von Partikeln abstrahiert. Partikel können Licht in Form von Emission, Transmission und Reflektion absondern. Dieses Modell lässt sich vereinfachen, indem eine indirekte Beleuchtung ignoriert und nur eine einzelne Lichtquelle feststgesetzt wird. Beim Rendern des Volumens wird die Intensität und Farbe der einzelnen Lichtstrahlen durch Raycasting berechnet [HJ11]:

$$I_\lambda(x,r) = \int_0^L C_\lambda(s) * \mu(s) * exp(-\int_0^s \mu(t)dt)ds \qquad (1)$$

$I_\lambda(x,r)$ beschreibt die Lichtintensität der Abtastpunkte s aus Richtung r, die an Punkt x aufgenommen wird. $C_\lambda(s)$ entspricht dabei dem abgetasteten Licht, $\mu(s)$ der Dichte des Partikels. Die schrittweise Absorption des Lichts nach dem Beerschen Gesetz wird durch die Exponentialfunktion dargestellt. Die Berechnung eines solchen Integrals für jeden Ray ist jedoch sehr aufwendig. Eine Effizienzsteigerung bietet die Angleichung des Intergals durch eine Riemann-Summe. Diese unterteilt das Integral in regelmäßigen Abständen Δs in konstante Segmente:

$$I_\lambda(x,r) = \sum_{i=0}^{L/\Delta s-1} C_\lambda(i\Delta s) * \mu(i\Delta s)\Delta s * \prod_{j=0}^{i-1} exp(-\mu(j\Delta s)\Delta s) \qquad (2)$$

Weitere Vereinfachungen folgen. Die Deckfähigkeit α wird als das Inverse der Transparenz t definiert, die Transparenz selbst entspricht $exp(-\mu(j\Delta s)\Delta s)$. Die Exponentialfunktion wird nun mit der Taylor Serie vereinfacht: $exp(-\mu(i\Delta s)\Delta s) \approx 1-\mu(j\Delta s)$. Mit $\mu(i\Delta s)\Delta s) \approx 1-t(i\Delta s) = \alpha(i\Delta s)$ ergibt sich die Compositing Gleichung:

$$I_\lambda(x,r) = \sum_{i=0}^{L/\Delta s-1} C_\lambda(i\Delta s) * \alpha(i\Delta s) * \prod_{j=0}^{i-1} 1-\alpha(j\Delta s) \qquad (3)$$

Zusammenfassend iteriert der Algorithmus mittels Raycasts über jeden Pixel des Ergebnisbildes und ermittelt die Gewichtung des abgetasteten Integralwertes für den einzelnen Pixel.
Eine rekursive Auffassung der Compositing Gleichung ergibt das *Front-To-Back-Compositing*, nach [WMG98]:

$$\begin{aligned} C &= c_s * \alpha_s * (1-\alpha) + C \\ \alpha &= \alpha_s * (1-\alpha) + \alpha \end{aligned} \qquad (4)$$

Diesem liegt der Gedanke zu Grunde, die behandelten Volumendaten aus der Perspektive des Betrachters von vorne nach hinten zu durchlaufen. Deckfähigkeit und Farbwert eines Abtastpunktes werden mit der bereits angesammelten Deckfähigkeit $(1-\alpha)$ verrechnet und zum kumulierten Farbwert addiert, wodurch der Term für den nächsten Rekursionsschritt gebildet wird. Ein großer Vorteil dieser Methode ist die Möglichkeit zur Beschleunigung des Renderingprozesses durch *Early-Ray-Termination*. Sobald α sich 1 annähert, können alle weiteren Abtastungen entlang des Rays abgebrochen werden, da im hinteren Bereich abgesondertes Licht durch die bereits angehäufte Deckkraft abgeblockt wird.
Eine zweite intuitive Methode zum Ermitteln der Pixelwerte bietet das *Back-To-Front-Compositing*, bei dem der berechnete Farbwert eines vorherigen Abtastwertes mit Transparenz und Farbwert des aktuellen Voxels verrechnet wird, nach [WMG98]:

$$\begin{aligned} C &= C * (1-\alpha_s) + c_s * \alpha_s \\ \alpha &= \alpha * (1-\alpha_s) + \alpha_s \end{aligned} \qquad (5)$$

In der Praxis wird diese Methode jedoch selten verwendet, da sie nicht die Möglichkeit der *Early-Ray-Termination* bietet.

2.2. Integration

Volumendaten liegen in Form eines dreidimensionalen Gitternetzes V vor. Voxel repräsentieren Werte des Volumens, wie z.B. Dichte oder Druck an diskreten Positionen im Raum [HJ11]. Durch diesen Aufbau der Volumendaten ist es jedoch sehr unwahrscheinlich, dass ein Raycast beim Abtasten genau auf ein Voxel trifft. Zum Auslesen der Werte $f(x,y,z)$ zwischen den Gittern muss Interpolation angewandt werden, die mit einer hohen Rechenleistung verbunden ist. Es existiert eine Vielzahl von anwendbaren Interpolationsmöglichkeiten.

Die einfachste Interpolationsmethode stellt die *Nearest-Neighbor-Funktion* dar. Diese ordnet einem Abtastwert an einer beliebigen Position den Wert des nächstgelegenen Voxels zu [HJ11]:

$$f(x,y,z) = V(round(x), round(y), round(z)) \tag{6}$$

Ein großer Vorteil dieses Algorithmus ist seine Effizienz. Jedoch werden durch diese Interpolationsmethode konstante Wertebereiche um die Voxel hervorgebracht. Dies führt zu einer kantigen Darstellung des Ergebnisbildes, weshalb *Nearest-Neighbor-Interpolation* nur selten angewandt wird.
Eine in der Praxis dagegen häufig verwendete Interpolationsmethode stellt die *lineare*, bzw. *trilineare Interpolation* dar. Für eine dreidimensionale Umgebung lässt sich der Algorithmus als drei aufeinanderfolgende Interpolationsetappen mit sieben linearen Interpolationen implementieren (Fig. 2). Es seien x_d, y_d und z_d die Entfernung des Abtastpunktes an Position (x,y,z) zum linken, unteren, hinteren Voxel in der Interpolationsumgebung. Die ersten vier Interpolationen laufen entlang der X-Achse [WMG98]:

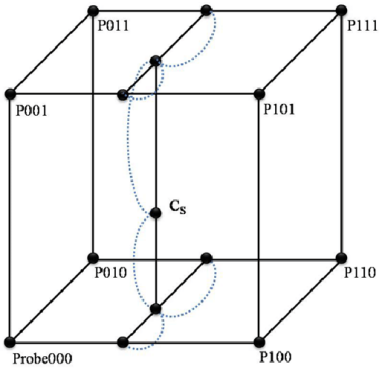

Figure 2: *Schema einer trilinearen Interpolation [CCH11].*

$$\begin{aligned} p_{00} &= p_{000} * (1 - x_d) + p_{100} * x_d \\ p_{01} &= p_{001} * (1 - x_d) + p_{101} * x_d \\ p_{10} &= p_{010} * (1 - x_d) + p_{110} * x_d \\ p_{11} &= p_{011} * (1 - x_d) + p_{111} * x_d \end{aligned} \tag{7}$$

Die so resultierenden Werte werden entlang der Y-Achse verrechnet:

$$\begin{aligned} p_0 &= p_{00} * (1 - y_d) + p_{10} * y_d \\ p_1 &= p_{01} * (1 - y_d) + p_{11} * y_d \end{aligned} \tag{8}$$

Als letztes erfolgt eine lineare Interpolation entlang der Z-Achse, aus der das Interpolationsergebnis berechnet wird:

$$C_s = p_0 * (1 - z_d) + p_1 * z_d \tag{9}$$

Eine *trilineare Interpolation* eliminiert die Nachteile der *Nearest-Neighbor-Interpolation*, jedoch treten weiterhin kantige Übergänge zwischen Voxeln mit stark schwankenden Werteunterschieden auf. *Trikubische Interpolation* verkörpert eine dritte Interpolationmethode. Ziel dieses Interpolationsalgorithmus ist die Berechnung eines Polynoms in Hermite-Form, das als Spline den Bereich zwischen zwei Voxeln interpoliert. Anders als bei den vorherigen Interpolationsmethoden werden dabei nicht nur die direkten Nachbarn sondern auch die weitere Umgebung des Abtastpunktes berücksichtig [ML94] (Fig. 3). Die daraus resultierende hohe Genauigkeit erfordert aufgrund der großen räumlichen Ausdehnung der Interpolation viel Rechenkraft, sodass *trikubische Interpolation* selten eingesetzt wird.

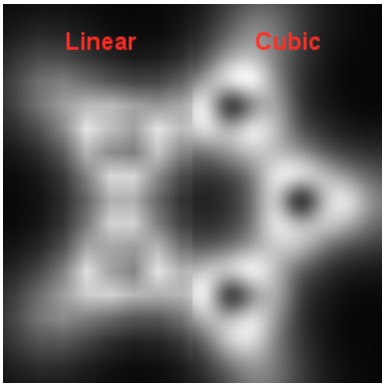

Figure 3: *Vergleich trilinearer und trikubischer Interpolation [Rui19].*

Ungeachtet der Interpolationsmethode ist Aliasing ein häufig auftretender Bildfehler. Dieses resultiert in Treppeneffekten im Ergebnisbild und wird durch eine unzureichende Abtastfrequenz verursacht. Eine Lösung bietet die Orientierung der Abtastfrequenz an der *Nyquist-Rate*, wodurch schnell wechselnde Signale entsprechend häufiger abgetastet werden [ML94].

2.3. Klassifizierung

Die im Renderingintegral benötigten Farbdaten und Deckfähigkeiten sind nicht unmittelbar im Volumen gespeichert. Damit Volumendaten durch Abtasten von Raycasts zu einer grafischen Repräsentation verarbeitet werden können, muss demnach zunächst eine Abbildung der in den Voxeln gespeicherten Volumenwerte, wie z.B. Dichte, auf eine optische Eigenschaft erfolgen. Dies geschieht mit Hilfe von *Transfer-Funktionen*, welche die Beschaffenheit des Volumens abhängig von Nutzereingaben definieren [Kin02]. Stellt ein gespeicherter Voxelwert beispielsweise eine niedrige H_20 Konzentration dar, so kann dies auf einen Knochen hindeuten. Entsprechend können passende Farbwerte gewählt werden, um diesen von anderen Objekten im Volumen (z.B. Fettgewebe) abzuheben.
Werden Voxelwerte zuerst auf ihre Farbwerte und Deckkraft abgebildet und diese anschließend interpoliert, spricht man von *Preklassifikation* [HJ11]. Eine frühe Zuweisung der Voxelwerte kann aufgrund der Nichtlinearität und der glättenden Eigenschaften der Interpolation Bildfehler verursachen, die besonders unter hoher Auflösung der Volumendaten auftreten, da Abgrenzungen zwischen Objekten nur unscharf dargestellt werden können. Eine Lösung bietet die vorzeitige Verrechnung der zugewiesenen Farbe eines Voxels mit seiner Deckkraft. Der resultierende Vektor $(R\alpha, G\alpha, B\alpha, \alpha)$ wird anschließend interpoliert. Besonders für den Bereich der medizinischen Visualisierung bietet *Preklassifikation* große Vorteile. *Partial-Voluming* beschreibt den Umstand, dass ein Voxel gleichzeitig mehrere verschiedene Objekte innerhalb des Volumens repräsentiert. Folglich kann eine eindeutige Zuordnung der optischen

Eigenschaften nicht erfolgen. Im Zuge der *Preklassifikation* können Voxel jedoch unter Verwendung einer statistischen Analyse von äußeren Attributen in verschiedene Materialtypen, wie z.B. Knochen oder Muskelfasern, aufgeteilt werden. Aufgrund dieser Zuordnung kann anschließend eine Abbildung auf Farbwerten und Lichtdurchlässigkeit erfolgen.
Eine beliebte Alternative zur *Preklassifikation* bietet das Modell der *Postklassifikation*. Hierbei werden die Volumenwerte zunächst direkt von Raycasts interpoliert und das Ergebnis anschließend auf *RGB*α Werte abgebildet [Erf07]. *Postklassifikation* ist einfacher zu implementieren und resultiert in schärferen Bildern als *Preklassifikation*, wodurch es, außer im medizinischen Bereich, das häufiger vertretene Modell darstellt (Fig. 4).

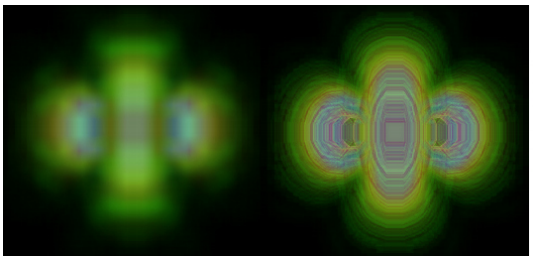

Figure 4: *Vergleich Preklassifikation(links) und Postklassifikation(rechts), nach [Erf07].*

2.4. Schattierung

Schattierung trägt entscheidend zum Verständnis der Volumendaten durch den Betrachter bei, indem Intensität und Farbe reflektierten Lichts eines Voxels abhängig vom Winkel des Betrachters und Beschaffenheit der Oberfläche bestimmt werden (Fig. 5). Aufgrund

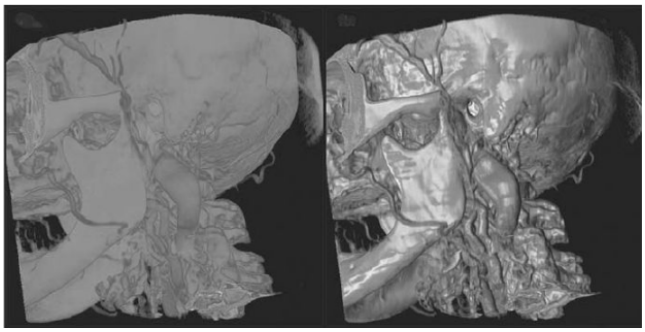

Figure 5: *Darstellung des CT Scan eines menschlichen Kopfes ohne(links) und unter Verwendung von Phong-Schattierung(rechts) [HJ11].*

seiner Einfachheit wird dabei üblicherweise *Phong-Schattierung* eingesetzt. Diese kann in seiner Ursprungsform jedoch nicht verwendet werden, da in den Volumendaten keine Oberflächen und somit auch keine Normalen als Senkrechte der Oberfläche vorliegen. Eine approximative Lösung stellt die Berechnung des Gradienten ∇f an jedem Voxel dar. Der Gradientenvektor setzt sich bei einer kontinuierlichen Funktion $f(x,y,z)$ aus den partiellen Ableitungen in jede Koordinatenrichtung zusammen und zeigt daher immer in die Richtung der größten Änderung. Ein starker Gradient kennzeichnet einen Materialübergang und damit eine starke Reflektion. Aufgrund des diskreten Aufbaus der Volumendaten müssen Filter zum Bestimmen des Gradienten eingesetzt werden. Am weitesten verbreitet ist dabei der *Central-Difference-Gradient*, der aus den lokalen Differenzen der Voxelwerte entlang aller drei Koordinatenachsen berechnet wird [HJ11]:

$$\begin{bmatrix} g_x \\ g_y \\ g_z \end{bmatrix} = \begin{bmatrix} V(x-1,y,z) \\ V(x,y-1,z) \\ V(x,y,z-1) \end{bmatrix} - \begin{bmatrix} V(x+1,y,z) \\ V(x,y+1,z) \\ V(x,y,z+1) \end{bmatrix} \tag{10}$$

Auch außerhalb von *Phong-Schattierung* wird der Gradient benötigt. Im Rahmen der Klassifikation kann Opazität an die Stärke des Gradienten gekoppelt werden, um Oberflächenumrisse zu betonen und den Einfluss fehlerhafter Daten abzuschwächen. Besonders bei medizinischen Volumendaten werden häufig mehrere Gewebeschichten übereinander liegend dargestellt. Damit die verschiedenen Ebenen für das menschliche Auge hinreichend unterscheidbar abgebildet werden, muss die Deckkraft des Gewebeinnenraumes gegenüber den Gewebegrenzen gedämpft werden. Dies wird durch eine simple Skalierung der Deckkraft mit dem lokalen Gradienten implementiert [Lev88]. Da sich der Gradient am stärksten an Materialübergängen ausprägt, wird somit das Profil des Gewebes akzentuiert.

3. Optimierungstechniken

Durch die Anwendung von komplexen Raycasts zum Abtasten der Voxelwerte und aufwendige Interpolationen ist das Rendern von Volumendaten mit einem hohen Rechenaufwand verbunden, dem die CPU als zentrale Recheneinheit oft nicht gewachsen ist. Entsprechend wird der Renderingvorgang häufig auf die GPU (Graphics Processing Unit) ausgelagert. Effiziente Kommunikation durch hohe Bandbreite und paralleles Berechnen mehrerer Rays aufgrund paralleler Recheneinheiten auf der GPU bewirken eine signifikante Beschleunigung des Renderingvorgangs [KW03]. Aber auch auf Softwareebene kann eine Optimierung des Renderingprozesses erfolgen. Mit Hilfe der im Folgenden ausgeführten Algorithmen kann die Anzahl an durchgeführten Abtastungen und Berechnungen entlang des Rays minimiert werden, wodurch eine Beschleunigung des Renderingprozesses unter geringem Qualitätsverlust des Ergebnisbildes erzielt wird.
Eine intuitive Methode zum Simplifizieren des Renderingprozesses stellt die bereits erwähnte Beschleunigung durch *Early-Ray-Termination* dar. Diese wird bei *Front-To-Back-Compositing* angewendet. Hierbei durchquert ein Ray das Volumen von vorne nach hinten, wodurch sich auch die angesammelte Deckkraft α summiert. *Early-Ray-Termination* folgt dem Prinzip, die Akkumulation von Abtastpunkten entlang des Rays zu unterbrechen, wenn der Einfluss des einzelnen Punktes auf den berechneten Pixel minimal wird. Diese Technik führt jedoch zu einem systematischen Fehler im Ergebnisbild, der aus einer Differenz zwischen dem tatsächlich im Volumen vorhandenen und dem vom Raycast zurückgeliefertem Licht resultiert. Eine weiterentwickelte Version dieser harten Raytermination stellt ein *Russisches-Roulette-Schema* dar, das die-

sen Fehler eliminieren soll [DH92]. Dabei werden mit Hilfe einer Wahrscheinlichkeitsverteilung Rays mit geringem Einfluss terminiert, sobald α einen bestimmten Schwellwert erreicht. Die Gewichtung der verbleibenden Rays wird dagegen erhöht, wodurch die Gesamtmenge an Licht im Volumen nicht verändert wird.
Volumendaten enthalten häufig weite Bereiche, die mit leerem Raum gefüllt sind. Eine intensive Abtastung durch Rays von Voxelwerten in diesen Regionen ist von wenig Interesse, da Berechnungen in diesem Teil des Volumens nicht zum Ergebnisbild beitragen. *Empty-Space-Skipping* verkörpert eine weitere Methode zur Beschleunigung des Renderingprozesses, indem ebendieser transparente oder leere Raum des Volumens beim Abtasten übersprungen wird. Doch effizientes Traversieren von leeren Regionen im Volumen ist mit großem Aufwand verbunden. Entsprechend existieren unterschiedliche Implementierungsansätze, die auf unterschiedliche Datenstrukturen zurückgreifen.
Eine häufig angewandte Technik ist *Content-Based-Space-Skipping*, bei dem aufgrund ihrer fehlenden Deckkraft als unsichtbar klassifizierte Abtastpunkte übersprungen werden. Dabei unterteilen zusätzliche Datenstrukturen das Volumen in grobe Abschnitte und speichern zudem Informationen, wie z.B. räumliche Grenzen oder minimale- bzw. maximale Volumenwerte innerhalb der Region. Beim Rendern wird vor der konkreten Verrechnung der Abtastpunkte auf die Datenstruktur zurückgegriffen und überprüft, ob in dem behandelten Abschnitt überhaupt interessante Daten vorliegen. Auf diese Weise kann die Abtastfrequenz entlang des Rays angepasst werden, wenn ein als transparent klassifizierter Bereich durchquert wird [KW03].
Eine zweite Möglichkeit stellt *Geometry-Based-Space-Skipping* dar. Dieses beschreibt das Überspringen von Abtastungen entlang des Rays abhängig von den konkreten Positionen der Abtastpunkte. PARC (Polygon Assisted Ray Casting) verkörpert eine weit verbreitete Auslegung dieser Methode. Dabei wird das Volumen in verschiedene Zellen gegliedert und deren Regionsgrenzen durch Polyeder angenähert [SA95]. Enthält eine Zelle nur Voxelwerte unter einem bestimmten Schwellwert, so wird diese als leer klassifiziert. Zum Einsatz kommen zwei Tiefenpuffer, die auf Anfang und Ende des Polyeders verweisen. Da der Polyeder die behandelte Volumenregion vollständig umschließt, können mit Hilfe der Tiefenpuffer Start- und Endpunkte der Abtastungen entlang des Rays festgelegt werden, wodurch leere Regionen des Volumens effektiv übergangen werden können.
Cohen und Sheffer [CS94] diskutieren *Proximity-Clouds* als eine weitere Ausführung von *Empty-Space-Skipping*. Diese folgen dem Ansatz, unter Verwendung einer Distanzkarte die Abtastfrequenz von Rays, die weit von einem interessanten Objekt entfernt sind, zu regulieren (Fig. 6). Jede Schicht der Wolke codiert eine bestimmte Distanz zum nächstgelegenen nichtleeren Objekt. Durchquert ein Ray eine Wolke, kann durch diesen codierten Wert eine bestimmte Distanz innerhalb des Volumens übersprungen werden. Reduktion der Abtastanzahl entlang des Rays und Fokus der Abtastpunkte um einflussreiche Objekte im Volumen bewirken eine Effizienzsteigerung des Renderingvorgangs. Da die gespeicherten Volumendaten in leeren Regionen ohnehin wertlos sind, können an ihrer Stelle die Werte der Distanzkarte gespeichert werden. Der Einsatz dieser Technik erfordert demnach keinen zusätzlichen Speicherplatz.
Danskin und Hanrahan [DH92] stellen eine Verknüpfung von *Empty-Space-Skipping* und *Importance-Sampling* vor, bei der neben der Opazität von Abtastpunkten auch vorhandene Homogenität im Datensatz berücksichtigt wird. Vor dem Hintergrund des *Importance-Sampling* werden folgende Anforderungen an eine hochwertige Abtastmethode gestellt:

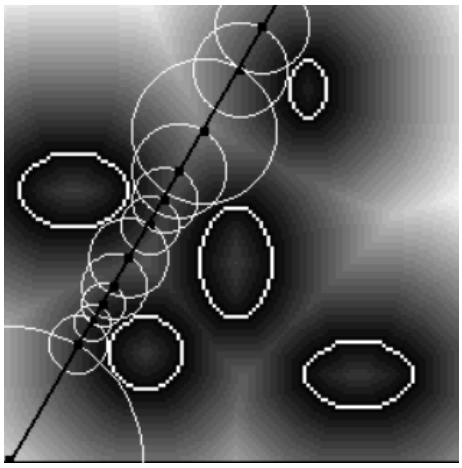

Figure 6: *Raytraversierung durch ein Volumen unter Einsatz von Proximity Clouds. Je weiter sich der Ray von den interessanten Regionen (weiß umrandet) entfernt, desto größer werden die Abstände zwischen den Abtastungen [CS94].*

- Bereiche mit geringen Volumenwerten und entsprechend geringem Einfluss auf das Renderingintegral sollen selten abgetastet werden.
- Mit steigender Deckkraft entlang des Rays soll die Abtastfrequenz abnehmen.

Dazu werden an Pyramiden angelehnte Datenstrukturen verwendet. Diese Volumenpyramiden erfassen die abgebildeten Daten auf ihren Stufen in unterschiedlichen Auflösungen. Die erste Ebene repräsentiert dabei die originalen Daten, alle folgenden Ebenen enthalten eine reduzierte Version der ersten Ebene (Fig. 7). Eine Vielzahl an unterschiedlichen Typen von Volumenpyramiden für unterschiedliche Optimierungsansätze existieren. Die Mittelwertpyramide stellt die Basis vieler Beschleunigungsalgorithmen dar. Diese speichert in jedem Eintrag der n*ten* Schicht der Pyramide den Mittelwert der Kinder aus der $n-1$*ten* Schicht, wodurch jede Ebene einen gröberen Näherungswert für die untere Ebene darstellt. Maximum-, bzw. Minimumpyramiden speichern Extremwerte des

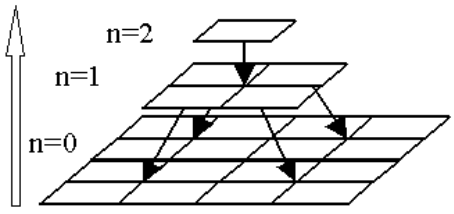

Figure 7: *Aufbau einer Pyramidendatenstruktur [IFP19].*

Volumendatensatzes, um Bereiche mit insignifikanten Volumendaten effizient zu überspringen. Aufgrund der Maximum- und Minimumpyramiden lässt sich eine dritte Pyramide, die Entfernungspyramide, berechnen. Diese bildet sich aus dem Unterschied zwischen den *RGB*α-Werten aus beiden Extremwertpyramiden. Ein geringer Wert der Entfernungspyramide deutet auf einen wertehomogenen Bereich im Volumen hin. Diese Region kann in der Folge effizient durch Abtasten einer höheren Ebene der Mittelwertpyramide approximiert werden, ohne dass große Rundungsfehler zu befürchten sind. Im Umkehrschluss signalisiert ein hoher Wert der Entfernungspyramide einen großen Abtastbedarf.
Ein einfacher Algorithmus, der sich dieses Pyramidenschema zu Nutze macht, ist *Presence-Acceleration*. Dabei wird auf eine Kombination von Maximum(α)- und Mittelwertpyramide zurückgegriffen. Unterschreiten die Werte der Maximumspyramide einen bestimmten Schwellwert, so zeigt dies eine Volumenregion mit geringer Opazität und somit geringer Bedeutung für den Renderingvorgang. Ein gründliches Abtasten der Region ist nicht nötig, stattdessen wird repräsentativ für die gesamte Region ein einzelner Wert der Mittelwertpyramide auf einer höheren Ebene ausgewertet.
Als Gegenstück zu *Presence-Acceleration* präsentieren Danskin und Hanrahan das Verfahren der *Homogeneity-Acceleration*. Dieses verwendet eine Entfernungspyramide zusammen mit einer Mittelwertpyramide und bezieht sich im Gegensatz zu *Presence-Acceleration* nicht auf die Deckkraft, sondern auf die Homogenität der behandelten Daten einer Volumenregion. In Regionen mit hoher Homogenität kann aufgrund der Gleichheit der Volumendaten ein einzelner Abtastpunkt der Mittelwertpyramide ausgewertet werden, da dieser eine gute Annährung der restlichen Volumenwerte der Region darstellt.
β-*Acceleration* raffiniert diese Technik, indem es die entlang des Rays akkumulierte Opazität einbezieht. Im Kern dieser Methode steht der Gedanke, dass der merkliche Fehler einer groben Abtastung gering ausfällt, wenn dieser an eine geringfügige Deckkraft gekoppelt ist. In diesem Sinne beschleunigt der Algorithmus durch Entnahme von Abtastwerten aus immer höheren Ebenen der Mittelwertpyramide, während die optische Distanz entlang des Rays steigt. Danskin und Hanrahan empfehlen eine Implementation von *Homogeneity-Acceleration* unter Berücksichtigung von β-*Acceleration*, um die Vorteile beider Techniken zu kombinieren.

4. Zusammenfassung

Diese Arbeit stellt die Visualisierung von Volumengrafiken durch *Direct-Volume-Rendering* vor. Dabei werden Volumendaten in Form eines diskreten Voxelgitters repräsentiert. Abtastwerte im Volumen werden mit Hilfe von Integration der Voxel ausgewertet, optische Eigenschaften durch *Transfer-Funktionen* interpretiert und entlang von Rays zu einem Pixelwert im Ergebnisbild summiert. Einfache Schattierungstechniken, wie *Phong-Schattierung*, tragen entscheidend zum Realismus des abgebildeten Volumens bei.
Ferner wurden verschiedene Modelle zur Optimierung des Renderingprozesses, wie *Early-Ray-Termination* und *Empty-Space-Skipping*, in unterschiedlichen Ausführungen vorgestellt. Aufgrund dieser Algorithmen ist es möglich, selbst umfangreiche Volumendaten mit blickdichten Strukturen oder leeren Bereichen effizient zu rendern.
Zukünftige Forschungen im Bereich der Volumengrafik werden sich insbesondere mit der Limitation des Speichers auseinandersetzen müssen. Mit der Entwicklung immer leistungsfähigerer Computer können immer exaktere und detailliertere Volumendaten bereitgestellt werden. Auch die Implementation der Optimierungsprozesse ist abhängig von zusätzlichen Datenstrukturen und vervielfältigt so den Bedarf an externem Speicher mit einer hohen Lese- und Schreibgeschwindigkeit. Eine mögliche Lösung dieses Problems repräsentiert die von Boer et al. vorgestellte Architektur zur Leistungssteigerung der Speicherbandbreite. Diese ermöglicht durch eine Kombination von RDRAM und Cache-Architektur einen parallelen Zugriff von acht Speichereinheiten auf die Volumendaten, wodurch die Leserate des Speichers beträchtlich erhöht wird [DBGHM97]. Trotz erheblicher Forschung in diesem Bereich stellen das kontinuierliche Wachstum der zu verarbeitenden Daten und die gleichzeitig limitierte Menge an verfügbaren Speicher, sowie begrenzte Bandbreiten für Lesen und Schreiben von Daten auf der GPU eine große Hürde für die modernen Volumengrafik dar.

References

[CCH11] CHUNG J., CHO S., HAH J.-M.: A python-based docking program utilizing a receptor bound ligand shape: Pythdock. *Archives of pharmacal research 34* (09 2011), 1451–8. 3

[CS94] COHEN D., SHEFFER Z.: Proximity clouds–an acceleration technique for 3d grid traversal. *The Visual Computer 11*, 1 (1994), 27–38. 5

[DBGHM97] DE BOER M., GRÖPL A., HESSER J., MÄNNER R.: Latency-and hazard-free volume memory architecture for direct volume rendering. *Computers & Graphics 21*, 2 (1997), 179–187. 6

[DH92] DANSKIN J. M., HANRAHAN P.: Fast algorithms for volume ray tracing. *VVS 92* (1992), 91–98. 5

[Erf07] ERFURT R.: Vorintegriertes volume rendering: Slicing vs. raycasting. 4

[FK10] FOGAL T., KRÜGER J.: Tuvok, an Architecture for Large Scale Volume Rendering. In *Proceedings of the 15th International Workshop on Vision, Modeling, and Visualization* (November 2010). 1

[HJ11] HANSEN C. D., JOHNSON C. R.: *Visualization Handbook*. Academic Press, August 30, 2011. 2, 3, 4

[IFP19] IFP: Research projects in image processing. http://www.ifp.illinois.edu/~chenyq/research/ip/ip_research.html, 2019. Accessed: 2019-11-29. 5

[Kin02] KINDLMANN G.: Transfer functions in direct volume rendering: Design, interface, interaction. *Course notes of ACM SIGGRAPH 3* (2002). 3

[KW03] KRUGER J., WESTERMANN R.: Acceleration techniques for gpu-based volume rendering. In *Proceedings of the 14th IEEE Visualization 2003 (VIS'03)* (2003), IEEE Computer Society, p. 38. 4, 5

[Lev88] LEVOY M.: Display of surfaces from volume data. *IEEE Computer graphics and Applications 8*, 3 (1988), 29–37. 4

[ML94] MARSCHNER S. R., LOBB R. J.: An evaluation of reconstruction filters for volume rendering. In *Proceedings Visualization'94* (1994), IEEE, pp. 100–107. 3

[Rui19] RUIJTERS D.: Cuda cubic b-spline interpolation (ci). http://www.dannyruijters.nl/cubicinterpolation/, 2019. Accessed: 2019-11-29. 3

[SA95] SOBIERAJSKI L. M., AVILA R. S.: A hardware acceleration method for volumetric ray tracing. In *Proceedings of the 6th conference on Visualization'95* (1995), IEEE Computer Society, p. 27. 5

[WMG98] WITTENBRINK C. M., MALZBENDER T., GOSS M. E.: Opacity-weighted color interpolation for volume sampling. In *IEEE Symposium on Volume Visualization (Cat. No. 989EX300)* (1998), IEEE, pp. 135–142. 2, 3