

Fernunterstützung und Zusammenarbeit mit 3D Punktwolken

Masterarbeit
von

Kai Westerkamp

An der Fakultät für Informatik
Fraunhofer IOSB (IAD)

Erstgutachter:	Prof. Dr.-Ing. Rainer Stiefelhagen
Zweitgutachter:	XXXX
Betreuender Mitarbeiter:	M.Sc. Adrian Hoppe

Bearbeitungszeit: 01.06.2017 – 30.11.2017

Abstract

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Zusammenfassung

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

Inhaltsverzeichnis

1	Einleitung	1
1.1	VR	1
2	Stand der Technik	3
2.1	Section	3
3	Punktwolke	5
3.1	Frames aufnehmen und bereinigen	6
3.1.1	Aufnahme und Glättung	6
3.2	Zusammenfügen von Frames	7
3.2.1	Ungenauigkeiten im Lighthouse Tracking	7
3.2.2	Kalibrierung Kinect zu Vive	7
3.3	Ergebnisse	7
3.4	Section	8
4	3D Tiles un GLTF	9
4.1	3D Tiles	9
4.1.1	Tileset und Tiles	9
4.2	glTF	9
4.2.1	Struktur	10
4.2.1.1	Buffers and Accessors	10
	Literaturverzeichnis	11

1. Einleitung

(TODO)

ToDo

1.1 VR

2. Stand der Technik

related

2.1 Section

paper

3. Punktwolke

Das Erzeugen der Punktwolke sollte einfach und schnell funktionieren und mit einer Kinect erfolgen. Aus einem Frame der Kinect, bestehend aus Farbbilde und Tiefenbild, lässt sich einfach eine Punktwolke relativ zur Tiefenkamera der Kinect errechnen. Eine Aufnahme beinhaltet aber nur alle Informationen die aus den 2D Bilder zurückgerechnet werden können. Das heißt man erhält nur eine Seite des Objektes gut aufgelöst und man hat noch einige Artefakte die sich aus dieser Berechnung ergeben. Für die Darstellung in einer VR Umgebung ist dies nicht ausreichend. Der Betrachter kann sich frei in der virtuellen Welt bewegen und erkennt schnell die nicht vorhandenen Informationen und Fehler.

Ein Problem hierbei ist das Kanten und Flächen die nicht Senkrecht zur Kamera sind, durch unzureichende Informationen in den Ausgangsdaten falsch angenähert werden (siehe 3). **(Bild 2 ändern / vergrößern?)**

ToDo

Das zweite Problem das es zu lösen galt war das zusammenfügen von mehreren Aufnahmen aus unterschiedlichen Perspektiven zu einer großen zusammenhängenden Punktwolke. Um 2 Frames miteinander zu verbinden braucht man die relative Transformation zwischen den beiden Aufnahmen, bzw. der beiden Kamerapositionen. Bei bestehenden Algorithmen wird dies zum Beispiel durch Flankenerkennung oder zurückrechnen der Bewegung der Kamera erreicht **(quellen)**. Solche verfahren sind meist rechenaufwändig und zeitintensiv. Für diese Arbeit war es das Ziel die Kinect mit dem Lighthouse Tracking System zu

ToDo

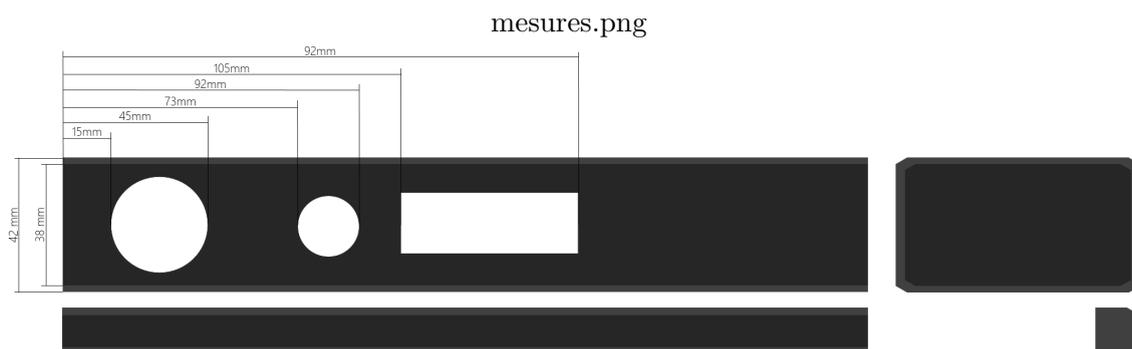
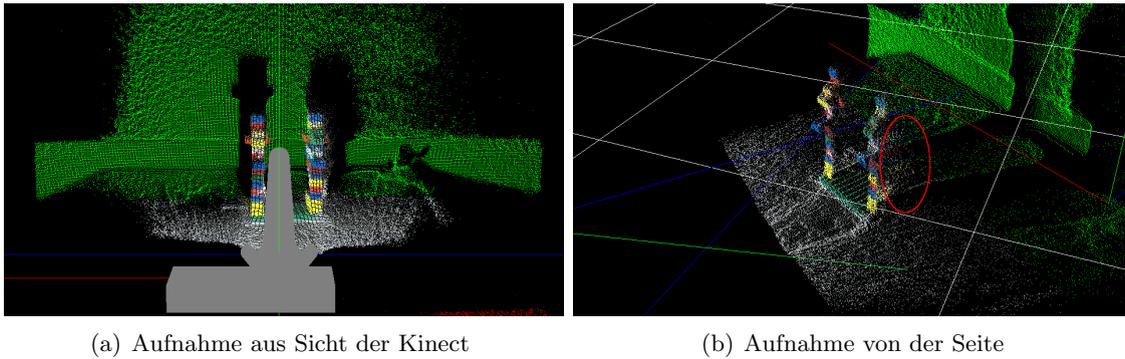


Abbildung 3.1: Abmessungen der Kienct. Der Tiefensensor liegt in der mittleren runden Öffnung. Quelle:[Kinb]



(a) Aufnahme aus Sicht der Kinect

(b) Aufnahme von der Seite

Abbildung 3.2: Aufnahmen aus verschiedenen Perspektiven In Bild b) sind falsche Punkte zu sehen die durch die Rekonstruktion aus einem 2D Bild entstehen.

verbinden. Die Trackingdaten aus SteamVR, bzw OpenVR geben uns eine globale Position aller Aufnahmen und vereinfachen das Erzeugen einer großen Punktwolke.

3.1 Frames aufnehmen und bereinigen

Als ersten Schritt wird ein Frame mit der Kinect aufgenommen und das Tiefenbild geglättet. Anschließend wird das Tiefen und Farbbild in 3D Punkte umgewandelt und unerwünschte Punkte verworfen.

3.1.1 Aufnahme und Glättung

Das Aufnehmen einer kleinen Punktwolke wird das Kinect SDK verwendet. Sowohl Tiefenbild auch als auch Farbbild kann man in der API erhalten. Anschließend wird das Tiefenbild geglättet. Die Rohdaten sind teilweise sehr verrauscht und man erhält eine Punktwolke mit glatteren Flächen. Hierfür braucht man einen Filter der zwar die Flächen glättet, aber gleichzeitig Objektkanten erhält. Ein Bilateral Filter erzielt den gewünschten Effekt ist aber relativ Rechenaufwändig. Verwendet wurde ein Gaus Filter und ein anschließender Vergleich zwischen Original und geglättetem Bild. **(Referenz Manuel Martins fastdepthnoiseremoval)**

ToDo

Nach der Glättung des Tiefenbildes wird dieses in eine Punktwolke umgewandelt. Hierfür wurde ebenfalls das Microsoft Kinect SDK verwendet das hierfür alle benötigten Methoden bereitstellt.

Nach der Umwandlung werden noch weiter Punkte verworfen. Zunächst werden alle Punkte zu denen keine Farbe zugeordnet werden kann verworfen. Auch alle Punkte die zu nah oder zu weit vom Sensor entfernt sind, werden nicht weiter betrachtet. Je weiter das Objekt entfernt, desto ungenauer werden die Aufnahmen. Im folgenden wurde ein Mindestabstand von 30cm und ein Maximalabstand von 90cm verwendet.

ToDo

Als letztes filtern wir alle Flächen die nicht Senkrecht zur Kamera sind. **(bild ref)** Diese Flächen entstehen durch die Umwandlung von einem 2D Tiefenbild in einer 3D Punktwolke. Die benötigten Informationen fehlen an dieser Stelle und Punkte werden auf die Fläche zwischen Oberflächenobjekt und Hintergrund gesetzt. Diese Ebene stimmt nicht mit der wirklichen Oberfläche überein und müssen entfernt werden. Hierfür wird die Oberflächennormale verwendet. Die Normale wird aus dem Tiefenbild geschätzt.

$$\begin{aligned}
 dzX Axis &= depthAt[x + 1, y] - depthAt[x - 1, y] \\
 dzY Axis &= depthAt[x, y + 1] - depthAt[x, y - 1] \\
 Normale &= Normalize(-dzX Axis, -dzY Axis, 1.0)
 \end{aligned}
 \tag{3.1}$$

Mit dem Skalarprodukt lässt sich der Winkel zwischen dem Kameravektor $(0, 0, 1)$ und Normale ausrechnen. Ein maximaler Winkel von 65° hat in den Tests ein gutes Ergebnis geliefert. **(grad symbol) (Quellen auf Kinect und Lighthouse)**

ToDo
ToDo

3.2 Zusammenfügen von Frames

Ein wichtiger Teil beim dem Aufnehmen der Punktwolke ist das zusammenführen von mehreren Frames. Hierfür wurde die Kinect mit dem Lighthouse Tracking System verbunden und verzichtet damit auf aufwändige Berechnungen.

(Foto Halterung) Im lokalen Koordinatensystem der Kinect, also jedes Frames liegt der Ursprung in dem Tiefensensor. die Transformation *transformControllerToKinect* zwischen dem Koordinatensystem des Controllers und der Kinect wurde bestimmt und die globale Transformation des Controllers *transformController* ist in der OpenVR API abfragbar Die Transformation der lokalen Punktwolke in ein globale ist mit diesen beiden Transformationen möglich.

ToDo

$$globalPosition = transformController * transformControllerToKinect * localPosition \quad (3.2)$$

3.2.1 Ungenauigkeiten im Lighthouse Tracking

Ein großes Problem sind Ungenauigkeiten im Tracking. <https://www.roadtovr.com/analysis-of-valves-lighthouse-tracking-system-reveals-accuracy/> <http://journals.sagepub.com/doi/full/10.1177/20416>

(bild)

ToDo

(schreiben)

ToDo

3.2.2 Kalibrierung Kinect zu Vive

Eine wichtige Transformation ist die zwischen dem Koordinatensystem der Kinect und dem des Vive Controllers.

Ist zum Beispiel die Transformation entlang der X Achse der Kinect verschoben so verstärkt sich der Fehler wenn man die Kinect um 180° dreht (siehe Abbildung()) **(grad zeichen)**

ToDo

Der Ursprung der Controllers lässt sich aus den Modellen von SteamVR auslesen. Bei der Kinect ist der offiziellen Doku entnehmbar das der Ursprung in dem Tiefensensor liegt (siehe [Kina]) Aber es gibt keine offizielle Dokumentation wo dieser exakt liegt. Im Bild 3.2.2 ist aus dem chinesischen Microsoft Forum eine von Benutzern vermessene schematische Darstellung der Kinect abgebildet. Der Tiefensensor liegt hinter der kleineren runden Öffnung, aber Fertigungsungenauigkeiten machen das exakte bestimmen nicht möglich ohne die Kinect zu zerlegen.

Eine digitale Kalibrierung gestaltet sich schwierig, da das Lighthouse Tracking für den Controller selber noch einige Ungenauigkeiten mit sich bringt.

3.3 Ergebnisse

Ziel einfach schnell, ohne große berechnung Alle Seiten

Idee und Umsetzung Verknüpfung Globale Tracking mit Kinect Aufnahmen mehrerer Frames Glättung und zusammenführung

Speichern 3D Tiles

Filtern Glättung Bilaterale Filterung Fast Depth Noise Removal Manuel Martin

Probleme

Kalibrierung Kinect zu Controller Kinect Spezifikationen Genauigkeit der Kinect / Lighthouse Tracking

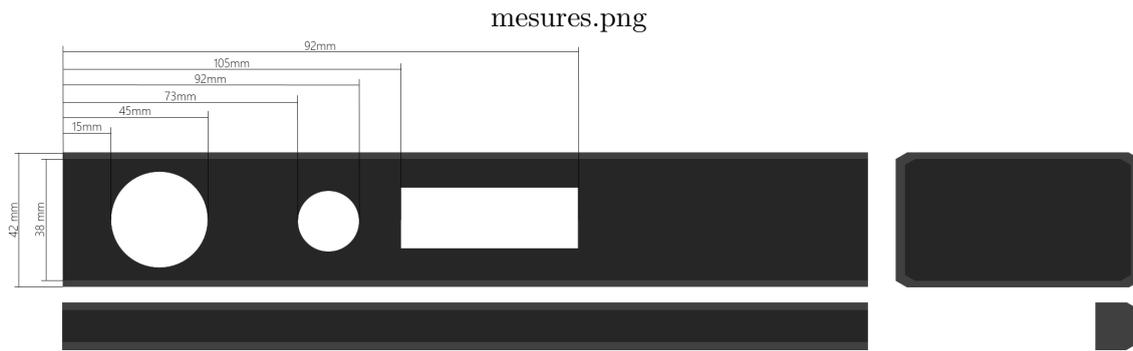


Abbildung 3.3: Abmessungen der Kienct. Der Tiefensensor liegt in der kleinen runden Öffnung. Quelle:[Kinb]

3.4 Section

ToDo (section)

4. 3D Tiles un GLTF

In diesem Kapitel wird ein grober Überblick über die Struktur und die Komponenten des GL Transmission Formats und der 3D Tiles gegeben.

3D Tiles und gltf 3D Tiles are an open specification for streaming massive heterogeneous 3D geospatial datasets Tile Struktur (Tielset) different Tiles (batched, instanced, points ...) Bounding Volumes und LOD Dynamic loading GLTF Struktur Optimized for OpenGL and streaming scenes, nodes, meshes materials animations ignored

Rendering Equation: [Kaj86] **(TODO)**

ToDo

4.1 3D Tiles

3D Tiles [3DT] ist eine neue offene Spezifikation für das streamen von massiven, heterogenen, geospatialen 3D Datensätzen. Die 3D Tiles können genutzt werden um Gelände, Gebäude, Bäume und Punktwolken zu streamen.

4.1.1 Tileset und Tiles

Die 3D Tiles e

4.2 glTF

Das GL Transmission Format (glTF [GLT]) ist ein Format zum effizienten Übertragen von 3D Szenen für GL Api's wie WebGL, OpenGL ES und OpenGL. GLTF dient als effizientes, einheitliches und erweiterbares Format zur Übertragung und Laden von 3D Daten. Im Vergleich zu aktuellen Standards wie COLADA ist glTF optimiert, schnell heruntergeladen und in eine Applikation geladen zu werden. In einer JSON formatierten Datei (.gltf) wird eine komplette Szene samt Szenegraf, Materialien und deren zugehörigen Shadern, Kameras, Animationen und Skinning Informationen übertragen. Dabei kann auf externe Dateien verwiesen werden. Diese sind zum Beispiel Binärdaten oder Bildern die für das einfache und effiziente Übertragen von Geometrie, Texturen oder den nötigen GLSL Shadern genutzt werden

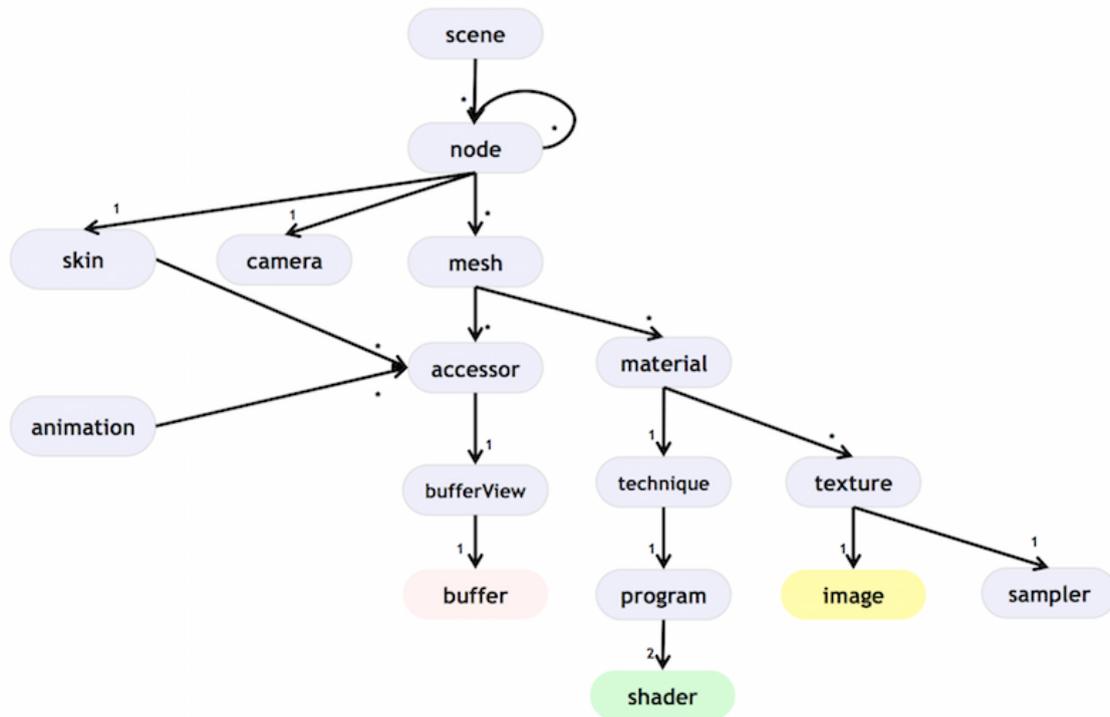


Abbildung 4.1: Struktur einer glTF Datei

4.2.1 Struktur

Die .gltf Datei bildet den Kern jedes Modells und ist eine JSON formatierte Datei. In Ihr werden alle grundlegenden Informationen wie zum Beispiel die Baumstruktur und die Materialien gespeichert (siehe Abb. 4.2.1). Eine Szene bildet hierbei den Startpunkt für die zu rendernde Geometrie. Szenen bestehen aus Knoten (Nodes) im Szenengraf, die wiederum Knoten als Kinder haben können. Jeder Knoten kann eine Transformation im lokalen Raum definieren, bestehend aus eine Translation, einer Rotation und eine Skalierung. Jeder Knoten kann eine Mesh und damit die eigentliche Geometrie referenzieren.

4.2.1.1 Buffers and Accessors

Buffer sind die eigentlichen Daten in einem Binären Block. Diese können entweder als externe Datei (.bin) oder als BASE64 encodierter String in der JSON Datei angefügt werden. Die Hauptaufgabe der Buffer ist es große mengen an Daten wie die Geometrie effizient zu übertragen.

glf right Handed y Axis up in Meters and radians

Literaturverzeichnis

- [3DT] *3d tiles spezifikation*. <https://github.com/AnalyticalGraphicsInc/3d-tiles>. Accessed: 2017-09-13.
- [GLT] *Gltf spezifikation*. <https://github.com/KhronosGroup/glTF>. Accessed: 2017-09-13.
- [Kaj86] James T. Kajiya: *The rendering equation*. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '86, pages 143–150, New York, NY, USA, 1986. ACM, ISBN 0-89791-196-2.
- [Kina] *Kinect dokumentation koordinatensysteme*. <https://msdn.microsoft.com/de-de/library/dn785530.aspx>. Accessed: 2017-11-02.
- [Kinb] *Kinect tiefensensor position*. <https://social.msdn.microsoft.com/Forums/sqlserver/ja-JP/05a6d2b3-9096-4236-b77a-691c5f047066/kinect-for-windows-v2-?forum=windowsgeneraldevelopmentissuesja>. Accessed: 2017-11-02.

Ich versichere wahrheitsgemäß, die Arbeit selbstständig angefertigt, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Abänderungen entnommen wurde.

Karlsruhe, xx.xx.xx

.....

(Max Mustermann)