Darstellung gigantischer Punktwolken auf Android Geräten

Jakob Krause

December 7, 2015

Einführung

Motivation

- Live Darstellung der Ergebnisse bei Flügen des Archaeocopters

Aufbau der Arbeit

-

Vergleichbare Arbeiten

Das Interesse an der Darstellung von großen Punktwolken ist durch das Aufkommen von modernen 3D Scanner in den letzten Jahrzehnt stark gewachsen. Für die Visualisierung von Punktmengen existieren Bibliotheken wie PCL[2]. Z.B greift die Android App "KiwiVewer"[1] auf PCL zurueck. Auch streaming wird unterstützt. Große Punktmengen sind damit alleine allerdings nicht zu bewältigen.

Erste Vorschläge zur Darstellung von sehr großer Punktmengen wurden durch das QSplat[6] Verfahren gemacht. Die Punktemenge wird durch 'multiresolution' Hierarchie auf Basis von 'bounding spheres' modelliert. Abhängig von der Kameraposition wird die Struktur bis zu einer gewissen Tiefe (Auflösung) durchlaufen. Das Verfahren kann modifiziert auch zum streaming von Punktwolken genutzt werden [7] . Für HD Displays ist der Algorithmus zu rechenintesiv weil die Berechnung pro Display Punkt von der CPU erledigt wird.

Moderne Ansätze Benutzen einen 'Multi Resolution' Octree[8]. Die Daten werden 'out-of-core' verwaltet, also dynamisch von der Festplatte geladen. Der Octree bietet den Vorteil, das Punkte dynamisch hinzugefügt werden können. Hier werden in den inneren Knoten die unterschiedliche Detailstufen gespeichert.

Eine weitere Möglichkeit ist ein Knn-Tree zu verwenden[5]. Allerdings mit der Beschränkung, dass die Punktmenge von Anfang an vollstaendig ist.

Diese Arbeit richtet sich besonders an die Darstellung auf mobilen Geräten. Die einzelnen Knoten werden auf Anfrage über das Netzwerk vom einem Server übertragen.

Beide Verfahren nutzen einen LRU-Cache um Verzögerungen zu verringern. Die Implementierung erfolgte in C++ und OpenGl ES. Für die Netzwerkkommunikation wird 'http pipelining' sowie eine Wavletkompression verwendet. Punktmengen von bis zu 470M Punkten konnten mit dem System auf eine iphone4 dargestellt werden. Der Server war mit 24GB RAM und einem i7 Prozessor ausgestattet.

Gewählter Lösungsansatz

Architektur

Die Applikation besteht entspricht einer Client-Server Architektur.

Server

Der http Server stellt dem Client die Datenstruktur zur Verfügung um benötigte Punkte zu ermitteln. Des weiteren liefert er diese Punkte auf Anfrage aus.

Da es nötig ist Punkte kontinuierlich hinzufügen kommt ein 'Multi Resolution' Octree[8] zum Einsatz.

Die Knoten des Baumes enthalten lediglich Keys zu den Daten.

Die eigentlichen Punkte werden in einer Key-Value Datenbank gespeichert und für Anfragen zu Verfügung stellt.

Das Model selbst ist auf die Blätter verteilt. Die inneren Knoten erhalten vereinfachte Abbilder von seinen Kindern.

- Programmiersprache Java
- dynamischer 'Multi Resolution' Octree
- 'Redis'[3] als Key-Value Datenbank
- Kompression

Client

Der Android-Client erhält ein Abbild der Datenstruktur vom Server und synchronisiert diese regelmaessig. Beim traversieren der Struktur werden an den Server Anfragen gesendet.

Empfangene Daten werden in einem 'last recently used' Cache gespeichert. Dieser Cache wird gleichzeitig von OpenGl als Buffer für die Punkte benutzt.

Es existiert ein Buffer für die Punktposition sowie für die RGB Werte.

Die Anfragen zum Server laufen über das http Protokoll.

- Programmiersprache Java

- 'Volley'[4] for asynchrone http Anfragen
- OpenGl ES für das Rendering

Schwierigkeiten

Dokumentation der Durchführung

References

[1] Kiwi viewer. `http://www.kiwiviewer.org/`, note = Accessed: 2014-12-03.

[2] Point cloud library. `http://www.pointclouds.org/`. Accessed: 2014-12-03.

[3] Redis. `http://redis.io/`. Accessed: 2014-12-03.

[4] Volley. `https://github.com/mcxiaoke/android-volley`. Accessed: 2014-12-03.

[5] Marcos Balsa Rodriguez, Enrico Gobbetti, Fabio Marton, Ruggero Pintus, Giovanni Pintore, and Alex Tinti. Interactive Exploration of Gigantic Point Clouds on Mobile Devices. In David Arnold, Jaime Kaminski, Franco Niccolucci, and Andre Stork, editors, *VAST: International Symposium on Virtual Reality, Archaeology and Intelligent Cultural Heritage*. The Eurographics Association, 2012.

[6] Szymon Rusinkiewicz and Marc Levoy. Qsplat: A multiresolution point rendering system for large meshes. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '00, pages 343–352, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[7] Szymon Rusinkiewicz and Marc Levoy. Streaming qsplat: A viewer for networked visualization of large, dense models. In *Proceedings of the 2001 Symposium on Interactive 3D Graphics*, I3D '01, pages 63–68, New York, NY, USA, 2001. ACM.

[8] Michael Wand, Alexander Berner, Martin Bokeloh, Philipp Jenke, Arno Fleck, Mark Hoffmann, Benjamin Maier, Dirk Staneker, Andreas Schilling, and Hans-Peter Seidel. Special section: Point-based graphics: Processing and interactive editing of huge point clouds from 3d scanners. *Comput. Graph.*, 32(2):204–220, April 2008.