Von Scandaten zu visualisierten Punktwolken

Martin Graner[1]

Zusammenfassung
Laserscanning hat sich in den letzten Jahrzehnten rasant weiterentwickelt. Altbekannte Aufnahmeverfahren wie Photogrammetrie sind verbessert worden. Hardware ist schneller, genauer, mit höherer Messreichweite und günstiger geworden. Mobile Geräte haben steigende Beliebtheitswerte und ein wachsendes Angebot von Herstellern. Daraus resultieren wiederum mehr Anwender, welche mehr Punktwolkenprojekte produzieren. Doch nicht nur in den Aufnahmeverfahren und der Hardware gibt es Fortschritt, sondern auch in der Softwareentwicklung. Immer mehr Softwareprodukte unterstützen das Einlesen, Verarbeiten und Visualisieren von Punktwolken. Des Weiteren verschieben sich die Anwendungen von bisherigen Desktoplösungen zu Cloudanwendungen. Daraus ergeben sich Schwierigkeiten in der Datenweitergabe zwischen Softwarelösungen, wie auch die effiziente Visualisierung immer größer werdender Projekte. Die Datenweitergabe von Punktwolken erfolgt in der Regel mittels bekannter Austauschformate. Die größte Schwierigkeit bei Austauschformaten ist zum einen die Befolgung des spezifizierten Standards von Firmen, zum anderen die Marktdurchdringung. Ein Format mit exzellenten Eigenschaften, wie kleiner Dateigröße, schneller Einlesegeschwindigkeit und optimaler Vorsortierung zum Prozessieren und Visualisieren wird nicht zwangsläufig vom Markt unterstützt, da Firmen stark darauf bedacht sind Kunden im eigenen Ökosystem zu halten.
In diesem Beitrag gehen wir auf den Arbeitsablauf vom Laden der Daten, auf dem Weg zur fertigen Visualisierung, ein. Außerdem werden die derzeit vom Markt am besten unterstützen Austauschformate miteinander verglichen. Zuletzt werden Visualisierungsstrategien diskutiert, da die derzeitige Hardware nicht für die Visualisierung riesiger Punktwolkenprojekte mit naiven Ansätzen ausgelegt ist.

Schlagwörter Laser scanning · Visualisierung · Punktwolken · Formate

1 Einführung

Durch die breitere Akzeptanz von Laserscanning im Aufmaß und als Grundlage zur Modellierung von Bestandsgebäuden steigt das Ausmaß von Projekten. Zusätzlich gibt es weitere Neuerungen in der Scannertechnologie. Reduzierung der Aufnahmezeit, Erhöhung der Distanzmessung und Vergrößerung der Scanauflösung führt zu immer umfangreicheren Punktwolken.

Daraus resultieren Schwierigkeiten im Bereich Datenmanagement, Dateigröße, Prozessierung und Visualisierung.

Im Kapitel 2 wird kurz auf den notwendigen Prozessablauf vom Datenträger in die Grafikkarte eingegangen. Danach werden im Kapitel 3 Punktwolkenformate, deren Speichergröße und Einlesegeschwindigkeit diskutiert. Zuletzt wird in Kapitel 4 und 5 die eigentliche Visualisierung besprochen.

2 Vom Datenträger in die Grafikkarte

In der Visualisierung mittels Computer müssen Daten mithilfe einer GPU (Graphical Processing Unit) in ein Format gebracht (rendering) werden, welches auf Bildschirmen angezeigt werden kann. Dabei können bestimmte Geometrien, wie Dreiecke, Quadrate und Linien, teilweise auch Punkte, direkt visualisiert werden, alles weitere muss jedoch in diese Geometrien überführt werden. Der Arbeitsablauf der Hardware folgt in diesem Fall vom Einlesen der Daten von einem Datenträger in den Arbeitsspeicher, entweder lokal über direkt verbundene Hardware oder über das Internet und von dort in den VRAM (Video Random Access Memory) der GPU.

Von dort kann die GPU dann auf die Daten zugreifen und diese prozessieren.

[1] PointCab GmbH, Talstr. 8, 73249 Wernau, Deutschland, E-Mail: martin.graner@pointcab-software.com

3 Punktwolkenformate

Unter dem Begriff Punktwolke verbirgt sich eine Ansammlung von Punkten im dreidimensionalen Raum. Diese Punktsammlung kann optional mit Metadaten, wie Scaninformationen und Projektionsmatrizen, verknüpft sein. Jedem Punkt in der Sammlung können diverse Attribute zugeordnet werden, wobei eine Punktwolke ohne Attribute auch zulässig ist. Typische Attribute sind Reflektivitäts bzw. Intensitätswerte, Farbinformationen, Zeitstempel, Segment- und Klassifikationsnummern, Validität und Normalen. In der Regel haben alle Punkte in der Punktwolke die gleiche Attributszusammensetzung. Bei Änderungen der Attribute gruppiert man Punkte in der Sammlung mit den gleichen Attributen zusammen. Im weiteren Verlauf werden wir Punkte mit den zugehörigen Attributen als Daten zusammenfassen.

Punktwolken werden in verschiedenen Formaten gespeichert. Standardmäßig hat jeder Gerätehersteller sein eigenes proprietäres Format. Dieses ist auf die Laserscannersysteme des Herstellers abgestimmt und wird im ersten Schritt, bei der Datenaufnahme in einem Rohformat abgespeichert.

Nach der Datenaufnahme werden diese Rohdateien per Kabel oder Wifi auf einen Rechner transferiert, um diese dort weiter zu prozessieren. Typische Schritte sind unter anderem die Verbesserung des SLAMs (simultaneous localisation and mapping), Kolorierung, Streupunktfilterung, Registrierung von terrestrischen Scans und das Entfernen bewegter Objekte. Die weitere Bearbeitung der Punktwolke erfolgt in der Regel nicht in der Punktwolkenprozessierungssoftware, sondern in Drittlösungen, in der Regel GIS (Geoinformationssystem), CAD (Computer-aided design) oder BIM (Building Information Modelling) Softwarepaketen.

Aufgrund der Tatsache, dass die meisten Drittsoftwarelösungen nicht die proprietären Punktwolkenformate der Gerätehersteller lesen können, müssen daher die Dateien in ein Austauschformat gebracht werden, welches von der genutzten Drittsoftware verarbeitet werden kann.

Typische Punktwolkenaustauschformate im Gebrauch sind unter anderem XYZ, E57, LAS und LAZ.

Zu unterscheiden sind hier binär- und textbasierte Formate. Während XYZ komplett textbasiert ist und keine Metadaten beinhaltet, liegen bei den anderen genannten Formaten die Daten Binär ab und sie enthalten Metadaten in Binär- oder Textform.

Ein weiterer Unterschied zwischen den genannten Formaten ist, dass im LAZ-Format die Daten komprimiert abgespeichert sind und dadurch deutlich weniger Speicherplatz verbraucht.

3.1 Speicherplatzbedarf der Formate

Den verbrauchten Speicherplatz bei unkomprimierten Binärformaten kann man generalisiert abschätzen. In (1) sehen wir, dass die Dateigröße sich aus der Summe der enthaltenen Attribute sowie Punkte berechnet. In der Regel werden die reinen Koordinaten in 24 Byte und die Farbkanäle pro Kanal in einem Byte gespeichert. Weitere Attribute fallen je nach gewählter Datentypenart in die gleiche Größenordnung. Metainformationen sind in der Regel zu vernachlässigen, da sie nur einen Bruchteil der Dateigröße ausmachen.

Im Gegensatz dazu kann bei textbasierten Formaten kein genauer Speicherplatzverbrauch berechnet werden. Beim XYZ-Format liegt dies daran, dass der Speicherplatzverbrauch nicht nur mit der Punktzahl und den vorhandenen Attributen skaliert, sondern auch mit dem gewählten Koordinatensystem sowie der gespeicherten Punktgenauigkeit. Dies liegt daran, dass jedes Zeichen im Textformat ein Byte groß ist. Das heißt, bei einem globalen Projekt ist in jedem Punkt die Translation vom lokalen in das globale Projekt als zusätzliche Textzeichen enthalten.

$$D = \sum(xyz + R + RGB) \tag{1}$$

wobei D = Dateigröße, xyz = Punktposition, R = Reflektivität, RGB = Farbinformation

In Abb. 1 sieht man die Dateigröße der bereits erwähnten Formate, sowie zwei Binärformate ohne Metadaten. Dabei fällt auf, dass eine Änderung der genutzten Datentypen einen starken Einfluss auf die Dateigröße hat. Einen deutlich größeren Einfluss hat jedoch die Kompression, wie bei dem kaum erkennbaren Balken vom LAZ-Format ersichtlich.

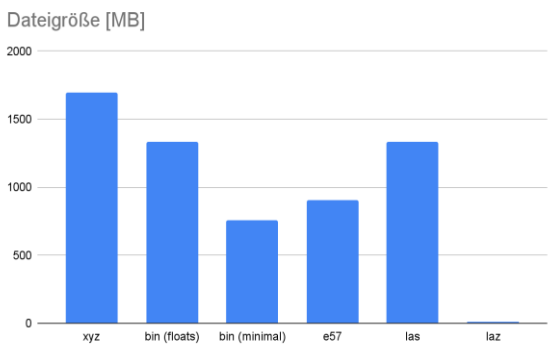

Abbildung 1 Dateigröße identischer Punktwolke mit 49,8 Mio. Punkten in diversen Austauschformaten

3.2 Einlesegeschwindigkeit der Formate

Die Einlesegeschwindigkeit ergibt sich aus zwei Schritten: Das Lesen sowie die Interpretation der Daten. Die theoretisch typischen Lesegeschwindigkeiten von Speichermedien findet man in Tabelle 1. Zu beachten ist, dass sich je nach Hardware sich die Geschwindigkeit um ca. Faktor zehn unterscheidet. Die resultierende Geschwindigkeit bis zu den bereitgestellten Daten hängt somit vom Speichermedium (interne SSD, externe Festplatten, Server über Internet, …), als auch von der Formatspezifikation selbst ab.

Die Interpretation der Daten kann auch aus mehreren Schritten bestehen. Dekomprimierung, Dequantisierung SCHÜTZ (2016, S. 59), Wertebereichadaption und Koordinatentransformation. Je nach eingesetzter Methode oder Algorithmen können diese Schritte unterschiedlich rechenintensiv sein.

Heutige Dekompressionsverfahren sind in der Regel schneller als die Lesegeschwindigkeit einer internen SSD-Festplatte. In Kombination mit den anderen Interpretationsschritten kann die Geschwindigkeit der Interpretation der Daten aber langsamer als die Lesegeschwindigkeit sein. Als Beispiel sei das Parsen und Umwandeln von Text in Binärdaten genannt, das deutlich langsamer als die typische Lesegeschwindigkeit einer SSD ist.

Die Lese-, sowie die Interpretationsschritte können alle parallel erfolgen. Es ist also nicht notwendig, zuerst die gesamte Datei zu lesen und dann zu interpretieren. Somit ergibt sich eine typische Flaschenhalssituation und der begrenzende Faktor kann klar determiniert werden.

Da die Lesegeschwindigkeit einer harten Grenze des gewählten Speichermediums unterliegt, muss man auf der anderen Seite die Anzahl und Rechenintensität der Interpretationsschritte beachten. Bei einer hohen Lesegeschwindigkeit des Mediums, wie bei einer SSD, kann man nicht viele Interpreationsschritte durchführen, bevor diese der begrenzende Faktor werden. Auf der anderen Seite ist beim Download von Daten aus dem Internet, der begrenzende Faktor stets die Lese- bzw. Downloadgeschwindigkeit und niemals die Interpretation. Im Falle von Daten aus dem Internet ist es also deutlich besser, kleinere Datenmengen lesen zu müssen und diese dafür in das benötigte Format zu interpretieren.

Tabelle 1 Typische Lesegeschwindigkeit von Daten

Typ	Geschwindigkeit GB/s
SSD	3,5
HDD	0,2
Externe Festplatte	0,06
250 MBit Internet	0,003

In Abb. 2 sieht man die komplette Einlesedauer der genannten Formate. Zu beachten ist hierbei, dass die Ergebnisse nicht direkt miteinander vergleichbar sind, da E57, LAS und LAZ über externe Bibliotheken, die restlichen über Selbstimplementation durchgeführt wurden. Des Weiteren ist die Lesegeschwindigkeit bei der Selbstimplementation nicht optimal, da nur einzelne Byte anstatt größerer Blöcke gelesen wurden.

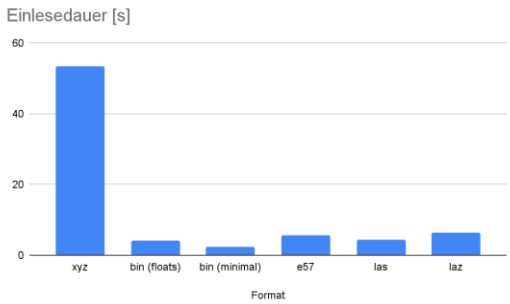

Abbildung 2 Einlesedauer identischer Punktwolke mit 49,8 Mio. Punkten in diversen Austauschformaten von SSD

4 Grafik APIs

Eine API (Application Programming Interface) ist ein Verbindungsstück zwischen verschiedener Hardware und Software oder zwischen Programmen. Damit man nicht für jede verfügbare Hardware Programmcode erstellen muss, benutzt man APIs. Dies gilt auch für Grafikkarten. Dort sind die bekanntesten APIs OpenGL, Vulkan, DirectX und Metal. Während DirectX und Metal proprietär nur unter Windows bzw. MacOS laufen, unterstützt OpenGL und Vulkan alle herkömmlichen Betriebssysteme.

Die verschiedenen APIs haben seit den Jahren ihrer Entstehung diverse Umstellungen in ihren Programmierkonzepten und -mustern. Das derzeit vorherrschende Vorgehen ist das Aufbauen diverser Pipelines. Man kann sich das wie Fließbandarbeit vorstellen, wo an verschiedenen Stationen die Daten spezialisiert bearbeitet und dann weitergegeben werden.

5 Visualisierung

Nachdem nun der Arbeitsablauf vom Laden und Interpretieren der Punktwolken diskutiert wurde, kommen wir zur eigentlichen Visualisierung.

Wie in Kapitel 2 gezeigt wurde, kann die gesamte Punktwolke nicht in Echtzeit von einem Medium in den Arbeitsspeicher geladen werden. Des Weiteren ist es ab einer gewissen Projektgröße auch nicht mehr möglich, das Projekt im Arbeitsspeicher und noch weniger im VRAM der Grafikkarte zu halten. Zusätzlich muss bei einem Szenenwechsel (Bewegung der Kamera), das visualisierte Bild möglichst schnell wieder aufgebaut werden, präferiert in Millisekunden oder schneller.

Übliche Ansätze um diese Limitierungen zu umgehen sind das Erstellen diverser Detailgrade (Level of Detail, LOD), mit Octree oder kd-tree Aufteilungen. Ein detaillierter Überblick über räumliche Datenstrukturen findet sich in SAMET (2006). Generell ist es wichtig, dass sichtbare Regionen mit einer hohen Punktanzahl visualisiert werden, während nicht sichtbare oder verschattete Regionen nicht visualisiert werden müssen. Um möglichst wenig Punkte pro Region für das beste Visualisierungsergebnis anzeigen zu müssen, sollte man bei der Punkteverteilung der einzelnen Regionen einen geeigneten Ansatz wählen. Wie von SCHÜTZ (2016, S. 17) gezeigt, ist eine geeignete, jedoch rechenintensive Verteilung Possion-Disk Subsampling. Dabei wird die Distanz zwischen allen Punkten minimiert und somit Löcher vermieden, während gleichzeitig die benötigte Punktanzahl minimiert wird.

Selbst mit dem Aufbau von LODs muss man die einzelnen Regionen schnell und effizient durch die Visualisierungspipeline schicken. Der naive und bisher übliche und für die Hardware ausgelegte Weg, GPU native Punktprimitive durch Rasterizer und Fragment Shader zu visualisieren, wird zunehmend durch Compute Shader ersetzt (SCHÜTZ et al., 2021).

6 Fazit & Ausblick

Die derzeit im Markt vorherrschenden Austauschformate für Punktwolken sind eine nicht zufriedenstellende Lösung. Weiterentwicklungen in der Kompressionstechnologie und das Verschieben von Anwendungen in die Cloud machen die Neuentwicklung offener Austauschformate notwendig. Dabei ist die größte Hürde nicht der Forschungsstand oder die Implementation eines Standards, sondern das Erreichen der Marktdurchdringung.

Visualisierung von vielen Daten ist ein altes und gut erforschtes Problem. Jedoch sind weiterhin Techniken zur effizienten Punktwolkenvisualisierung zu entschlüsseln, vor allem im Hinblick auf größer werdende Projekte.

Literaturverzeichnis

Schütz, M. (2016). Potree: Rendering Large Point Clouds in Web Browsers.

Schütz, M., Kerbl, B. & Wimmer, M. (2021), Rendering Point Clouds with Compute Shaders and Vertex Order Optimization. Computer Graphics Forum, 40: 115-126. https://doi.org/10.1111/cgf.14345

Samet, H. (2006). Foundations of multidimensional and metric data structures. Morgan Kaufmann.