Visualisierungstechniken zur Analyse und Interpretation von 3D-Punktwolken

Zusammenfassung

3D-Punktwolken sind riesige ungeordnete Punktdatenmengen, die aus terrestrischen oder luftgestützten Scans generiert werden. So entstehen diskrete Oberflächenstrukturen, die mit weiteren Attributen wie Farbe angereichert werden können. Die echtzeitfähige Visualisierung dieser massiven Daten benötigt Out-of-Core-Verfahren und geeignete Renderingtechniken. Neben der interaktiven Visualisierung und Exploration von 3D-Punktwolken in einer Szene können andere Visualisierungstechniken die Arbeit mit 3D-Punktwolken erleichtern. Diese Arbeit stellt hierfür zwei Visualisierungstechniken für die Analyse und Interpretation der Daten vor: Bei der Betrachtung von Stadtgebieten oder ganzen Städten kann der Nutzer schnell die Übersicht verlieren. Für die verbesserte Orientierung wird die Integration einer Minimap in eine bestehende Visualisierungskomponente für 3D-Punktwolken mithilfe von *OpenStreetMap* vorgestellt.

Für eine exakte Interpretation kann es jedoch nützlich sein, Bereiche der 3D-Punktwolke in weiteren Ansichten als innerhalb einer gerenderten Szene zu visualisieren. Die hierfür entwickelte Technik ist die Erstellung von Profilansichten durch Querschnittsbildung. Der Nutzer erhält die Möglichkeit, mehrere Schnitte festzulegen, die anschließend in einer einheitlichen Darstellung visualisiert werden. Durch die Reduktion der Verdeckung kann der bestehende Platz besser für die Visualisierung relevanter Daten genutzt werden.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Durch Laserscans von Städten, Landschaften oder einzelnen Gebäuden entstehen 3D-Punktwolken – eine massive Menge von unorganisierten Punktinformationen. Diese Aufnahmen sind mithilfe von luftgestützten oder terrestrischen Scannern schnell und kostengünstig realisierbar. Aus den erfassten 3D-Punktwolken werden üblicherweise Dreiecksmodelle generiert, was jedoch zeitaufwendig ist. Die mit der Generierung von Dreiecksnetzen verbundene Approximation ist darüber hinaus nicht in jedem Fall erwünscht. Durch direkte Nutzung der 3D-Punktwolken für Analysen und zur Visualisierung kann näher an den Realdaten gearbeitet werden.

Bei der Verwendung von Punkten als Visualisierungsprimitiv entstehen im Gegensatz zu Dreiecksmodellen keine geschlossenen Oberflächen. Dieser Effekt kann jedoch durch Renderingtechniken mit an die Punktdichte angepassten Punktgrößen sowie durch Geometrie pro Punkt mit unterschiedlicher Form und Ausrichtung erreicht werden. Diese Renderingtechniken können des Weiteren für Gruppen von Punkten abhängig von Kontext und Semantik gewählt werden.

Am Lehrstuhl für *Computergrafische Systeme* wurde ein Framework zur Analyse, Prozessierung und Visualisierung der 3D-Punktwolken entwickelt. Die Verarbeitung mithilfe von räumlichen Datenstrukturen ermöglicht dabei interaktive Echtzeitvisualisierung der 3D-Punktwolken. Im Rahmen unserer Projektarbeit haben wir dieses Framework zunächst um einen Webdienst erweitert, über den die Vorverarbeitung zentral auf einem leistungsfähigen Server durchgeführt werden kann. Zudem haben wir die Komponenten zur Prozessierung optimiert, um die Leistungsfähigkeit der Hardware besser auszunutzen und von Multi-Core-Systemen zu profitieren.

Ein weiterer Schwerpunkt unserer Arbeit lag auf der Visualisierung der prozessierten Daten. Die vorhandenen Renderingtechniken wurden ergänzt und unter Nutzung neuerer Hardwarefeatures optimiert. Des Weiteren wurden Interaktions- und Analysetechniken entwickelt, um die 3D-Punktwolken umfassend explorieren zu können.

Teil der Arbeit war außerdem das Refactoring des bestehenden Codes, um zukünftige Entwicklung und Erweiterung des Frameworks zu erleichtern. Die Einführung von Tests stellt dabei die Korrektheit der Funktion von Kernkomponenten sicher.

1.1 Einordnung in den Lehrstuhlkontext

Der in der Projektarbeit entwickelte Prozessierungserver nutzt die bestehende Komponente *PCLib* (Abbildung 1.1). In der *PCLib* sind die Kernkomponenten zur Analyse und Prozessierung von 3D-Punktwolken und anderen Datenformaten implementiert. Das *PCTool* setzt darauf aufbauend Analyse- und Verarbeitungstechniken um. Unter anderem für grafische Nutzeroberflächen und vereinfachte nebenläufige Implementierungen wird in den Lehrstuhlkomponenten *Qt5* genutzt.

Zur Visualisierung von 3D-Punktwolken und einer Reihe anderer Datenformate wurde die Anwendung *PCViewerOsg* entwickelt, welche *OpenSceneGraph* als Rendering-System nutzt. In unserer Projektarbeit entwickelten wir einen auf die Visualisierung von 3D-Punktwolken spezialisierten Viewer (*PCViewerGlow*). Bei diesem nutzen wir die am Lehrstuhl entwickelte Bibliothek *glow*, eine leichtgewichtige, objektorientierte Schnittstelle für *OpenGL*. Dies ermöglicht die Entwicklung eines flexibel erweiterbaren Renderingsystems bei Nutzung neuerer Hardwaremöglichkeiten. Die ebenfalls am Lehrstuhl entwickelte Bibliothek *libzeug* ermöglicht eine schnelle Umsetzung von grafischen Nutzerschnittstellen zur Konfiguration der Anwendung. Die Komponenten des Lehrstuhlframeworks sind auf den Betriebssystemen Windows, Linux und Mac OS X einsetzbar.

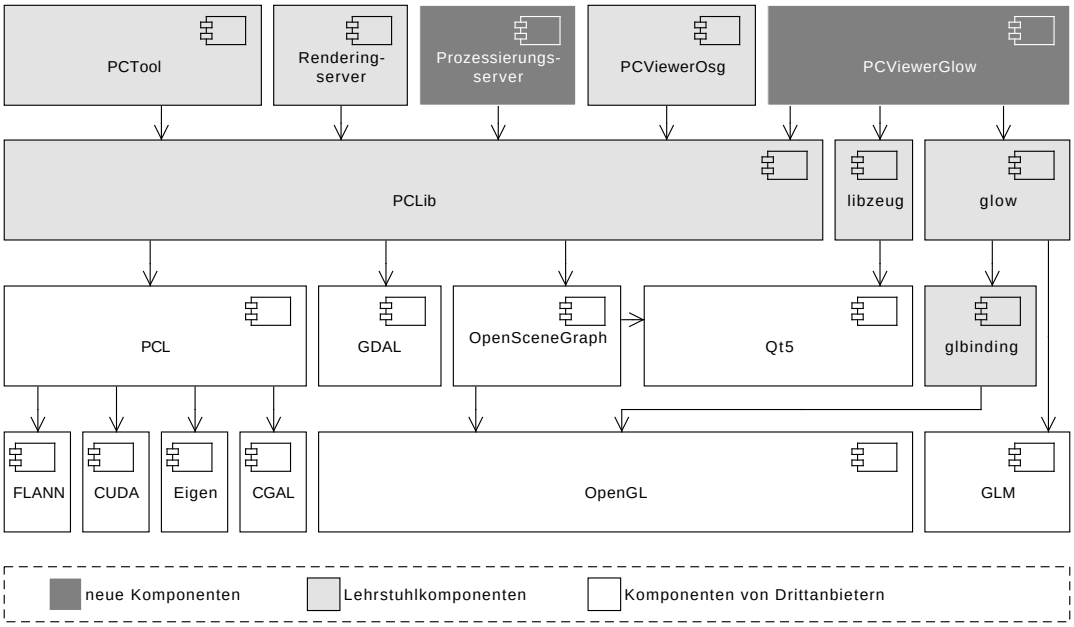


Abbildung 1.1: *Abhängigkeiten der Lehrstuhlkomponenten untereinander und von Bibliotheken von Drittanbietern.*

1.2 Verarbeitung und Visualisierung von 3D-Punktwolken

1.2.1 Räumliche Datenstrukturen

Erfasste 3D-Punktwolken bestehen in der Regel aus mehreren Milliarden Punkten. Für eine Visualisierung sind davon, abhängig von dem Betrachtungsstandort und der Blickrichtung, jeweils nur wenige Millionen Punkte tatsächlich relevant. Da die Punktdaten mehrere Gigabyte bis Terabyte umfassen, können sie nicht vollständig in den Haupt- oder Grafikspeicher geladen und müssen daher vorselektiert werden. Dies ist bei den ungeordneten massiven Punktdaten jedoch in Echtzeit nicht direkt umsetzbar. Daher werden die Daten zunächst mithilfe einer räumlichen Datenstruktur geordnet, wodurch effizienter Zugriff auf Teilbereiche möglich wird. Im gegebenen Anwendungsfall wird der Raum dreidimensional in überschneidungsfreie Teilbereiche aufgetrennt, welche dann wiederum rekursiv geteilt werden. Da die Punktdichten in verschiedenen Bereichen der 3D-Punktwolken sehr unterschiedlich sein können, bieten sich KD-Trees als Datenstruktur an. In diesen binären Suchbäumen werden die Daten zu gleichen Teilen auf benachbarte Teilbäume aufgetrennt, wodurch ein möglichst flacher Baum entsteht. Eine Suche in diesem Baum hat logarithmische Komplexität.

1.2.2 Serviceorientierte Prozessierung von 3D-Punktwolken

Die Funktionen der PCLib waren bisher nur über Desktopanwendungen wie das *PCTool* verfügbar. Die Prozessierungen sind jedoch rechenintensiv und benötigen auf Desktopsystemen viel Zeit. Daher entwickelten wir in der Projektarbeit einen Webdienst, mit dem die Prozessierungen zentral auf einem leistungsstarkem Server ausführbar sind. Die 3D-Punktwolken können über standardisierte Netzwerkprotokolle auf den Server übertragen werden, um dort über eine Webseite Prozessierungsaufträge zu erstellen. Nach der erfolgreichen Verarbeitung werden Nutzer per Mail informiert und können die prozessierten Daten vom Server abrufen.

1.2.3 Visualisierung und Rendering

Für die Echtzeitvisualisierung von 3D-Punktwolken verwenden wir die in Abschnitt 1.2.1 genannten KD-Trees. Die definierte räumliche Ausdehnung der Knoten im Baum ermöglicht die effiziente Priorisierung von Bereichen der 3D-Punktwolke. Darauf aufbauend wird durch die Implementierung eines Out-of-Core-Verfahrens (Gobbetti & Marton, 2004) sichergestellt, dass alle relevanten Punkte zum Rendern auf der Grafikkarte verfügbar sind. Vorhandener Haupt- und Grafikspeicher wird als Cache genutzt, um gerade benötigte Daten möglichst selten von langsamen Datenspeichern wie Festplatten oder Netzlaufwerken laden zu müssen. Die Interaktivität der Anwendung wird beim Betrachten der 3D-Punktwolken sichergestellt, indem die Anzahl der pro Bild gerenderten Punkte nach Bedarf limitiert wird.

Die für das Rendering ausgewählten Punkte können mithilfe verschiedener Renderingtechniken visualisiert werden. Dabei gibt es sowohl bildraum- als auch objektraumbasierte

Verfahren. Eines davon ist beispielsweise das Splatting (Botsch et al., 2004): Dabei werden die Punkte als Kreisscheiben oder Ellipsen – anhand ihrer Normalen ausgerichtet – im Objektraum gerendert. Die Normalen können über eine Approximation der Oberfläche aus der unmittelbaren Umgebung jedes Punktes generiert werden.

1.2.4 Interaktion

Neben der Visualisierung sind Techniken für die Interaktion und Analyse (Wand et al., 2007) ein weiterer wesentlicher Bestandteil der Arbeit mit 3D-Punktwolken: Bei schweren Verkehrsunfällen werden seit 2011 3D-Scans der Unfallstellen von der Berliner Polizei[1] eingesetzt, um ein schnelles Räumen der Unfallstelle ohne Verlust der Beweismittel zu ermöglichen. Im Nachhinein können so weiterhin Messungen zur Rekonstruktion des Unfalls durchgeführt werden. Hierfür sind Hilfsmittel zum exakten Messen von Abständen und Flächen in 3D-Punktwolken erforderlich. Ein weiteres Anwendungsgebiet ist die Rekonstruktion und digitale Erfassung von Gebäuden: Stark verfallene oder historisch wichtige Gebäude können gescannt und von Experten interaktiv exploriert werden. Neben Messungen sind auch multiperspektive Ansichten (Yu et al., 2008) und Querschnittsansichten Werkzeuge für die Exploration und Analyse von 3D-Punktwolken.

1.3 Einordnung in den Projektkontext

In dieser Bachelorarbeit werden geeignete Visualisierungstechniken zur Analyse und Interpretation von 3D-Punktwolken innerhalb des Gesamtsystems beschrieben. Die umgesetzten Techniken umfassen ein Minimapwidget für die verbesserte Interaktion und Navigation sowie ein Tool für die Erstellung von Querschnittsansichten.

1.4 Aufbau der Arbeit

Diese Arbeit ist in sechs Kapitel gegliedert. Das nächste Kapitel stellt die verwandten Arbeiten vor. Dabei werden – wie in allen weiteren Kapiteln – die Minimap und der Querschnitt getrennt betrachtet. Nach der Analyse des bestehenden Systems folgen die notwendigen Konzepte für die Visualisierungstechniken und Anforderungen im dritten Kapitel. Der vierte Abschnitt beschäftigt sich mit der Implementierung der Konzepte im bestehenden Programmrahmen. Nach der Einordnung des Minimapwidgets in die bestehende Architektur erfolgt die Erläuterung der Interaktion und die Anzeige des Kartenwidgets sowie die Umsetzung von Navigations- und Interkationshilfen. Die Querschnittserstellung und -anzeige wird anhand der Auswahl der Achsen, Selektion der Punkte und der Visualisierung dargestellt. Das fünfte Kapitel evaluiert die Ergebnisse auf Grundlage der in Kapitel drei aufgestellten Anforderungen und Konzepte. Ein Fazit schließt diese Bachelorarbeit ab.

[1] http://www.berlin.de/polizei/aufgaben/verkehrssicherheit/verkehrsunfallaufnahme (26. Juni 2014)

Kapitel 2

Verwandte Arbeiten

Riesige Datenmengen erfordern geeignete Visualisierungstechniken, die es dem Nutzer ermöglichen den Überblick zu behalten. Dabei soll es auch möglich sein, die Aufmerksamkeit des Nutzers auf bestimmte Bereiche zu fokussieren. Die Minimap und das Querschnittstool sind mögliche Umsetzungen dieses Forschungsgebiets – der Fokus- und Kontextvisualisierung. Die effiziente Bereitstellung von Punkten aus massiven 3D-Punktwolken erfordert zudem geeignete Datenstrukturen. Nachfolgend werden einige relevante Arbeiten aus diesen Bereichen vorgestellt und in den Kontext des Projekts eingeordnet.

2.1 Fokus- und Kontextvisualisierung

Bei der Arbeit mit massiven 3D-Punktwolken ist es für den Nutzer von Bedeutung den Überblick über die geladenen Daten zu behalten und gleichzeitig auf bestimmte Bereiche zu fokussieren. Hierfür werden sinnvolle Konzepte im Bereich der Fokus- und Kontextvisualisierung erforscht – einen Überblick über bestehende Techniken geben Cockburn et al. (2008). Möglichkeiten, die fokussierten Ansichten in einen Kontext zu setzen, bieten unter anderem *3D Lenses* oder *multiperspektive Ansichten*. Glander et al. (2008), Glander et al. (2011) und Pasewaldt et al. (2011) bieten beispielhafte Umsetzungen, welche die Visualisierung von Fokus und Kontext innerhalb einer Szene zeigen.

Um eine 3D-Punktwolke in einen Fokus zu setzen, können einige Teilbereiche detaillierter als andere dargestellt werden. Soll jedoch die gesamte 3D-Punktwolke kontextuell eingeordnet werden, sind Hilfsmittel wie Karten eine geeignete Umsetzungsmöglichkeit. Kartenmaterial kann innerhalb einer gerenderten Szene über *Transparency Labels* oder *Glowing Roads* visualisiert werden (Vaaraniemi et al., 2013). Die gerenderten 3D-Punktwolken enthalten jedoch bereits Techniken für die Fokussierung auf unterschiedliche semantische Klassen (Discher, 2013). Werden die Visualisierung von semantischen Klassen, Originalfarbe und Kontext innerhalb einer Szene gerendert, besteht die Gefahr, den Nutzer zu stark von den eigentlichen Informationen abzulenken. Die Kontextvisualisierung erfolgt daher in einer gesonderten Kartenansicht – wie in *Google Maps*[1] oder *OpenStreetMap*[2]. Bei Querschnittsansichten werden multiperspektive Ansichten insbesondere für die Fokussierung auf die geschnittenen Bereiche der 3D-Punktwolke genutzt. Die

[1] www.google.de/maps
[2] www.openstreetmap.org

Visualisierung der Schnittebenen im Viewer erhält dabei die Kontextinformationenen.

2.2 Datenstrukturen

Das Selektieren der Punkte des Querschnitts erfolgt aus den Daten massiver 3D-Punktwolken, für eine effiziente Verarbeitung wird daher eine räumliche Datenstruktur benötigt. Passende räumliche Datenstrukturen sind unter anderem R-Trees (Gong et al., 2012) und KD-Trees (Bentley, 1975). R-Trees sind indexbasierte, räumliche Datenstrukturen, welche die enthaltenen Objekte in minimale Boundingboxen aufteilen und diese dann hierarchisch schachteln. Ein KD-Tree ist ein binärer Baum, der die Räume entlang der längsten Achse in zwei Unterräume mit gleicher Punktanzahl teilt.

Da die Berechnung des Querschnitts auf zweidimensionalen Vektoren arbeitet, könnte auch ein Quadtree (Finkel & Bentley, 1974) genutzt werden. Quadtrees teilen zweidimensionale Räme in exakt vier gleich große Unterräume und erstellen ein gleichmäßiges Raster. Die dreidimensionale Implementierung von Quadtrees sind Octrees (Meagher, 1987). Linke (2014) zeigt, warum KD-Trees anstelle von Quadtrees im aktuellen Viewer verwendet werden. Da für das Rendering bereits KD-Trees für die Sturkturierung der Daten vorhanden sind, können diese für das Prozessieren der Daten zur Querschnittserstellung ebenfalls genutzt werden. Eine Implementierung einer weiteren Datenstruktur ist daher nicht sinnvoll.

Kapitel 3

Konzept

Im Folgenden werden die Konzepte und Anforderungen für die Integration einer Minimap und eines Querschnittstools vorgestellt. Zunächst erfolgt die Analyse, ob und inwiefern sich diese genannten Visualisierungstechniken für das bestehende System eignen und welche Voraussetzungen bereits erfüllt sind.

3.1 Analyse des bestehenden Systems

Das bestehende System verfügte initial über keine Funktionalitäten für die gesonderte Darstellung einer Minimap oder eines Querschnitts.

Minimap Bei der Visualisierung von 3D-Punktwolken kann der Nutzer in beiden Viewern mithilfe einer Terrain-Navigation[1] oder einer Flight-Navigation[2] durch die Szene der geladenen 3D-Punktwolken navigieren. Beim Betrachten mehrerer 3D-Punktwolken kann der Nutzer jedoch die Orientierung verlieren: Neben der Zuordnung der visualisierten Bereiche zu den einzelnen 3D-Punktwolken und der Lokalisation in Weltkoordinaten fehlt dem Nutzer die Information, wo sich die Kamera während der Navigation befindet. Überschneidungsbereiche der 3D-Punktwolken sind bei gleicher Renderingtechnik gegebenenfalls aus der Visualisierung nicht erkennbar.

Im Prozessierungsservice fallen ähnliche Probleme auf: Bei der Auswahl der zu prozessierenden 3D-Punktwolken muss der Nutzer sich anhand der Dateinamen und Metadaten – wie beispielsweise Boundingboxen – zwischen sämtlichen hochgeladenen Dateien des Projekts entscheiden. Die zusätzliche Information, wo sich die 3D-Punktwolken räumlich befinden, können für die Auswahl von großem Nutzen sein.

Diese Navigations- und Orientierungsprobleme können insbesondere für luftgestützte 3D-Punktwolken gelöst werden. Das Mapping zwischen 3D-Punktwolkendaten und realen Weltkoordinaten lässt sich bei luftgestützten 3D-Punktwolken leicht generieren und kann in eine Karte übertragen werden (Abbildung 3.1). Im bestehenden System wurde dieses Mapping bisher jedoch nicht visualisiert.

In 3D-Punktwolken aus Befliegungsdaten werden für die Interpretation der Koordinaten EPSG-Codes gespeichert: ein System weltweit eindeutiger 4- bis 5-stelliger

[1] Navigation basierend auf der Rotation um den Schnittpunkt des Sichtvektors mit der xy-Ebene
[2] Navigation basierend auf der Rotation um das Projektionszentrum

(a) *Luftgestützte 3D-Punktwolke.* (b) *Kartenansicht.*

Abbildung 3.1: *Vergleich der Visualisierung einer luftgestützten 3D-Punktwolke im Viewer mit der zugehörigen Darstellung in einer Karte.*

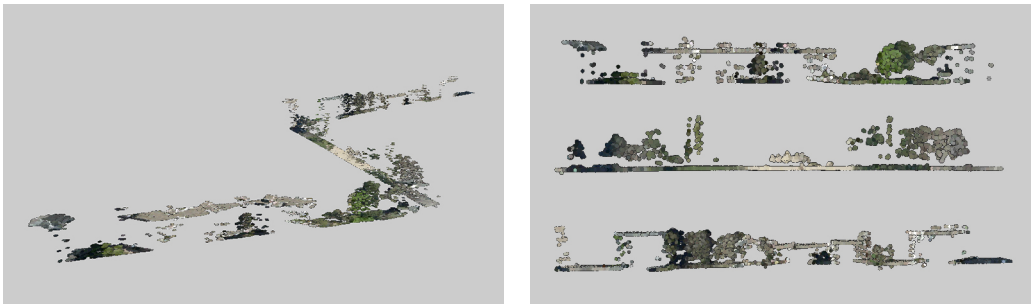
Schlüsselnummern für Koordinatenbezugssysteme, entwickelt von der European Petroleum Survey Group. Diese können für die korrekte Zuordnung der Koordinaten der 3D-Punktwolke und der Karte genutzt werden. Terrestrische 3D-Punktwolken enthalten hingegen meist keinen EPSG-Code und können ohne zusätzliche Vorverarbeitungsschritte nicht eindeutig auf Kartenmaterial übertragen werden.

Querschnittstool Initial gab es für den Nutzer ausschließlich die Möglichkeit, 3D-Punktwolken in einer gerenderten 3D-Szene zu betrachten. Für eine exakte Analyse oder die Interpretation der 3D-Punktwolken kann es jedoch von Nutzen sein, die räumlichen Daten in multiperspektiven Ansichten zu visualisieren. Durch Reduktion der Verdeckung kann der zur Verfügung stehende Platz besser für die Visualisierung der relevanten Daten genutzt werden. Eine Möglichkeit zur Erstellung einer multiperspektiven Ansicht von 3D-Punktwolken ist die Querschnittsbildung entlang mehrerer Schnitte. Diese Querschnitte können in einer einheitlichen Darstellung visualisiert werden (Abbildung 3.2).

Es war bisher nicht möglich, Querschnitte und Profilansichten zu erstellen. Für das Selektieren von Punkten sind die zum Out-of-Core-Rendering erstellten räumlichen Datenstrukturen jedoch ein geeigneter Ausgangspunkt.

3.2 Minimap zur verbesserten Navigation und Orientierung

Die Minimap soll die Navigation und Orientierung innerhalb der Szene erleichtern. Die Visualisierung muss sowohl in beiden Viewern als auch im Webfrontend des Prozessierungsservers ähnlich nutzbar sein. Dabei sind jedoch in der Kartenanzeige des Prozessierungsservers nicht alle Funktionalitäten sinnvoll, da die Karte nur der Visualisierung der Lage der selektierten 3D-Punktwolken dient (Bäumer, 2014). Die Integration der Minimap erfolgt als zusätzliches Widget in die jeweilige grafische Benutzeroberfläche,



(a) *Originale Punktwolke.* (b) *Multiperspektive Ansicht.*

Abbildung 3.2: *Multiperspektive Ansicht einer schmalen 3D-Punktwolke.*

um sie zu jedem Zeitpunkt erreichbar zu machen und sie als Navigationshilfe nutzen zu können.

Kartendienste Für die Umsetzung der Minimap ist die Auswahl des Kartendiensts eine grundlegende Entscheidung. Für die interaktive Darstellung von Kartenmaterial stehen insbesondere folgende Dienste zur Auswahl:

- *Google Maps* ist einer der bekanntestes Kartendienste. Die zur Verfügung stehende API ist jedoch nur unter der Annhame eines umfassenden Lizenzvertrages nutzbar.
- *Bing Maps* wurde von Microsoft entwickelt. Es stehen sieben verschieden APIs für die Nutzung zur Verfügung.
- *Yahoo Maps* ist ein von Yahoo entwickelter Internet-Kartendienst mit einer Flash-, Ajax- und *Simple*-API. Die Weiterentwicklung wurde jedoch im September 2011 eingestellt[3].
- Der Online-Kartendienst *Yandex.Maps* des russischen Unternehmens Yandex, welcher über drei verschiedene APIs eingebunden werden kann. Die zur Verfügung gestellten Daten umfassen Russland, Weißrussland, Kasachstan und die Ukraine. Weitere Daten werden über den externen Anbieter *NAVTEQ* zugänglich gemacht.
- *OpenStreetMap* ist ein weiterer Kartendienst, der im Gegensatz zu den meisten Diensten seine Karten frei verfügbar anbietet. OpenStreetMap stellt darüber hinaus auch die Geodaten zur Verfügung. Bei der Verwendung von OpenStreetMap-Daten muss eine spezielle Namensnennung erfolgen.

Die Umsetzung der Minimap erfolgt in diesem Projekt über OpenStreetMap, wobei die Darstellung der Daten webbasiert über *OpenLayers*[4] erfolgt.

[3] Quelle: Wikipedia: Yahoo Maps, besucht am 9. Juni 2013
[4] http://openlayers.org

OpenStreetMap ist ein 2004 gegründetes Projekt zur Erstellung einer frei[5] verfügbaren Weltkarte. Die Daten umfassen unter anderem – wie andere Kartendienste – Straßen, Häuser, Flüsse, Wälder, Eisenbahnnetze aber auch semantische Informationen zu Veranstaltungsorten. Die Daten werden dabei von mehr als 1,6 Millionen[6] ehrenamtlichen Nutzern gesammelt, ausgewertet und gewartet. Zusätzlich nutzt OpenStreetMap andere freie Datenquellen, wie beispielsweise frei verfügbares Material von Landesämtern. Die durch OpenStreetMap zur Verfügugen gestellten Daten sind insbesondere in Gebieten mit großer Bevölkerungsdichte sehr aktuell und vollständig. Das über OpenStreetMap über Deutschland verfügbare Material übersteigt das Material kostenpflichtiger Hersteller um etwa 27 Prozent (Neis et al., 2012). OpenStreetMap stellt im Gegensatz zu anderen Kartendiensten nicht nur Kartenmaterial zur Verfügung, sondern auch die dahinter liegenden Geodaten, so können die Daten für Naviationsanwendungen oder auch eigene Kartendarstellung genutzt werden.

Interaktion Sofern sie einen gültigen EPSG-Code aufweisen, soll während der Betrachtung von 3D-Punktwolken in der Minimap der Bereich der globalen Boundingbox zu sehen sein, der die Boundningboxen aller geladenen 3D-Punktwolken zusammenfasst. Dieser Bereich soll beim Laden neuer 3D-Punktwolken dynamisch angepasst werden. Innerhalb der Minimap kann – wie in anderen Kartendiensten – frei navigiert werden: Dazu gehören das Verschieben der Karte und beliebiges Zoomen. Das Zoomlevel der Minimap ist dabei unabhängig vom Zoomlevel der 3D-Punktwolken-Visualisierung.

Die Navigation in den Viewern soll sich ebenfalls auf die Minimap auswirken. Bei der Verschiebung der Kamera wird auch das Zentrum der Karte verschoben. Ebenso kann der Nutzer in die Minimap klicken und so das Kamerazentrum versetzen.

Navigationshilfen Für eine verbesserte Orientierung innerhalb der Karte ist zusätzlich die Visualierung der einzelnen Boundingboxen und des Sichtfrustums geplant. Die Darstellung des Sichtfrustums soll sowohl die Kameraposition als auch die Blickrichtung und Blickwinkel verdeutlichen.

Weitere Anforderungen Unter Berücksichtigung aller gewünschten Features muss die Minimap performant in die Viewer integriert werden. Wenn 3D-Punktwolken geladen wurden, die keine EPSG-Codes enthalten, darf die Minimap kein undefiniertes Verhalten zeigen und muss erkennbar darstellen, dass die Metadaten der geladenen 3D-Punktwolken nicht interpretiert werden können.

Die Minimap des Prozessierungsservers weist eine geringere Funktionalität auf, da nur ein Teil benötigt wird. Hierfür ist vor allem die Darstellung der Boundingboxen neben der Kartenanzeige und der Navigation von Bedeutung.

[5] unter der *Open Database Licence (ODbL) 1.0* bzw. *Database Contents License (DbCL) 1.0*
[6] www.openstreetmap.org/stats/data_stats.html, Stand 10.7.2014

3.3 Querschnittsansichten für 3D-Punktwolken

Bei der Erstellung von Querschnittsansichten soll für den Nutzer die Möglichkeit geschaffen werden, aus den bestehenden 3D-Punktwolken (Höhen-)Profilansichten für luftgestützte oder (Raum-)Querschnitte für terrestrische 3D-Punktwolken zu erstellen. Dafür sollen mehrere Schnittebenen[7] orthogonal zur xy-Ebene vom Nutzer ausgewählt werden, die zusammenhängen müssen und eine Schnittform bilden. Das Querschnittstool wird vorerst nur im neu entwickelten Viewer implementiert und erst zukünftig und bei Bedarf in den bereits vorhanden Viewer integriert. Das Tool soll vor der Portierung in einen anderen Viewer ausreichend entwickelt und getestet werden, darüber hinaus muss der genaue Funktionsumfang erst festgelegt werden.

Die Erstellung des Querschnitts soll sich aus folgenden Schritten zusammensetzen: die Konfiguration der Erstellungsparameter durch den Nutzer, die Visualisierung der Schnittebenen, das Selektieren der Punkte und die Visualierung des Ergebnisses.

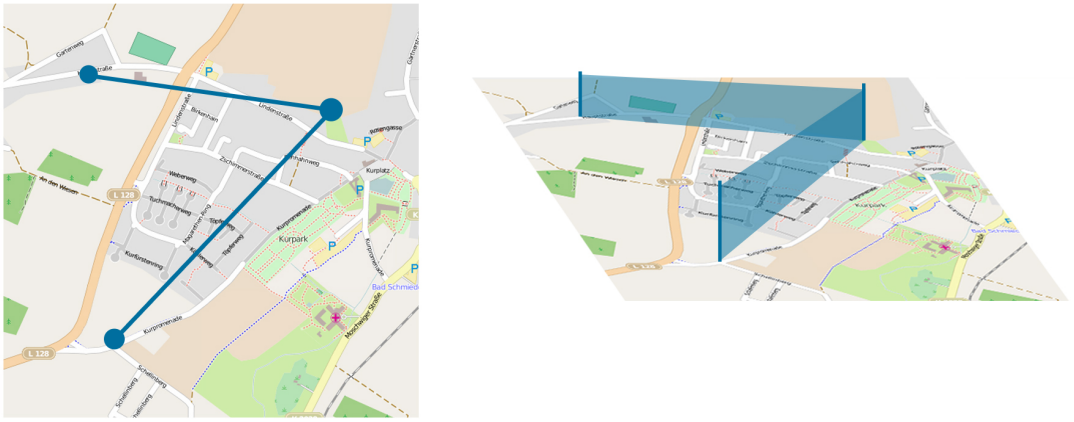
Konfiguration der Querschnittserstellung Die Erstellung und Visualisierung von Querschnittsansichten sollen durch den Nutzer konfigurierbar sein: Neben den Schnittebenen sollen sowohl die Breite des Korridors um die Ebene als auch die einzubeziehenden semantischen Klassen auswählbar sein (Jendruk, 2014). Wenn alle Konfigurationsparameter gesetzt sind, kann die Berechnung über die grafische Benutzeroberfläche initiiert werden.

Eine Möglichkeit der Schnittebenenauswahl ist die Projektion in eine zweidimensionale Ebene: Die Auswahl der Schnittebenen erfolgt über die Selektion einzelner xy-Positionen im Koordinatesystem des Viewers – im folgenden *Querschnittspunkte* genannt. Diese Querschnittspunkte definieren die Seitenbegrenzungen, die als Positionen orthogonaler Geraden zur xy-Ebene interpretiert werden und zwischen denen die Schnittebenen aufgespannt werden (Abbildung 3.3).

Das Setzen der Querschnittspunkte soll punktweise und entweder in der Minimap oder über Picking durch den Nutzer erfolgen. Während in der Minimap ein Doppelklick in der Karte einen Punkt hinzufügen kann, muss für das Picking ein zusätzlicher Modus aktiviert werden. Dieser Modus verhindert das versehentliche Hinzufügen von Querschnittspunkten während der Nutzung weiterer Picking-Modi (Eyssen, 2014). Bei falsch hinzugefügten Querschnittspunkten soll es die Möglichkeit geben, den zuletzt hinzugefügten Punkt oder alle Querschnittspunkte zu löschen.

Visualisierung der Schnittebenen Zur Hervorhebung der bereits hinzugefügten Schnittebenen ist die Visualisierung dieser Ebenen in der Minimap und im Viewer geplant. Insbesondere sollen die exakte Ausrichtung der Schnittebenen, die Breite der Korridore und eine Übersicht über die eingeschlossenen Punkte visualisiert werden. Die Korridore um die Schnittebenen könnten dabei durch transparente Quader dargestellt werden (Abbildung 3.4). Darüber hinaus soll der Nutzer die Möglichkeit haben, die Visualiserung der Schnittebenen zu deaktivieren.

[7] Eine Ebene, die mit einer 3D-Punktwolke oder einem Körper geschnitten wird, um die Punkte der 3D-Punktwolke bzw. des Körpers, die auf dieser Schnittebene liegen, zu betrachten. Bei Querschnitten mit 3D-Punktwolken muss ein gewisser Bereich um diese Schnittebene betrachtet werden, da die Dichte der 3D-Punktwolken stark variieren kann.



(a) *Karte mit ausgewählten Querschnittspunkten.*

(b) *Aufgespannte Schnittebenen.*

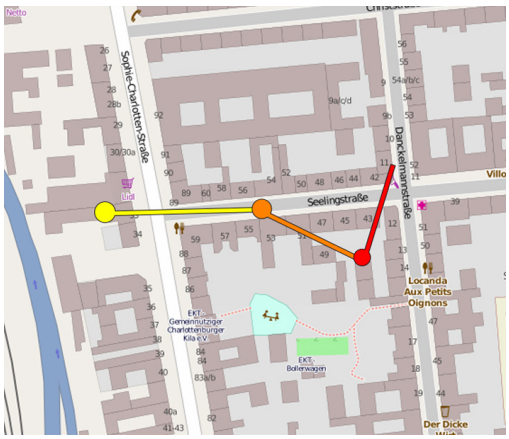
Abbildung 3.3: *Aufspannen der Schnittebenen.*

Selektion der Punkte Nach dem Starten der Querschnittsberechnung müssen die Punkte aus den aktiven 3D-Punktwolken extrahiert werden. Die Selektion erfolgt anhand des Abstands zu den Schnittebenen. Der maximale Abstand wird durch die vom Nutzer angegebene Korridorbreite bestimmt. Da die Schnittebenen orthognonal zur xy-Ebene sind, bietet sich die Berechnung der Abstände im Zweidimensionalen an. Die bestehenden räumlichen Datenstrukturen sollen für eine optimale Vorauswahl der Punkte genutzt werden.

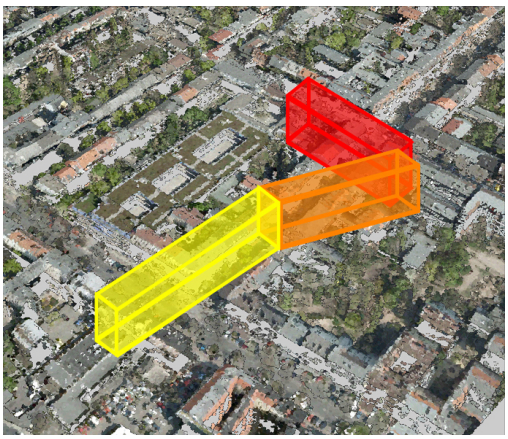
Visualisierung des Querschnitts Nachdem die Auswahl der Punkte erfolgt ist, müssen diese als Querschnitt visualisiert werden. Die Schnittebenen müssen für die Visualisierung zusammenhängend in eine Ebene transformiert und als ein einziger Korridor interpretiert werden. In Abbildung 3.4 ist die Entstehung des Korridors dargestellt, welcher – in einer Seitenansicht gerendert – die Querschnittsansicht bildet. Die Visualiserung soll in jeweils einem separaten Widget erfolgen, sodass mehrere Querschnittsansichten zur gleichen Zeit möglich sind.

Die Querschnittsansicht soll sämtliche Punkte des Querschnitts darstellen. Querschnittsansichten von luftgestützten und terrestrischen 3D-Punktwolken können in ihrem Seitenformat stark varrieren, daher ist es notwendig, sowohl eine Scroll- als auch eine Zoomfunktionalität zu implementieren. In dieser Ansicht soll ebenfalls die Entfernung der einzelnen Punkte zur Schnittebene visualisiert werden.

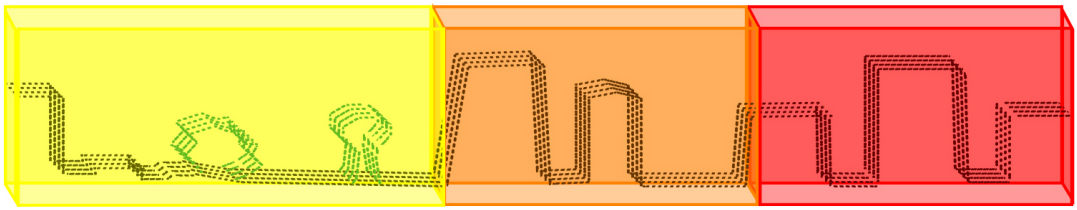
Zuordnung der Schnittebenen Um dem Nutzer die exakte Zuordnung der Schnittebenen in jeder Stufe der Querschnittsbildung zu erleichtern, soll ein einheitliches Farbmapping umgesetzt werden. Jede Schnittebene hat – abhängig von ihrer Erstellungsreihenfolge – eine klar zugeteilte Farbe. Diese Farbe wird in der Minimap und Schnittebenendarstellung im Viewer genutzt und bleibt bis zur Querschnittsansicht erhalten (Abbildung 3.4). Die Festlegung der Farben ist in Kapitel 4.2.2 beschrieben.



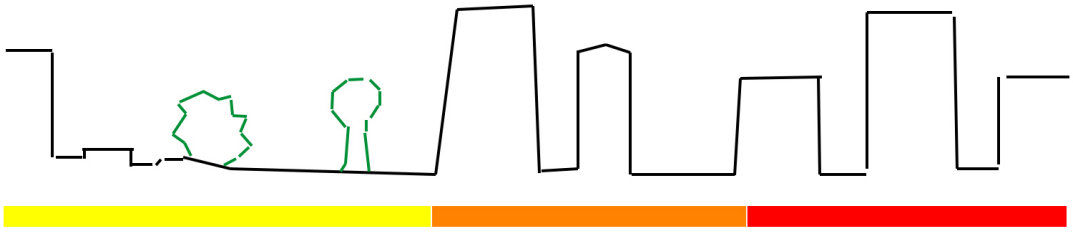
(a) *Viewer und Minimap mit gesetzten Querschnittspunkten.*



(b) *Visualisierung der Korridore im Viewer.*



(c) *Zusammenführung der Korridore.*



(d) *Visualisierung des Querschnitts als Seitenansicht.*

Abbildung 3.4: *Konzeptionelle Entstehung der Querschnittsansicht mit einheitlicher Schnittebenen-Farbgebung über die verschiedenen Ansichten hinweg.*

Kapitel 4

Implementierung

Der folgende Abschnitt beschreibt die Implementierung der in Abschnitt 3 vorgestellten Konzepte sowie die Eingliederung in die bestehende Architektur.

4.1 Minimapwidget

Die von OpenStreetMap zur Verfügung gestellten Daten werden für eine interaktive Minimap über OpenLayers gerendert und angezeigt. Diese Darstellung muss webbasiert in das bestehende System integriert werden. Zudem können die Interaktion und Visualisierung weiterer Daten über OpenLayers vorgenommen werden.

4.1.1 Visualisierung von OpenStreetMap-Daten mit OpenLayers

Für die interaktive Darstellung von OpenStreetMap-Daten wird OpenLayers genutzt. OpenLayers ist eine freie, unter der *2-clause BSD License* verfügbare JavaScript-Bibliothek zur webbasierten Visualisierung von Kartenmaterial ohne serverseitige Abhängigkeiten. Das Projekt gehört zur *Open Source Geospatial Foundation*. Über die Google Maps-ähnliche API können auch geschlossene Formate wie die Daten von *Virtual Earth* oder *Google Maps* angezeigt werden. Darüber hinaus besteht die Möglichkeit verschiedene Marker, Polygone und Linienzüge über verschiedene Layer in die Karte zu zeichnen.

4.1.2 Eingliederung in die bestehende Architektur

Die webbasierte Integration der Minimap wird über das *QtWebKit* von Qt5 realisiert. Das QtWebKit ermöglicht es, dynamische Webinhalte in einer Anwendung zu integrieren – Umgesetzt wird diese Funktionalität in der Klasse `QWebPage`, während `QWebView` diese Funktionalität als `QWidget` kapselt. Die Klasse `QWebFrame` repräsentiert Frames innerhalb einer Website (Abbildung 4.1) und kapselt die eigentlichen HTML- und JavaScript-Funktionalitäten.

Das QtWebKit enthält bereits eine JavaScript-Umgebung. Um die Rückkopplung der Navigation aus der Map heraus in den Viewer zu implementieren, muss eine weitere Technologie genutzt werden: Die *QtWebKit Bridge* erweitert das QtWebKit um den Zugriff auf native Objekte aus der JavaScript-Umgebung heraus. So kann der Zugriff

auf einzelne Objekte in der JavaScript-Umgebung ermöglicht werden. Die Technologie basiert dabei auf Qts *Signals and Slots*-Mechanismus[1].

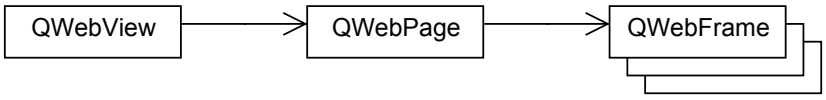


Abbildung 4.1: *Struktur eines QWebViews.*

Integration in die Viewer Die Beschreibung erfolgt für beide Viewer vereinheitlicht, da sich die Integration konzeptionell nicht unterscheidet. Die Visualisierung der Minimap wird über zwei Klassen realisiert: das `MinimapWidget` und das `MinimapCanvas` (Abbildung 4.2). Das `MinimapCanvas` erbt von `QWebView` und kapselt die Funktionalitäten zur Interaktion mit der eigentlichen Karte. Dazu gehören die Navigation aus dem Viewer oder aus der Minimap und das Zeichnen von Boundingboxen und dem Sichtfrustum (Abschnitt 4.1.4). Das `MinimapWidget` stellt ein abstrahiertes Interface für die Interaktion mit der Minimap bereit: Die mit dem `MinimapWidget` interagierenden Klassen arbeiten im Koordinatensystem des Viewers, während das `MinimapCanvas` in einem eigenen Koordinatensystem arbeitet (Abschnitt 4.1.3). Das `MinimapWidget` wandelt die Werte zwischen diesen Koordinatensystemen um, bevor es die Funktionsaufrufe weiterleitet. Zur Integration der Minimap wird das `MinimapWidget` in die GUI eingebunden und die Navigationssignale mit der Navigationsklasse verbunden.

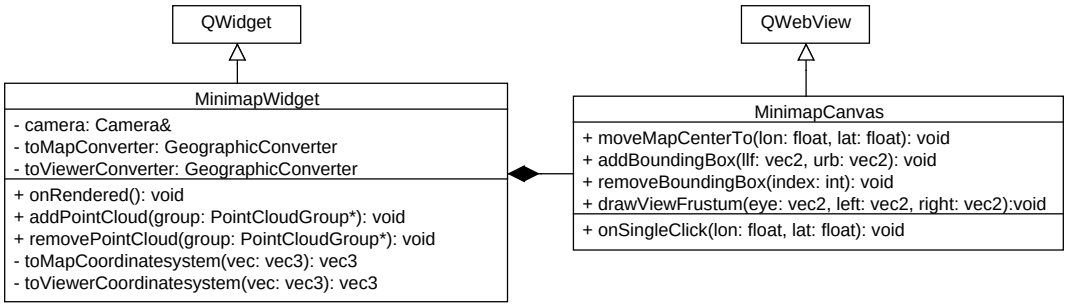


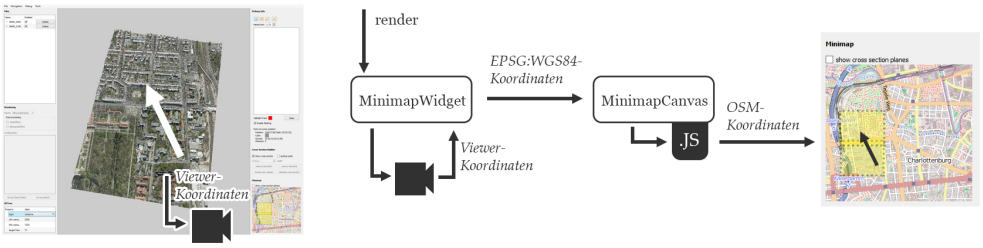
Abbildung 4.2: *Architektur des Minimapwidgets.*

Integration in den Prozessierungsserver Die Integration der Minimap in den Prozessierungsserver erfolgt ähnlich. Die Karte wird über OpenLayers in das Frontend eingebunden (Bäumer, 2014).

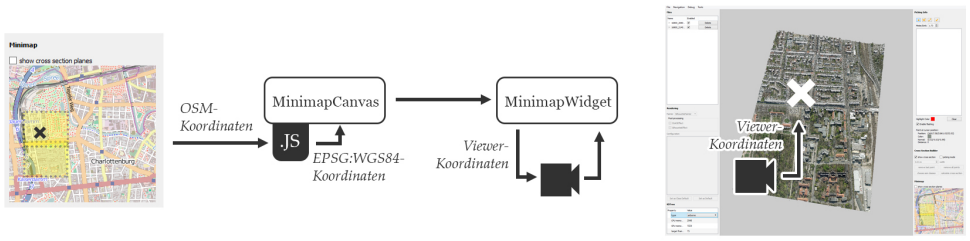
[1] `http://qt-project.org/doc/qt-5/signalsandslots.html`

4.1.3 Navigation und Interaktion

Die in Abschnitt 3.2 vorgestellten Konzepte zur Navigation in der Minimap erfordern einen Austausch der Koordinatensysteme zwischen Viewer und Minimap (Abbildung 4.3). Wird in der Szene navigiert und das Projektionszentrum verschoben, stellt das `MinimapWidget` diese Änderung beim folgenden Renderaufruf fest und extrahiert die Koordinaten des Positionszentrums. Die Koordinaten müssen aus dem Koordinatensystem des Viewers (definiert durch einen beliebigen EPSG-Code) in das Koordinatensystem der Minimap umgerechnet (`EPSG:WGS84`) werden, bevor über das `MinimapCanvas` das Zentrum der Karte verschoben wird. Die OpenStreetMap-internen Koordinaten werden automatisiert aus dem EPSG:WGS84-Koordinatensystem durch OSM berechnet. Klickt der Nutzer in der Minimap, wird die Position des Klicks auf das Projektionszentrum des Viewers übertragen, die Umwandlung der Koordinaten erfolgt wieder im `MinimapWidget`.



(a) *Übertragung der Projektionszentrumverschiebung im Viewer auf die Minimap.*



(b) *Übertragung eines Sprungbefehls in der Minimap auf den Viewer.*

Abbildung 4.3: *Übertragung der Navigation zwischen Viewer und Minimap.*

4.1.4 Navigations- und Interaktionshilfen

Die Visualisierung der Boundingboxen und des Sichtfrustums erfolgt über OpenLayers. Für beide Visualisierungen werden Layer über dem Basislayer erstellt.

Boundingboxen Die Visualisierung der Boundingboxen erfolgt über das Zeichnen simpler Boxen mit transparenter Füllfarbe (Abbildung 4.5(a)). Die Verwaltung der Boundingboxen übernimmt das `MinimapWidget` (Abbildung 4.2). Die Boundingboxen werden beim Import einer 3D-Punktwolke an die Minimap übergeben, welche die Koordinaten der Boundingbox ausliest und speichert. Parallel zu dieser Struktur wird durch das Script eine zweite Struktur von OpenLayers Boundingboxen erstellt, welche für das Zeichnen genutzt

werden. Beim Löschen der Boundingboxen – durch das Entfernen von 3D-Punktwolken aus dem Viewer – ermittelt das `MinimapWidget` den Index der gelöschten Boundingbox, damit das Löschen der Boundingbox in OpenLayers indexbasiert erfolgen kann.

Sichtfrustum Für den Nutzer sollen das Projektionszentrum und das Sichtfrustum visualisiert werden. Das Frustum soll dabei den Blickwinkel und die Blickrichtung visualisieren. Die Position des Projektionszentrums wird über einen Punkt visualisiert, an den das Sichtfrustum als halbtransparentes Dreieck grenzt (Abbildung 4.5(b)).

Der Blickwinkel wird über die Verkürzung des Sichtfrustums visualisiert. Hierfür werden die Eckpunkte des auf die xy-Ebene begrenzte Sichtvolumens in der Höhe gemittelt, die mit der Position des Projektionszentrums ein Dreieck aufspannen. Für die zweidimensionale Visualisierung wird dieses Dreieck auf die xy-Ebene projiziert (Abbildung 4.4).

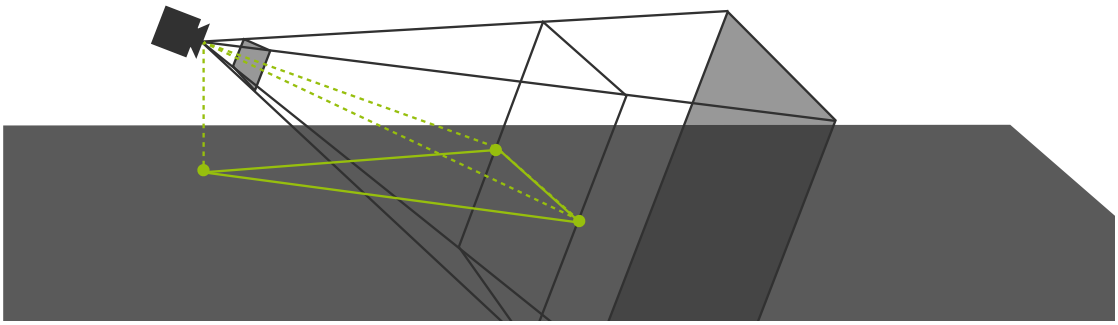
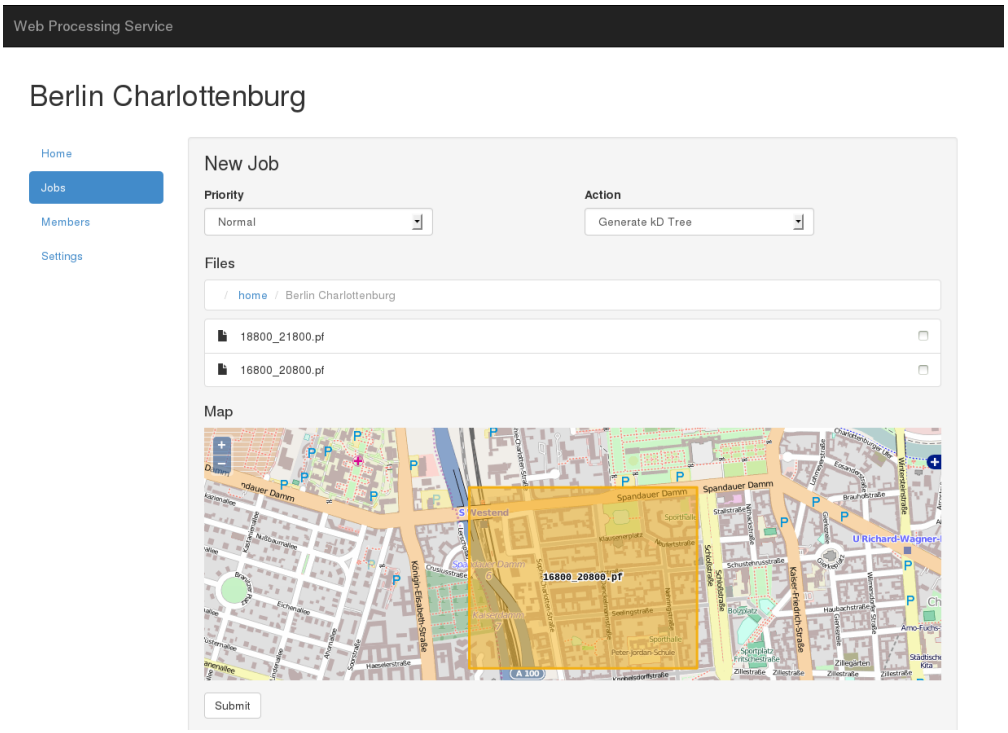
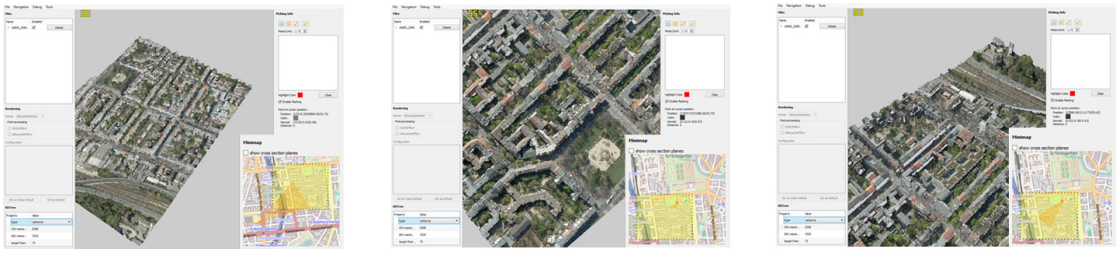


Abbildung 4.4: *Zweidimesionale Projektion des Sichtfrustums.*



(a) *Minimap mit Bounding im Frontend des Prozessierungsservers.*



(b) *Minimap des neuen Viewers mit Sichtfrustum in verschiedenen Winkeln.*

Abbildung 4.5: *Visualisierung von Boundingboxen und des Sichtfrustums.*

4.2 Querschnittserstellung und -anzeige

Die Querschnittserstellung setzt sich aus drei Schritten zusammen: der Konfiguration des zu erstellenden Querschnitts, der Selektion der Punkte und der Viusalisierung des berechneten Querschnitts. Neben der Einordnung in die bestehende Architektur, wird die Konfiguration des Querschnitts insbesondere unter dem Augenmerk der Schnittebenenvisualisierung vorgestellt. Anschließend erfolgt die Vorstellung des Selektionsalgorithmus' und der Visualisierungstechnik der Ergebnispunkte.

4.2.1 Eingliederung in die bestehende Architektur

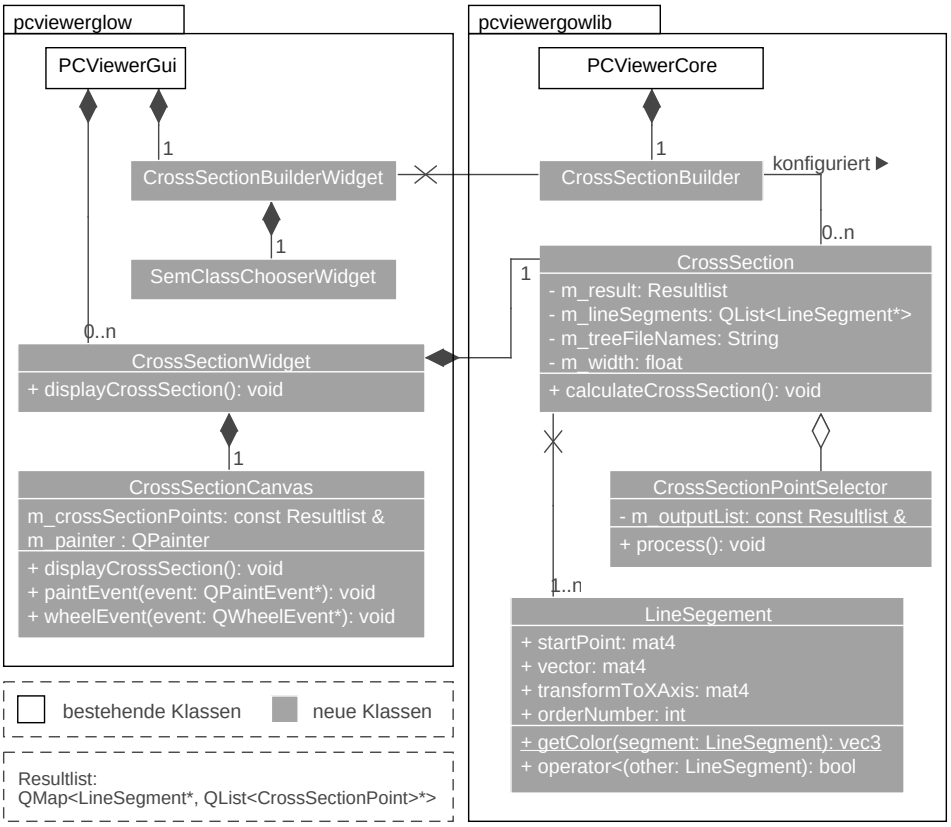


Abbildung 4.6: *Eingliederung der Querschnittsklassen in die bestehende Architektur.*

Das Querschnittstool wird – wie in Kapitel 3 konzipiert – vorerst nur in den neuen Viewer integriert. Die Beschreibung der Eingliederung kann daher spezialisiert erfolgen. Die grundlegenden Komponenten können gut in eine andere Architektur integriert werden, da sie eine lose Kopplung zu den bestehenden Komponenten besitzen. In Abbildung 4.6 werden die wichtigsten Klassen, die für die Querschnittserstellung[2] und -anzeige

[2]Querschnitt: <engl.> cross section

verantwortlich sind, dargestellt.

Die Komponenten zur Querschnittserstellung verteilen sich über zwei Pakete des bestehenden Viewers. Das Anwendungspaket `pcviewerglow` enthält die grafischen Komponenten zur Konfiguation und Visualierung. Diese Komponenten werden vom `PCViewerGui` gehalten, der das zentrale Management der Widgets durchführt. Alle weiteren Komponenten befinden sich im Bibliothekspaket `pcviewerglowlib`: Der `PCViewerCore`, welcher für das Kernkomponentenmanagement zuständig ist, enthält einen Builder für die Konfiguration und Erstellung von Querschnitten.

Einzelne Querschnitte werden durch `CrossSection`s repräsentiert. Sie enthalten neben den Punkten des Querschnitts die gesetzten Konfigurationsparameter: die Schnittebenen (siehe Abschnitt 4.2.3), die ausgewählten semantischen Klassen, die Dateinamen der genutzten 3D-Punktwolken sowie die Korridorbreite. Jede CrossSection hält zusätzlich einen `CrossSectionPointSelector`, der die eigentliche Selektion der Punkte vornimmt. Die Konfiguration eines Querschnitts erfolgt über den `CrossSectionBuilder`, für den das `CrossSectionBuilderWidget` eine grafische Nutzerschnittstelle anbietet. Das `SemClassChooserWidget` stellt eine Kapselung der Auswahl semantischer Klassen des `CrossSectionBuilderWidget`s dar (siehe Abschnitt 3.3). Nach erfolgreichem Abschluss der Berechnung werden die Punkte des Querschnitts in einem `CrossSectionWidget` visualisiert. Jedes Querschnittswidget wird zum `PCViewerGui` hinzugefügt und enthält die zugehörige `CrossSection`.

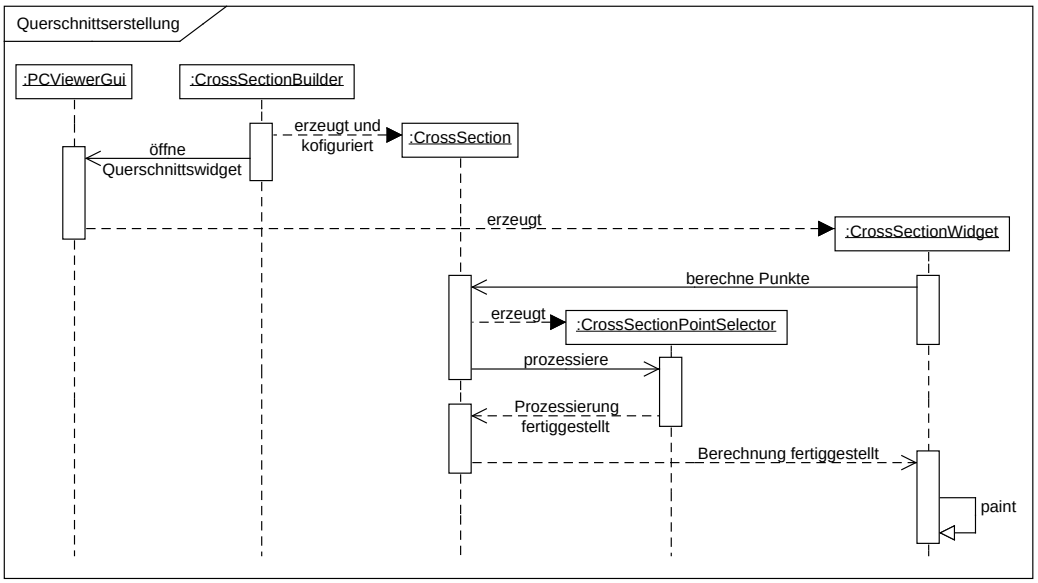


Abbildung 4.7: *Beteiligte Komponenten der Querschnittserstellung. Die GUI-Klassen des Builders werden nicht visualisiert.*

Nach der Erstellung eines Querschnitts (`CrossSection`) durch den `CrossSectionBuilder` sendet dieser ein Signal an die `PCViewerGui`-Komponente (Abbildung 4.7). Diese erstellt

daraufhin ein `CrossSectionWidget`, welches das Management der `CrossSection` übernimmt und die ersten Informationen anzeigt. Anschließend wird die Berechnung – über den `CrossSectionPointSelector` – der Punkte gestartet, nach deren Abschluss die Punkte visualisiert werden.

4.2.2 Konfiguration des Querschnitts

Die Konfiguration des Querschnitts erfolgt über eine Nutzerschnittstelle (Abbildung 4.8). Konfigurierbar sind neben der Korridorbreite die semantischen Klassen. Zudem können die Querschnittsunkte bearbeitet werden.

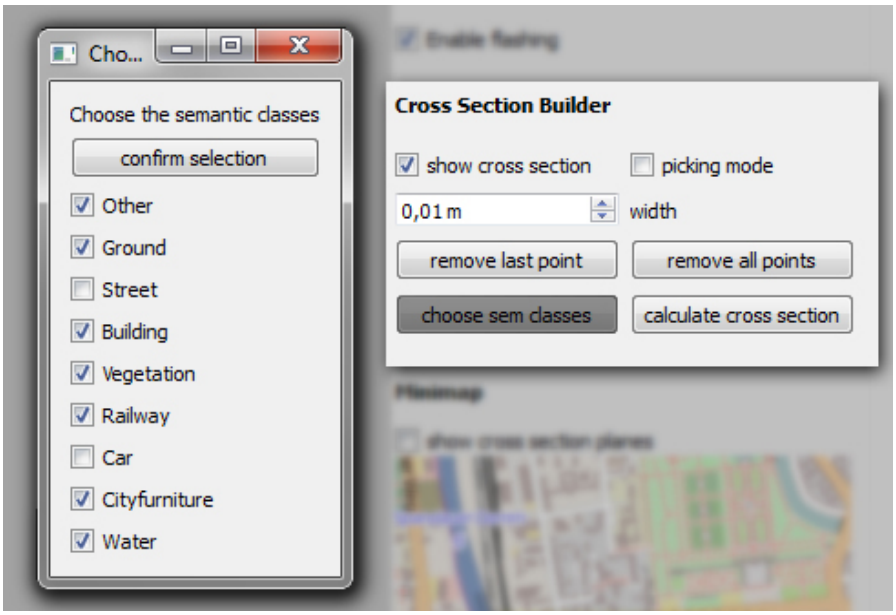


Abbildung 4.8: *GUI für die Querschnittskonfiguration.*

Verwalten der Querschnittspunkte Die Querschnittspunkte sollen durch den Nutzer gesetzt und einzeln oder vollständig gelöscht werden können (Abschnitt 3.3). Diese Verwaltung der Querschnittspunkte erfolgt über den `CrossSectionBuilder`. Dieser stellt ein Interface für die Verwaltung der Querschnittspunkte zur Verfügung: Das Interface beinhaltet sowohl Anfragen für das Hinzufügen als auch Benachrichtigungen für das Hinzufügen und Löschen von Querschnittspunkten. Die Minimap und der Picker (Eyssen, 2014) nutzen dieses Interface für das Hinzufügen von Querschnittspunkten. Die Anfragen des Pickers werden nur in einem über die GUI aktivierten Modus angenommen, um das versehentliche Hinzufügen zu verhindern (Abbildung 4.8). Werden Querschnittspunkte über die GUI des Builders gelöscht, werden die entsprechenden Punkte oder Ebenen aus der Visualisierung entfernt.

Visualisierung der Querschnittspunkte und Schnittebenen Die Querschnittspunkte werden zusammen mit den entstehenden Schnittebenen in der Minimap und im Viewer

visualisiert. Während sich die Minimap insbesondere zur Darstellung der exakten Ausrichtung eignet, kann in der Szene die Breite der Korridore um die Schnittebenen visualisiert werden (Abbildung 4.9).

In der Minimap erfolgt die Visualisierung in zusätzlichen Overlay-Layern. Die Querschnittspunkte werden als kleine Kreise mit Kontur dargestellt, die Schnittachsen als einfache Linien. Da die Liniendicke und Punktgröße zoomunabhängig definiert werden, kann der Nutzer die Position der Elemente beliebig genau betrachten.

Gemeinsame Visualisierung der Korridore und der Schnittebenenausrichtung in der gerenderten 3D-Punktwolke und in der Minimap.

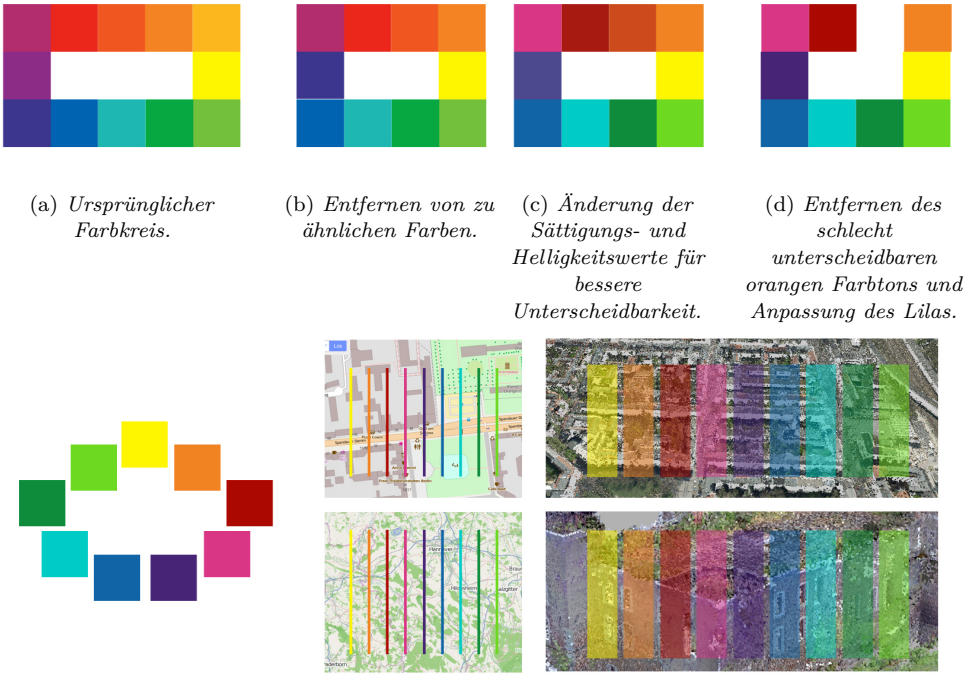
Abbildung 4.9: *Visualisierung der Querschnittspunkte und Schnittebenen.*

Die Visualisierung der Korridore im 3D-Punktwolkenrendering erforderte eine grundlegende Veränderung der Architektur, da bisher nur das Rendering von 3D-Punktwolken vorgesehen war. Die Architekturanpassung sollte Kompatibilität zu den bisherigen Renderingkomponenten sicherstellen und parallel zu der bestehenden Architektur aufgebaut werden. Renderbare Geometrien werden durch `Geometry`s repräsentiert und durch den `GeometryManager` und den `GeometryRenderDirector` verwaltet. Im Gegensatz zum Painter-Konzept der `PointCloud`s (Jendruk, 2014) gibt es kein globales Management der Renderingtechniken: Einheitliche oder durch den Nutzer einstellbare Renderingtechniken werden bisher für das Rendern von Geometrie nicht benötigt. Darüber hinaus wäre eine Anpassung oder Übertragung des Painter-Konzepts auf Geometrie unverhältnismäßig für das bisher in nur einem Fall genutzte Geometrierendering.

Farbmapping Den Schnittebenen wird eine feste Farbe zugewiesen. Diese werden so gewählt, dass eine eindeutige Unterscheidung innerhalb einer Ansicht und die Zuordnung der Schnittebenen zwischen den verschiedenen Ansichten möglich sind. Die Schnittebenen werden in der Minimap als opake, dünne Linien gezeichnet; Das Rendering der Korridore erfolgt in der Szene von 3D-Punktwolken in Form von semitransparenten Boxen. Die

gewählten Farben müssen sowohl in der Minimap als auch in der Szene gut erkennbar sein. Die Erkennbarkeit der Korridore soll dabei nur bei mittels Luftbild eingefärbten 3D-Punktwolken sichergestellt werden, da auf die Auswahl der Punktfarbe durch den Nutzer kein Einfluss genommen werden kann.

Da für jeden Querschnitt beliebig viele Schnittebenen gewählt werden können, erfolgt die Farbwahl aus einem diskreten Farbverlauf, der zu einem Farbkreis geschlossen ist. Ein automatisch berechneter, nicht zum Kreis geschlossener Farbverlauf könnte die mögliche Verwechslung der Schnittebenenzuordnung bei mehrfach verwendeten Farben verringern, jedoch ist die Unterscheidbarkeit in den verschiedenen Visualisierungen nicht gewährleistet. Die entwickelten Farben basieren auf auf dem Farbkreis und den Regeln von Stone (2006). Die Entstehung ist neben der Visualisierung in der Minimap und der Szene in Abbildung 4.10 dargestellt.



(a) *Ursprünglicher Farbkreis.*

(b) *Entfernen von zu ähnlichen Farben.*

(c) *Änderung der Sättigungs- und Helligkeitswerte für bessere Unterscheidbarkeit.*

(d) *Entfernen des schlecht unterscheidbaren orangen Farbtons und Anpassung des Lilas.*

(e) *Finale Farbauswahl mit Sortierung.*

(f) *Opake Visualisierung auf Minimap und halbtransparente auf gerenderten 3D-Punktwolken.*

Abbildung 4.10: *Entstehung des Farbkreises.*

4.2.3 Selektion der Punkte

Die Selektion der Punkte erfolgt in einem separaten Thread. Aufgrund der Orthogonalität der Schnittebenen zur xy-Ebene können diese als zweidimensionale Liniensegmente repräsentiert werden. Eine zweidimensionale Repräsentation eignet sich besonders, da die Abstandsberechnungen auf die zweidimensionale euklidische Distanz beschränkt werden können. Die Liniensegmente werden durch die Klasse `LineSegment` repräsentiert

(Abbildung 4.6). Sie enthält den Startpunkt des Liniensegments, den Offsetvektor zum zweiten Punkt und eine Ordnungsnummer. Darüber hinaus wird bei der Erstellung eine Matrix berechnet, die das Transformieren in einen einheitlichen Korrdior ermöglicht. Die Ordnungsnummern der Liniensegmente werden für die Sortierung und Zuordnung der Farben genutzt.

Die Selektion der Punkte erfolgt in drei Schritten: das Laden der KD-Trees, das Selektieren der geschnittenen KD-Tree-Knoten und das Selektieren der Punkte aus diesen Knoten. Hierfür wird der folgende Algorithmus verwendet (Abbildung 4.11): Für die Berechnung erstellt die `CrossSection` einen `CrossSectionPointSelector`, der die Ergebnisliste der `CrossSection` befüllt. Das Laden der KD-Trees erfolgt dabei anhand einer Liste von Dateinamen, die der `CrossSectionBuilder` zum Start der Berechnung ermittelt und der `CrossSection` übergeben hat. Kann eine dieser Dateien nicht vollständig geladen werden, wird der Vorgang abgebrochen und der Nutzer erhält eine Fehlermeldung.

Nach erfolgreichem Lesen folgt die Selektion der Knoten, die mindestens einen Korridor schneiden. Diese Selektion erfolgt über ein Abstandskriterium: Liegt eine Ecke der Boundingbox des Knotens weniger als 0.5∗Korridorbreite zu einem der Liniensegmente entfernt, wird der Knoten ausgewählt. Die ausgewählten Punkte werden in Knotenlisten gespeichert, die neben den Knoten die Information enthalten, aus welchem KD-Tree die Knoten stammen. Im nächsten Schritt werden die Punkte der einzelnen Knoten geladen und einzeln mit dem Abstandskriterium geprüft. Liegen sie innerhalb eines Korridors – also weniger als $0.5 *$ Korridorbreite zu einem der Liniensegmente entfernt – werden sie der Ergebnisliste hinzugefügt. Die Ergebnisliste besteht aus einer Map, die als Schlüssel die Lineinsegmente enthält und als zugehörigen Wert eine Liste der für diesen Korridor enthaltenen Punkte. So bleibt die Zuordnung der selektierten Punkte zu den Korridoren erhalten.

4.2.4 Visualisierung der selektierten Punkte

Die Visualisierung der selektierten Punkte erfolgt zweidimensional in einem `QWidget`. Die Punkte werden über die in den Liniensegmenten gespeicherten Transformationen in einen einheitlichen Korridor transformiert und in einer Seitenansicht dargestellt. Die Visualisierung der Punkte ist beliebig zoombar. Die visualisierten Punktattribute sind die Position innerhalb des Querschnitts, die Farbe, der Abstand zur Schnittebene und die Zuordnung zu den Schnittebenen (Abblidung 4.12(a)). Der Abstand zur Schnittebene wird über Transparenz und die Größe visualisiert (Abbildung 4.12(b)). Die Linien unterhalb den Punkten stellen die Zuordnung zu den Schnittebenen dar. Der Nutzer hat darüber hinaus die Möglichkeit, die Punkte anhand ihrer Schnittebenenfarbe zu färben (Abbildung 4.12(c)).

Jeder der erstellten Querschnitte wird in einem separaten Fenster visualisiert. So erhält der Nutzer die Möglichkeit, mehrere Querschnitte gleichzeitig zu visualisieren und untereinander zu vergleichen.

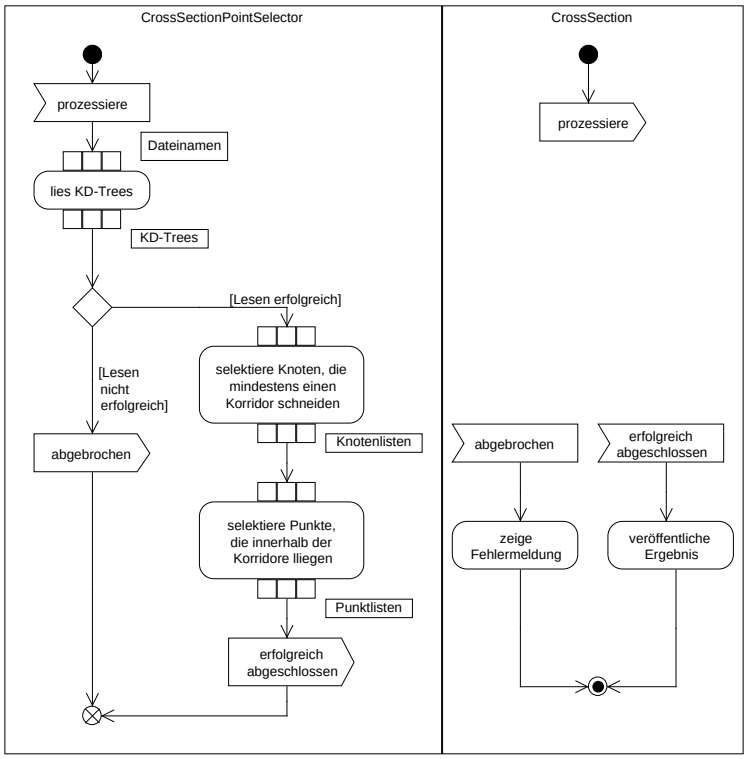
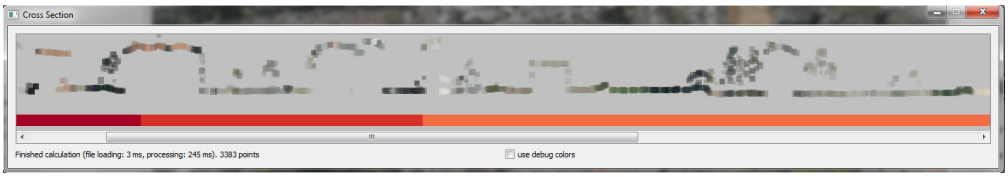
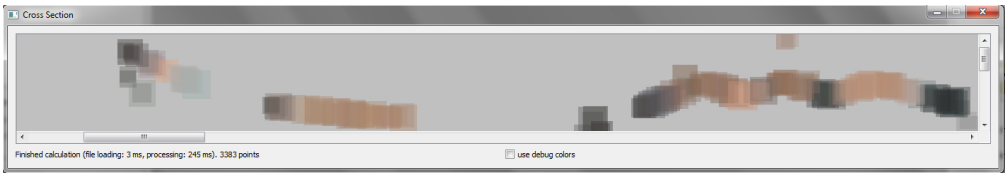


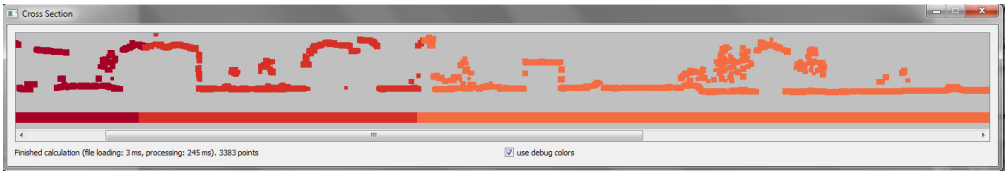
Abbildung 4.11: *Visualisierung der Querschnittspunkte und Schnittebenen.*



(a) *Visualisierung der Punkte.*



(b) *Visualisierung des Abstands zur Schnittebene über Transparenz und Größe.*



(c) *Visualisierung der Zugehörigkeit zu den Schnittebenen über Farbe.*

Abbildung 4.12: *Visualisierung der selektierten Punkte.*

Kapitel 5

Diskussion und Fazit

Im Folgenden werden die Ergebnisse der Implementierung evaluiert. Die Bewertung erfolgt dabei sowohl über die Performance als auch über die visuelle Qualität. Anschließend erfolgt eine Übersicht über die Erweiterungsmöglichkeiten und das Fazit.

5.1 Evaluation der Ergebnisse

Die Vorstellung der Implementierungsergebnisse erfolgt getrennt nach Minimap und Querschnittsansichten. Zuvor wird das Testsystem für die Performancetests vorgestellt.

5.1.1 Testumgebung

Die Performancetests wurden auf einem Desktop-PC mit einem 64-bit Windows 8.1 Pro-Betriebssystem und den folgenden Eigenschaften durchgeführt:

- Prozessor / RAM: Intel Xeon E5-1620 mit 3,6 GHz / 8 GB RAM
- Grafikkarte: GeForce GTX 660 mit 2048 MB Grafikspeicher

Da sich die Durchführung der einzelnen Tests unterscheidet, wird die Vorgehensweise zu jedem Test separat beschrieben.

5.1.2 Minimap

Die implementierte Minimap ist bisher nur in Kombination mit gültigen EPSG-Codes und einer funktionierenden Internetverbindung nutzbar. Fehlt ein gültiger EPSG-Code, kann die 3D-Punktwolke nicht in der Minimap visualisiert und die Navigationshilfen nicht genutzt werden. Um dem Nutzer diese Funktionalitäten auch in Spezialfällen zur Verfügung zu stellen, müssen neue Konzepte entwickelt und umgesetzt werden.

5.1.3 Querschnittsansichten

Für die Evaluation der Querschnittsansichten sind die Performance der Punktselektion und die Renderingperformance der Visualisierung des Querschnitts von besonderem Interesse. Darüber hinaus erfolgt die Einschätzung des Renderings der Korridore über die visuelle Qualität.

Visualisierung der Schnittebenen Die Visualisierung der Schnittebenen erfolgt wie in Kapitel 4.2.2 vorgestellt über transparente Boxen im Viewer. Dabei treten zwei Probleme auf: Färbt der Nutzer die 3D-Punktwolke in einheitlichen, selbst gewählten Farben, sind die Farben der transparenten Korridore gegebenefalls nicht mehr erkennbar (Abbildung 5.1). Die einfache Transparenz über `glBlendFunc` führt darüber hinaus zu falscher Verdeckung und Z-Fighting (Abbildung 5.2). Beide Probleme können über den Einsatz aufwendigerer Techniken gelöst werden: Die Farben der Schnittebenen können für den Nutzer veränderbar gemacht werden, während komplexere Algorithmen die Transparenzfehler beseitigen.

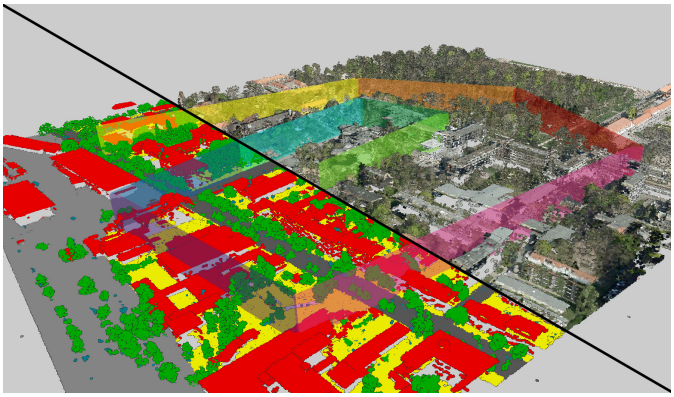


Abbildung 5.1: *Sichtbarkeit der Korridore um die Schnittebenen im Vergleich: links anahnd der semantischen Klassen in stereotypen Farben und rechts mittels Luftbild eingefärbte 3D-Punktwolke.*

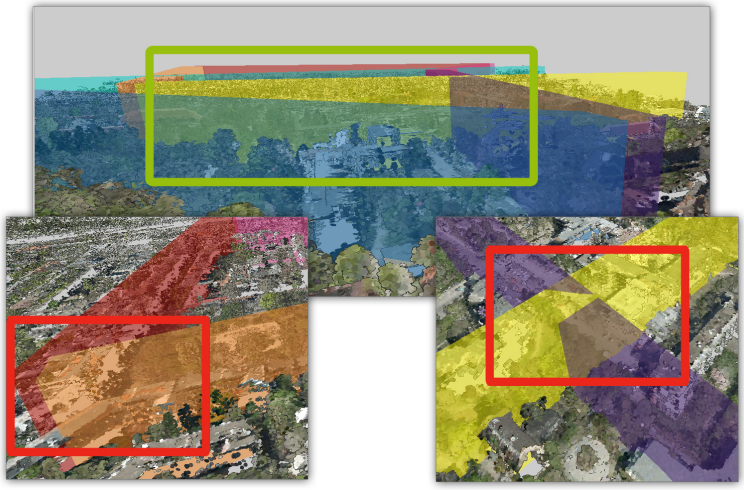


Abbildung 5.2: *Auftretende Fehler in der Transparenz: Verdeckung (links), Z-Fighting (rechts). Korrektes Rendering der Transparenz (oben).*

Berechnung des Querschnitts Die Berechnung des Querschnitts soll für realistische Querschnittsbreiten und -längen[1] maximal wenige Sekunden dauern. Zur Überprüfung wurde eine Performancemessung durchgeführt (Tabelle 5.1). Für den Testquerschnitt wurde die dichte[2] 3D-Punktwolke der »Deutschordenskommende in Siersdorf«genutzt. Die gewählte Schnittebene hat eine Länge von ca. 40 Metern und verläuft durch den Hauptteil des Herrenhauses (Abbildung 5.3). Zur kontinuierlichen Erhöhung der Punktanzahl wurde die Korridorbreite bei gleichbleibender Schnittebene schrittweise erhöht.

Breite	Punktanzahl	KD-Tree Ladezeit	Prozessierungszeit	davon Knotenladezeit
0.01 m	7 295	1 821	81	28
0.02 m	18 412	1 866	83	26
0.05 m	52 092	1 828	96	38
0.1 m	112 439	1 839	112	34
0.2 m	259 841	1 833	157	40
0.5 m	857 964	1 801	371	56
0.7 m	4 230 979	1 879	1 325	232
1 m	6 872 300	1 856	2 508	664
2 m	16 915 121	1 829	8 644	3 253

Tabelle 5.1: *Dauer der Berechnung für einen Test-Querschnitt durch die 3D-Punktwolke der »Deutschordenskommende Siersdorf«. Das Laden der KD-Tree-Strukturen erfolgt vor der eigentlichen Prozessierung. Die Prozessierungszeit beinhaltet die Knotenladezeit, da die benötigten Knoten erst während der eigentlichen Prozessierung geladen werden. Alle Zeitangaben sind in Millisekunden angegeben.*

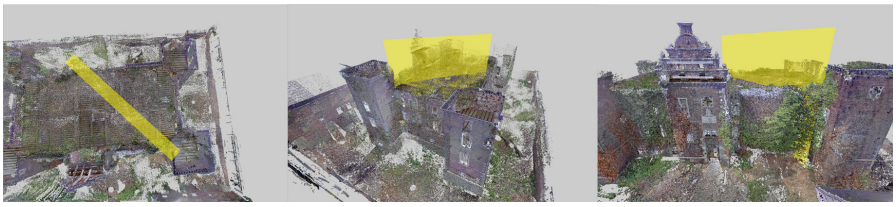


Abbildung 5.3: *Platzierung der etwa 40 m langen Test-Schnittebene.*

Die Messwerte zeigen, dass die Zeiten für die Erstellung eines Querschnitts stark von den Ladezeiten der Dateien abhängen. Etwa 30 Prozent der eigentlichen Prozessierungszeit werden für die Ladezeit der Knotendaten benötigt. Hinzu kommt die Ladezeit der KD-Tree-Struktur vor der eigentlichen Prozessierung.

Insgesamt bleiben die gemessenen Zeiten unter fünf Sekunden. Nur für große Querschnitte mit mehr als 16 Millionen Punkten dauert die Berechnung länger als zehn Sekunden. Die Messwerte bleiben – bis auf sehr große Querschnittsvolumen – im vorgegebenen Rahmen. Zur weiteren Verbesserung der Werte eignet sich der Einsatz einer SSD, um die Ladezeiten zu verkürzen.

[1] mehrere Meter lang und bis zu wenigen Metern breit
[2] durchschnittlich 27 652 Punkte pro m^3

Visualisierung des Querschnitts Die Visualisierung des berechneten Querschnitts erfolgt in einem `QWidget` über einen `QPainter`, welcher insbesondere für das Zeichnen einfacher Primitiven entwicklelt wurde. Wie im Abschnitt zuvor ermittelt, erreichen Querschnitte bereits früh Punktmengen von über 100 000 Punkten. Zur Evaluation der Renderingperformance wurden mehrere Querschnitte berechnet und die Renderingperformance über die Dauer des *paint*-Aufrufs in Millisekunden gemessen (Abbildung 5.4). Für jede Punktmenge wurden etwa 20 Messungen durchgeführt und zu einem Gesamtergebnis gemittelt.

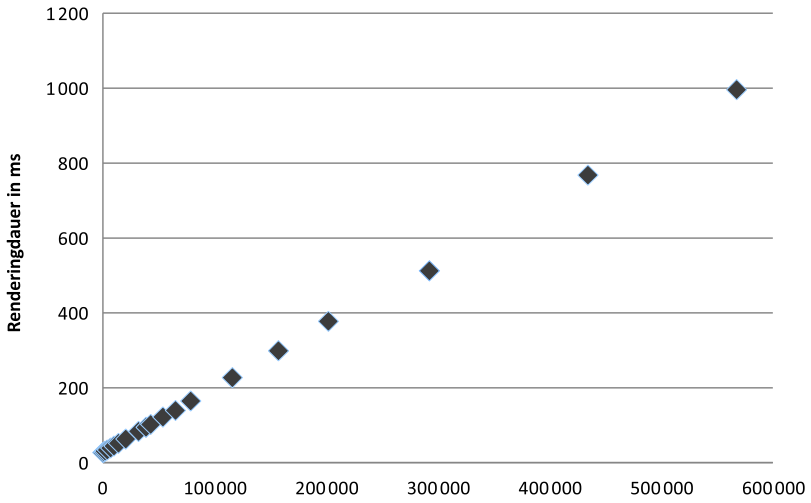


Abbildung 5.4: *Renderingzeit einer Querschnittsansicht abhängig von der Punktmenge.*

Bei etwa 300 000 Punkten werden für das Rendering eines Frames bereits 500 Millisekunden benötigt – umgerechnet etwa zwei Frames pro Sekunde. Eine so geringe Framerate ist für die Querschnittsansicht ungeeignet, da gerade beim Zoomen – dem Auslöser des paint-Events – Responsivität wichtig ist. Für Punktmengen weit über 1 000 Punkte ist die Framerate nicht mehr praktikabel.

5.2 Erweiterungsmöglichkeiten und Ausblick

Die bisherige Implementierung der Minimap und der Querschnittsansichten enthalten einen bisher ausreichenden Funktionsumfang. Jedoch können weitere Features implementiert werden, um die in der Evaluation vorgestellten Grenzfälle abzudecken oder gänzlich neue Funktionalitäten hinzuzufügen.

Minimap Kann der Nutzer durch eine fehlende Internetverbindung oder fehlende EPSG-Codes die Funktionen einer OpenStreetMap-Karte nicht nutzen, müssen andere Konzepte entwickelt werden. Eine mögliche Variante ist die Nutzung einer gerenderten Draufsicht der Szene als Karte (Abbildung 5.5). Sind mehrere 3D-Punktwolken in der Szene geladen, können die Boundingboxen und das Sichtfrustum trotz fehlender Weltkarte zur leichteren Navigation beitragen.

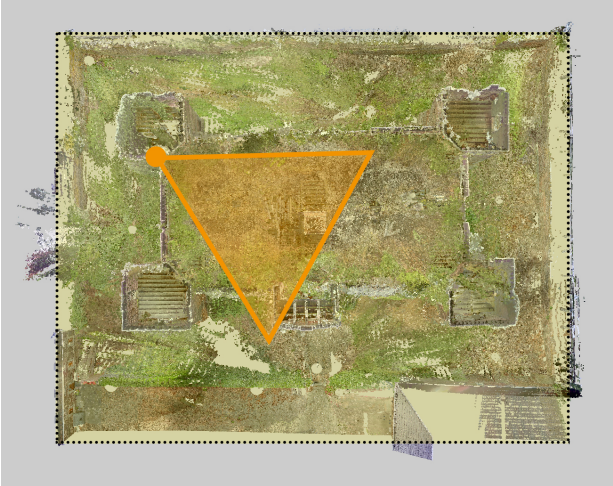


Abbildung 5.5: *Beispielhafte Umsetzung einer gerenderten Draufsicht als Minimap.*

Eine weitere Möglichkeit ist die Visualisierung mehrerer Ebenen als Karte bei terrestrischen Scans von mehrgeschossigen Bauwerken: Gerenderte Draufsichten von jedem Stockwerk könnten dynamisch für die aktuelle Position des Projektionszentrums ausgewechselt werden und die Navigation innerhalb eines Stockwerks verbessern. Weitere Konzepte können aus der Kombination beider Ansichten entwickelt werden.

Querschnittsansichten Die geringe Performance der Querschnittsvisualisierung bei Punktmengen über 100 000 Punkte erfordert eine Umstellung des Visualisierungskonzepts. Eine Möglichkeit ist das Entwickeln eines Out-of-Core-Verfahrens für die Auswahl der Punkte. Die dazu nötigen räumlichen Datenstrukturen wurden in Kapitel 2 vorgestellt. Die für das performante Rendering genutzten Verfahren wird von Tausche (2014) vorgestellt. Diese Verfahren könnten zusammen mit den Renderingtechniken aus den Viewern für die Visualisierung der Querschnitte genutzt werden. Diese Variante bietet darüber hinaus die Möglichkeit, die bereits implementierten Picking-Funktionalitäten (Eyssen, 2014) zum Auslesen von speziellen Punktdaten und zum Messen von Strecken und Flächen zu nutzen.

5.3 Fazit

Im Fokus dieser Arbeit stand die Entwicklung und Umsetzung von Visualisierungstechniken für die Analyse und Interpretation von 3D-Punktwolken. Die dazu umgesetzten Tools sind eine Minimap und Querschnittsansichten. Beide Features sollen dem Nutzer einfachere Mittel für die Analyse und Interpretation zur Verfügung stellen.

Die Implementierung der Minimap erfolgte webbasiert mit OpenStreetMap. Für eine verbesserte Navigation wurden Hilfsmittel wie eingezeichnete Boundingboxen oder das symbolisierte Sichtfrustum umgesetzt. Für das Frontend des Prozessierungsservers entstand eine Minimap mit ähnlichen Funktionen zur Visualisierung der selektierten 3D-Punktwolkendaten.

Die Umsetzung der Querschnittsansichten umfasst die Konfiguration des Querschnitts, die Berechnung der Querschnittspunkte und die Visualisierung der im Querschnitt enthaltenen Punkte. Nutzer können über die Minimap und Picking Querschnittspunkte setzen, welche die Schnittebenen aufspannen und den Querschnitt definieren. Zudem lassen sich die Breite der Korridore und die einberechneten semantischen Klassen über die GUI einstellen. Die anschließende Berechnung erfolgt über die vorhandenen räumlichen Datenstrukturen, deren Ergebnis in einem separaten Fenster visualisiert wird.

Defizite treten beim Rendering der Schnittebenen und der Visualisierung der Querschnitte auf. Darüber hinaus ist der Einsatz der Minimap ohne Internetverbindung nicht möglich. Für diese Probleme wurden Perspektiven und Möglichkeiten für die weitere Entwicklung und Verbesserung aufgezeigt.

Literaturverzeichnis

Bäumer, Y. (2014). Visualisierung und serviceorientierte Prozessierung von 3D-Punktwolken. Bachelorarbeit. Hasso-Plattner-Institut.

Bentley, J. L. (1975). Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9), 509–517.

Botsch, M., Spernat, M., & Kobbelt, L. (2004). Phong splatting. In *Proceedings of the First Eurographics Conference on Point-Based Graphics*, SPBG'04 (pp. 25–32).: Eurographics Association.

Cockburn, A., Karlson, A., & Bederson, B. B. (2008). A review of overview+detail, zooming, and focus+context interfaces. *ACM Computing Surveys (CSUR)*, 41(1), 2.

Discher, S. (2013). Echtzeit-Rendering-Techniken für 3D-Punktwolken basierend auf semantischen und topologischen Attributen. Master's thesis, Hasso-Plattner-Institut.

Eyssen, M. (2014). Interaktions- und immersive Visualisierungsstechniken für 3D-Punktwolken. Bachelorarbeit. Hasso-Plattner-Institut.

Finkel, R. A. & Bentley, J. L. (1974). Multidimensional Binary Search Trees Used for Associative Searching. 4, 1–9.

Glander, T., Trapp, M., & Döllner, J. (2008). 3D Generalization Lenses for Interactive Focus and Context Visualization of Virtual City Models. In *Information Visualisation, 2008. IV '08. 12th International Conference* (pp. 356 – 361).

Glander, T., Trapp, M., & Döllner, J. (2011). Concepts for Automatic Generalization of Virtual 3D Landscape Models.

Gobbetti, E. & Marton, F. (2004). Layered Point Clouds: A Simple and Efficient Multiresolution Structure for Distributing and Rendering Gigantic Point-sampled Models. *Comput. Graph.*, 28(6), 815–826.

Gong, J., Zhu, Q., Zhong, R., Zhang, Y., & Xie, X. (2012). An Efficient Point Cloud Management Method Based on a 3D R-Tree. 78(4), 373–381.

Jendruk, M. (2014). Rendering- und Beleuchtungstechniken für 3D-Punktwolken. Bachelorarbeit. Hasso-Plattner-Institut.

Linke, J. (2014). Efficient Preprocessing of 3D Point Clouds Based on Spatial Data Structures. Bachelorarbeit. Hasso-Plattner-Institut.

Meagher, D. (1987). High-speed Image Generation of Complex Solid Objects using Octree Encoding. US Patent 4,694,404.

Neis, P., Zielstra, D., & Zipf, A. (2012). The Street Network Evolution of Crowdsourced Maps: OpenStreetMap in Germany 2007–2011. 4, 1–21.

Pasewaldt, S., Trapp, M., & Döllner, J. (2011). Multiscale Visualization of 3D Geovirtual Environments Using View-Dependent Multi-Perspective Views. 19(3), 111–118.

Stone, M. (2006). Choosing Colors for Data Visualization. *Business Intelligence Network*.

Tausche, K. (2014). Out-of-Core Speichermanagement für das Rendering von massiven 3D-Punktwolken. Bachelorarbeit. Hasso-Plattner-Institut.

Vaaraniemi, M., Freidank, M., & Westermann, R. (2013). Enhancing the Visibility of Labels in 3D Navigation Maps. In *Progress and New Trends in 3D Geoinformation Sciences* (pp. 23–40). Springer.

Wand, M., Berner, A., Bokeloh, M., Fleck, A., Hoffmann, M., Jenke, P., Maier, B., Staneker, D., & Schilling, A. (2007). Interactive editing of large point clouds. In *Proceedings Symposium on Point-Based Graphics (PBG '07)*.

Yu, J., McMillan, L., & Sturm, P. (2008). Multiperspective Modeling, Rendering, and Imaging. In *ACM SIGGRAPH ASIA 2008 Courses*, SIGGRAPH Asia '08 (pp. 14:1–14:36). New York, NY, USA: ACM.