Das Las Vegas Reconstruction Toolkit

Kim Oliver RINNEWITZ, Sven Karl SCHALK,

Thomas WIEMANN, Kai LINGEMANN, Joachim HERTZBERG

Zusammenfassung

Die dreidimensionale Vermessung mittels terrestrischer Laserscanner hat sich in den vergangenen Jahren als Standardmethode zur Dokumentation von Bauwerken etabliert. Die Herausforderung besteht darin, aus den gewonnenen Punktdaten ein polygonales Oberflächenmodell zu erzeugen, das die vermessene Geometrie realitätsnah wiedergibt. Dieses Papier stellt eine Open-Source-Software vor, die die automatische, effiziente Erzeugung von texturierten Oberflächenmodellen (*Meshes*) aus farbigen 3D-Punktwolken erlaubt.

1 Einleitung

Seit kurzer Zeit richtet sich der Fokus in der Robotik zunehmend auf eine dreidimensionale Darstellung der Umgebungsgeometrien anstelle zweidimensionaler Karten. Dreidimensionale Karten bieten einige Vorteile wie etwa bei der Kollisionsvermeidung, Objekterkennung und semantischen Analyse der Szene. Mit der Entwicklung leistungsfähiger 3D-Scanner und preisgünstiger Sensorsysteme mit gleichzeitiger Texturerfassung wie der Microsoft Kinect wurde dieser Trend beschleunigt und viele neue Forschungsarbeiten ermöglicht. Der erreichte Detaillierungsgrad der von den Sensoren gelieferten 3D-Punktwolken kann sehr stark schwanken und übersteigt häufig eine performant handhabbare Datenmenge.

Das Las Vegas Surface Reconstruction Toolkit gibt dem Nutzer einige nützliche Werkzeuge in Form einer Open Source C++ Bibliothek an die Hand, die es ermöglichen, die Punktwolken in eine optisch wie auch Speicherplatz-optimierte, polygonale Flächendarstellung zu transformieren.

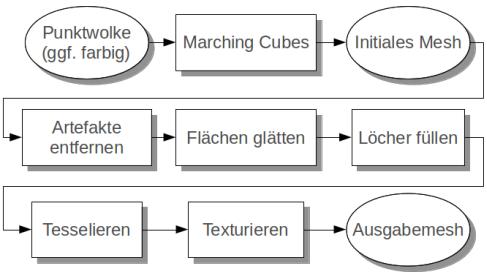

Abb. 1: Datenverarbeitungspipeline im Las Vegas Toolkit. Die Eingabepunktwolke wird zunächst mittels Marching Cubes in ein Mesh überführt, welches anschließend automatisch aufbereitet wird.

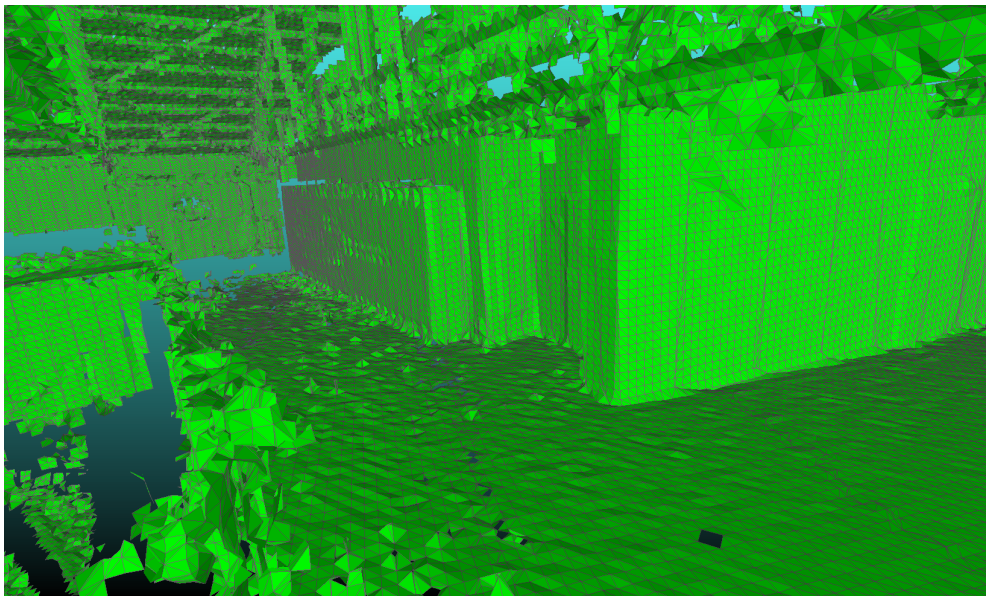

Abb. 2: Durch den Marching-Cubes-Algorithmus erzeugtes Mesh mit einer hohen Anzahl an Dreiecken. Das Mesh weist durch Sensorrauschen und Abschattungen verursachte Artefakte und Löcher auf.

2 Rekonstruktion

Die Basisfunktionalität der Software besteht in der Rekonstruktion von polygonalen Oberflächenmodellen aus 3D-Punktwolken. Dabei werden verschiedene Eingabeformate für Punktwolken unterstützt. So können neben ungefärbten auch farbige Punktwolken und Punktwolken mit bereits vorberechneten Punktnormalen geladen werden. Dadurch kann auf wiederholte Berechnung der Punktnormalen verzichtet werden. Die Rekonstruktion erfolgt nach dem bekannten Marching-Cubes-Verfahren (LORENSEN & CLINE, 1987) mit Hoppes Distanzfunktion zur Interpolation der gescannten Oberflächen (HOPPE 1992). Optional können auch Marching-Tetrahedrons verwendet werden (PAYNE & TOGA, 1990).

Zur Rekonstruktion ist eine Schätzung der Normalen der Datenpunkte notwendig, die auf Basis verschiedener Bibliotheken zur Nächsten-Nachbar-Suche durchgeführt werden kann. Unterstützt werden STANN, FLANN und nabo, die direkt in die Software integriert sind. Weitere Bibliotheken wie z.B. die Point Cloud Library (PCL, RUSU 2010) können optional eingebunden werden. Diese Bibliotheken werden auch bei der Berechnung der im Marching-Cubes-Algorithmus benötigten Distanzwerte verwendet.

Die Anzahl der Dreiecke, die durch Marching-Cubes-basierte Verfahren entstehen, hängt stark von der Auflösung des für die Rekonstruktion verwendeten Raumgitters ab. Alle durchgeführten Operationen zur Erzeugung der Dreiecke sind lokal, so dass sich dieser Prozess sehr gut parallelisieren lässt.

Das berechnete Mesh kann als binäre PLY-Datei oder im Wavefront-OBJ-Format geschrieben werden. Diese Standard-Austauschformate werden von vielen Software-Paketen wie Meshlab und Blender direkt unterstützt.

Durch Marching-Cubes-Algorithmen erzeugte Meshes weisen allerdings eine unnötig große Anzahl an Dreiecken auf, die zu erhöhtem Speicherbedarf und Performanceproblemen beim Rendern führen. Da die Meshes basierend auf den Laserscannerdaten generiert werden, führen Ausreißer in den Messdaten zu frei im Raum schwebenden Artefakten. Insbesondere bei stark rauschenden Daten treten Unebenheiten auf Flächen auf, die in der Realität glatt sind. Hinzu kommen Löcher an Stellen mit geringer Datendichte oder bei Verschattungen. Ein Teil dieser Probleme kann durch Filterung der Eingangsdaten verbessert werden, indem z.B. ein automatisches Outlier-Removal, wie es auch in der PCL implementiert ist, durchgeführt wird. Dies ist allerdings sehr rechenaufwändig und löst die Probleme nur bedingt. Aus diesem Grund wurden in der Software mehrere Verfahren zur Aufbereitung der initial erstellten Meshes implementiert. Deren Funktionalität soll in den folgenden Abschnitten präsentiert werden.

3 Meshoptimierung

Zur Beseitigung der oben beschriebenen Probleme in den initial erzeugten Meshes verfügt die Software über verschiedene Optimierungsverfahren. So können frei schwebende Artefakte erkannt und gelöscht, Flächen geglättet und Löcher gefüllt werden. Durch Neutriangulierung wird die Datenmenge reduziert. Alle Optimierungsmaßnahmen erfolgen vollautomatisch, benötigen also keine Interaktion des Benutzers.

3.1 Entfernen von Artefakten („Remove Dangling Artifacts“, RDA)

Der Remove-Dangling-Artifacts (RDA) Algorithmus wurde entwickelt, um die durch Fehlmessungen entstehenden Artefakte in den Rekonstruktionen zu entfernen. Dazu werden im Mesh frei schwebende Flächen gesucht, die nicht mit dem Rest des Meshes verbunden sind und deren Fläche eine vom Benutzer festgesetzte Größe nicht überschreitet. Zur Ermittlung der Größe der Flächen wird ein rekursives Region-Growing angewendet: Ausgehend von einem beliebig gewählten Dreieck wird getestet, ob die Flächennormale modulo eines Toleranzwertes nicht vom Startdreieck unterscheidet. Ist dies der Fall, wird das Dreieck zur aktuellen Region hinzugefügt und die Suche wird von dort aus rekursiv weitergeführt, bis ein Dreieck gefunden wird, das dem Normalenkriterium nicht mehr genügt. Die Größe der zu entfernenden Regionen wird dabei nicht über einen Flächengröße, sondern über die Anzahl der Dreiecke in der Region parametrisiert, um von der Skalierung der Eingabedaten unabhängig zu sein.

Der RDA-Algorithmus ist insbesondere für den integrierten Hole-Filling-Algorithmus (siehe Kap. 3.3) von Bedeutung, da er Artefakte, die fälschlicherweise als Löcher erkannt und geschlossen werden, zuvor entfernt. Ein Algorithmus zur Detektion von Löchern basiert in der Regel auf einer einfachen Randkonturverfolgung. Kann eine geschlossene Randkontur gefunden werden, so wird diese als Loch interpretiert. Die Notwendigkeit des RDA-Algorithmus wird insbesondere am Beispiel eines einzelnen, unzusammenhängenden, also frei im Raum erzeugten Dreiecks ersichtlich: Eine Konturverfolgung zur Löchererkennung würde das Dreieck umlaufen, eine geschlossene Randkontur feststellen und ein deckungs-

gleiches Dreieck über das ursprüngliche ziehen, wodurch sich der optische Eindruck nicht ändert aber ein geschlossenes Objekt ohne Loch erzeugt würde, welches, semantisch schlecht begründbar, im Mesh verbliebe. Ein weiterer nützlicher Nebeneffekt der mit der Entfernung kleiner Artefakte einhergeht, ist eine geringfügige Datenreduktion.

Ein Anwendungsbeispiel für den RDA-Algorithmus zeigt Abb. 3. Man kann deutlich erkennen, dass insbesondere an den Rändern des erfassten Bereichs viele unzusammenhängende Dreiecke aus der Rekonstruktion entfernt werden. Der markierte Bereich oben rechts im Bild zeigt ein Beispiel für das Entfernen eines größeren Clusters, der offensichtlich durch Fehlmessungen hervorgerufen wurde.

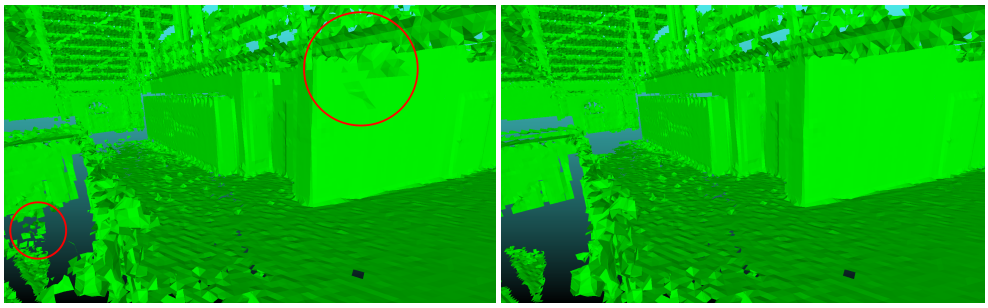

Abb. 3: Mesh vor und nach dem Entfernen von frei im Raum schwebenden Artefakten. Die hier gezeigte Art von Artefakten wird im Wesentlichen durch Messrauschen hervorgerufen.

3.2 Erkennung und Glättung von ebenen Flächen

Mit Hilfe des oben beschriebenen Region-Growing-Verfahrens lassen sich leicht ebene, zusammenhängende Flächen in den Rekonstruktionen ermitteln. Wie stark die durch Rauschen bedingten Unebenheiten in einer Fläche sein dürfen, damit diese noch als „eben“ erkannt wird, hängt von der Parametrisierung des Normalenkriteriums ab. Ein zu hoher Toleranzwert wird dafür sorgen, dass auch nicht-ebene Flächen als zusammenhängend detektiert werden. Wird der Wert zu klein gewählt, findet kein Clustering statt. Der ideale Wert hängt stark von der Qualität der Eingangspunktwolke ab. In der Praxis hat sich gezeigt, dass ein Wert von 0,2 Grad in vielen Datensätzen eine gute Separierung gewährleistet.

Wurde eine zusammenhängende Region erkannt, so werden alle zugehörigen Dreiecke in eine gemeinsame Regressionsebene gezogen, welche effizient mit einem RANSAC-Verfahren ermittelt wird. Dadurch tritt ein Glättungseffekt ein, der das Problem unebener Flächen bei rauschenden Messdaten stark reduziert. Um die ursprüngliche Geometrie nicht zu stark zu verändern (z.B. um Flächenübergänge bei Krümmungen zu erhalten), werden nur Regionen ausgeglichen, deren Größe einen vom Benutzer gewählten Schwellwert übersteigt. Um auch die Überganskanten zwischen Ebenen zu verbessern, werden diese miteinander Verschnitten und alle Vertices, die in der Nähe von Schnitten liegen in die berechnete Schnittlinie gezogen (s. Abb. 4).

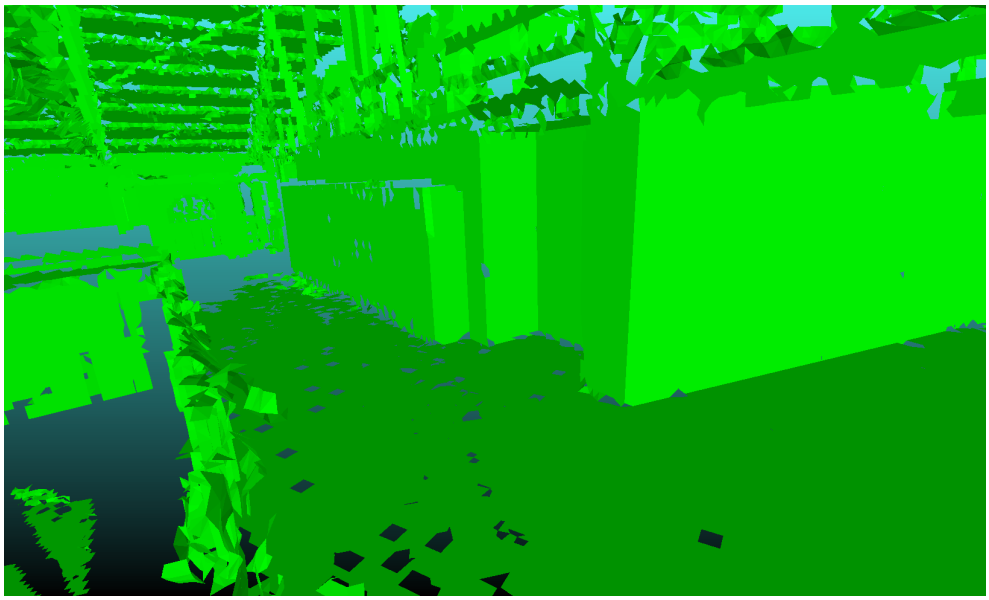

Abb. 4: Das optimierte Mesh nach Erkennung und Glättung ebener Flächen. Um auch die Kantenübergänge scharf zu rekonstruieren, wurden dich bei einander liegende Ebenen miteinander verschnitten.

3.3 Schließen von Löchern

Löcher automatisch in Meshes erkennen und schließen zu lassen, ist bereits insofern eine Herausforderung, als dass sich keine allgemeingültige Definition des Begriffes Loch finden lässt. Eine tellerförmige, frei im Raum hängende Scheibe würde man nicht als Loch, sondern eben genau als Scheibe bezeichnen. Schnürt man den Rand dieser Scheibe jedoch über ihrem Mittelpunkt zusammen, so erhält man eine Tassen-Form und schließlich eine Art Vase, die offensichtlich ein Loch hat, ohne etwas an der grundsätzlichen Struktur der Oberfläche geändert zu haben. Man kann sagen: „Ein Teller und eine Vase sind auf Mesh-Ebene isomorph“.

Ein zweites Problem, ebenfalls semantischer Art, ist die Festlegung, ab welcher Größe ein Loch als ungewolltes, zu schließendes Loch gelten soll. Das erste Problem wurde durch den RDA-Algorithmus bereits umgangen, während zur Lösung des zweiten die Anzahl der Lochkanten als Maß für die Größe des Loches herangezogen wird. Dies ist vor allem insofern zulässig, da die Kantenlängen aufgrund des Marching-Cubes-Verfahrens im Mittel nur wenig differieren.

Das Schließen der Löcher erfolgt dann mit einem Verfahren, welches ohne Neutriangulierung der Lochkontur auskommt. Die Löcher werden mittels sukzessiver Anwendung der Edge-Collapse-Operation auf die Randkanten der Lochkontur beseitigt. Bei einem Edge-Collapse wird eine Kante im Dreicksnetz eliminiert, indem die beiden Endvertices auf den Mittelpunkt der Kante gezogen und dort verschmolzen werden. Dieses Verfahren ermöglicht insbesondere das elegante Schließen von unebenen Lochkonturen beispielsweise über

aneinander stoßende Flächen. Außerdem müssen auf diese Weise keine neuen Annahmen über die Zugehörigkeiten neu eingefügter Flächen zu den bereits zusammengefassten Regionen getroffen werden. Bei konvexen Lochkonturen bereitet dieses Verfahren keine Probleme. Nicht-konvexe Konturen zu optimieren erfordert, Kanten zu erkennen, die nicht entfernt werden dürfen, um eine konsistente Meshstrutur zu gewährleisten. Zudem dürfen sich die nach dem Edge-Collapse entstehenden Flächen nicht mit anderen Flächen überlappen. Diese Überlegungen wurden bei der Implementierung des Edge-Collapse-Verfahren berücksicht.

3.4 Neutriangluierung

Eine Maßnahme zur Datenreduktion stellt die Neutriangulierung erkannter Flächen dar. Dazu werden die Konturen der im Rahmen der Flächenerkennung geglätteten Flächen extrahiert und anschließend durch einen Tesselator neu trianguliert. Da die Flächen geglättet sind, sind ihre Konturen plan, was eine problemlose Neutriangulierung ermöglicht.

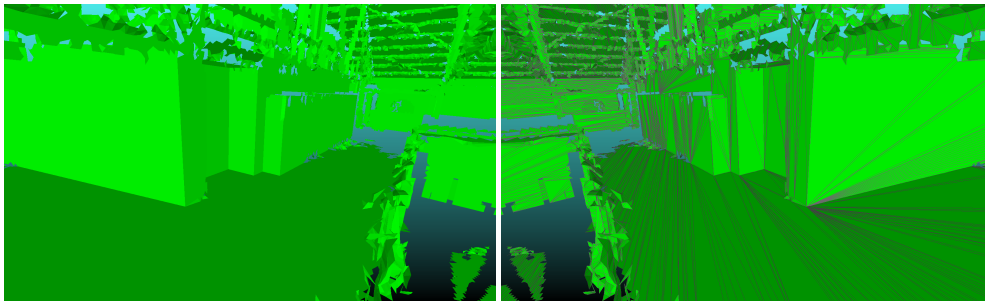

Abb. 5: Rekonstruktion nach dem Schließen der Löcher (links) und Neutriangulierung (rechts). Auf diese Art und Weise wurde eine Vielzahl der kleinen Dreiecke durch wenige große ersetzt.

Kleine Regionen, die nicht geglättet wurden, werden unverändert ins Output-Mesh übernommen. Auf diese Weise bleiben Details erhalten, während redundante Vertices und Dreiecke zusammengefasst werden. In unserer Software kommt zur Triangulierung der ebenen Konturen der OpenGL-Tesselator zum Einsatz. Prinzipiell könnten auch andere Triangluationsalgorithmen (z.B. die aus CGAL) verwendet werden. Es hat sich aber gezeigt, dass der OpenGL-Tesselator für unsere Zwecke sehr gute Ergebnisse liefert und vor allem im Vergleich zu CGAL deutlich schneller arbeitet.

Das Ergebnis der Neutriangulation der ebenen Anteile im Mesh zeigt Abb. 5 beispielhaft am betrachteten Datensatz. Die spitzen Dreiecke sind ein Besonderheit des verwendeten Tesselators. Da OpenGL sog. Triangle-Fans besonders schnell rendern kann, werden diese bevorzugt erzeugt. Wird die Software zur Rekonstruktion von Gebäudeinnenräumen oder bebauten Umgebungen eingesetzt, tritt durch Neutriangulierung eine erhebliche Datenreduktion ein, da in diesen Bereichen üblicherweise große plane Flächen vermehrt auftreten. Die Datenreduktion ist vor allem von Bedeutung, wenn das erzeugte Mesh weiterverarbeitet werden soll. Besonders bei großen Meshes führt die Datenreduktion zu einem deutlich schnelleren Rendering bei der Anzeige. Tabelle 1 stellt die Rekonstruktionen des für die Abbildungen verwendeten Datensatzes mit und ohne Neutriangulierung gegenüber.

Tabelle 1: Eigenschaften der Rekonstruktion ohne und mit Neutriangulierung

	Dateigröße [MB]	Anzahl Punkte	Anzahl Dreiecke
Initiale Punktwolke	132,5	4.253.689	-
Mesh ohne Neutriangulierung	10,8	221.443	371.640
Mesh mit Neutriangluierung	4,5	119.557	98.648

4 Automatisches Erzeugen von Texturen

Bei farbigen Punktwolken oder Punkwolken mit Reflektionswerten besteht die Möglichkeit, Texturen für die erzeugten Meshes durch die Software generieren zu lassen. Hierzu wird für jede neutriangulierte Region eine Textur erzeugt, indem die Kontur der Region abgetastet und die passenden Farbinformationen aus den entsprechenden Stellen der Punktwolke gewonnen werden. Unebene, nicht neutriangulierte Regionen erhalten keine Textur. Die Dreiecke dieser Regionen, die in der Regel recht klein sind, werden jedoch mit Farbwerten aus den entsprechenden Stellen der Punktwolke versehen, sodass insgesamt ein realitätsnah gefärbtes Mesh entsteht.

Als Ausgabeformat für texturierte Meshes steht das Wavefront-OBJ-Format zur Verfügung. Texturierung in den erzeugten Meshes kann je nach Anwendungsfall sinnvoll sein. Neben einem besseren optischen Eindruck für Menschen können Texturen hilfreich sein, wenn Roboter ein Mesh als Karte verwenden, in der sie sich lokalisieren. So ist es möglich, durch eine Auswertung von Kameradaten und einem Vergleich dieser Daten mit der texturierten Karte zusätzliche Anhaltspunkte zur Verbesserung einer Poseschätzung zu erhalten. Abb. 6 zeigt die aus den Remissionswerten der Eingabepunktwolke generierten Texturen auf unserem Referenzdatensatz.

5 Laufzeitverhalten und Parametrisierung

Die Laufzeit der Rekonstruktion hängt stark von den Eingabeparametern wie etwa der Größe der Gitterzellen des Raumgitters ab. Tabelle 2 zeigt beispielhaft die Laufzeiten des in den Abbildungen 1 bis 6 sichtbaren Datensatzes mit einer Datenmenge von ca. 4,3 Mio. Punkten. Hierbei hat die die Generierung der Texturen einen Großteil der Gesamtlaufzeit in Anspruch genommen.

Tabelle 2: Eigenschaften der Rekonstruktion ohne und mit Neutriangulierung

	Laufzeit [mm:ss]
Initiales Mesh entsprechend Abb. 1	01:14
Laufzeit nach Optimierungen entsprechend Abb. 4	01:28
Mit Triangulierung und Texturgenerierung entsprechend Abb. 6	03:54

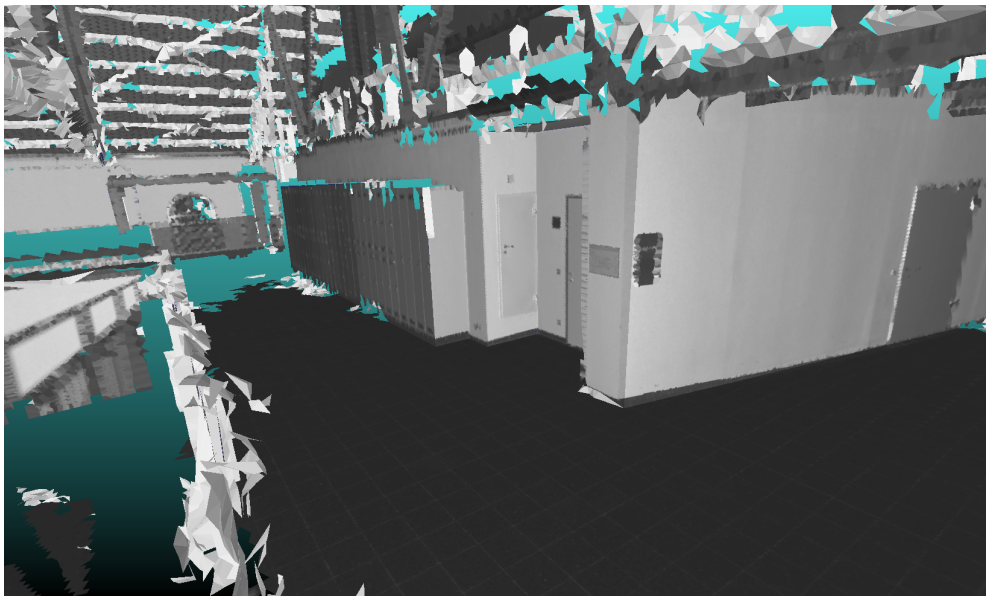

Abb. 6: Das neutriangulierte Mesh wurde mit Texturen versehen, die aus den Remissionsdaten des Laserscanners generiert wurden.

6 Zusammenfassung

Mit dem Las Vegas Reconstruction Toolkit steht eine Open-Source-Software zur Verfügung, die sowohl Rekonstruktion als auch Meshoptimierung und Texturierung vollautomatisch durchführen kann. Die in der Software implementierten Algorithmen wurden für Daten mit vielen ebenen Anteilen optimiert, wie sie z.B. beim Scannen von Gebäuden oder bebauten Flächen vermehrt auftreten. Alle Verfahren laufen nach der Parameterfindung vollautomatisch ab und benötigen keine Benutzerinteraktion.

Literatur

Lorensen W. E. & Cline H. E: *A High Resolution 3D Surface Construction Algorithm*, ACM SIGGRAPH, 1987

Hoppe H., DeRose T., Duchamp T., McDonald J., Stuetzle W.: *Surface Reconstruction from Unorganized Points*, Computer Graphics 26(2), 1992

Rusu R. B. & Cousins S.: *3D is Here: Point Cloud Library (PCL)*, Conference on Robotics and Automation, 2011

Payne B. & Toga A., *A Surface Mapping Brain Function on 3D Models*, IEEE Computer Graphics and Applications, 1990