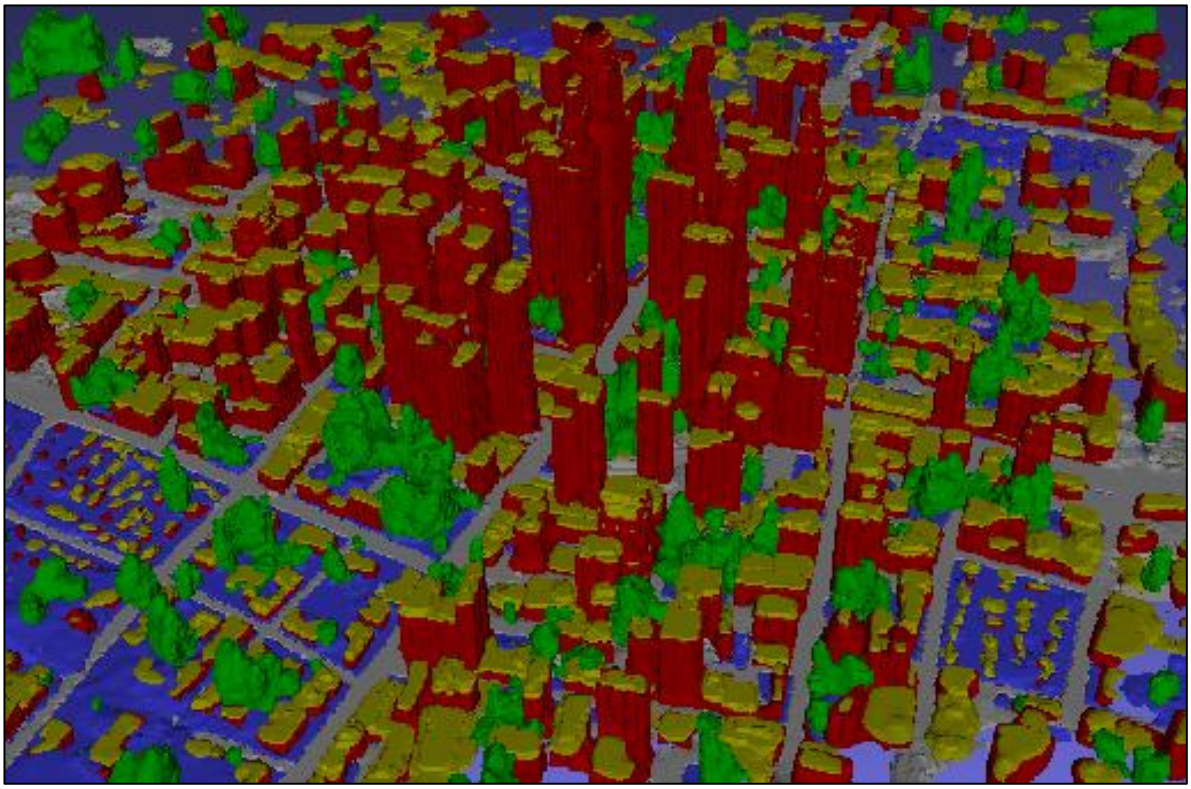
ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

IGP
Institute of Geodesy and Photogrammetry

Semantische Analyse von 3D Stadtmodellen



Bachelor-Arbeit Frühlingssemster 2016

Studiengang: Geomatik & Planung

Student: Luca Miotti

Betreuung: Maros Blaha

Leitung: Prof. Dr. Konrad Schindler

Diese Bachelor-Arbeit befasst sich mit der semantischen Analyse und Weiterverarbeitung von dreidimensionalen Stadtmodellen. Solche Modelle unterscheiden sich von den herkömmlichen Modellen aus der Photogrammetrie insofern, dass sie neben der Modellierung der Objekte diese auch semantisch klassifizieren. Der zur Generierung der Modelle verwendete Algorithmus wurde erst kürzlich an der ETH Zürich entwickelt. Er beruht auf einer Octree-Datenstruktur und teilt die Objekte momentan in die Klassen Gebäude, Boden, Vegetation, Dach und Rest (Clutter) ein.

Der Inhalt der Arbeit ist in zwei Teile aufgeteilt. Im ersten Teil wird die Genauigkeit des Algorithmus einerseits mit einem ähnlichen Algorithmus, der auf eine Grid-Datenstruktur setzt und andererseits mit einer künstlich generierten Groundtruth verglichen. Man versucht so die Frage zu beantworten ob es Unterschiede in Genauigkeit und Vollständigkeit zwischen Octree und Grid gibt, wie gross diese Unterschiede sind und ob dahinter eine Systematik besteht. Zudem soll die absolute Genauigkeit des Octree-Modells durch den Vergleich mit der Groundtruth bestimmt werden

Auch beim zweiten Teil steht die Abwägung von Genauigkeit bzw. in diesem Fall Detailgrad und Speicherplatzbedarf als Motiv im Vordergrund. Bei Objekten, deren Oberfläche aus einer Triangulierung besteht, erfolgt eine Einsparung in Speicherplatz indem die Anzahl Dreiecke der Triangulierung reduziert wird. Um dies zu erreichen werden bestimmte Eckpunktpaare der Triangulierung auf einen einzelnen Punkt zusammengelegt und das Modell wird neu trianguliert. Das Ziel ist nun dies so durchzuführen, dass das Modell möglichst wenig an Detailgrad verliert. Es wird ein Algorithmus implementiert, welcher die Triangulierung analysiert, berechnet welche Stellen gut für die Vereinigung geeignet sind und dies anschliessend visualisiert.

Inhaltsverzeichnis

Abbildungsverzeichnis

1. Verwendete Software

Im Rahmen dieser Arbeit wurden die folgenden Programme verwendet.

Geomagic Control

Geomagic Control ist ein Programm von Geomagic zur grafischen bzw. numerischen Analyse von 3D Modellen und deren Weiterverarbeitung. Es wurde in dieser Arbeit vor allem für die Berechnung von 3D Abweichungen zweier Modelle verwendet. [Geomagic 2015]

Matlab

Matlab ist eine proprietäre Programmiersprache der Firma The MathWorks. Sie wird zur Lösung mathematischer Probleme und anschliessender grafischer Darstellung der Ergebnisse verwendet. Das System von Matlab beruht auf Berechnungen mithilfe von Matrizen. [https://de.wikipedia.org/wiki/Matlab]

Für den Austausch von Funktionen zwischen verschiedenen Nutzern existiert die offizielle Website File Exchange. Für diese Arbeit wurden folgende Funktionen dieser Bibliothek verwendet:

- Helmert3D: Funktion zur Bestimmung der Parameter einer dreidimensionalen Helmert Transformation mittels allgemeiner Ausgleichung.
- Compute_edges: Funktion zur Berechnung der Kanten aus einer Liste von Dreiecken und dazugehörenden Eckpunkten.

Meshlab

MeshLab ist ein Freeware-Programm zum Darstellen, Bearbeiten, Filtern und Rendern von Triangulierungsnetzen von 3D-Modellen. [Meshlab, 2016]

2. Einleitung

2.1. Theoretische Grundlagen und Motivation

In der Photogrammetrie bzw. Bildverarbeitung wurden die Bereiche der 3D Rekonstruktion und der semantischen Bildsegmentierung bis anhin grösstenteils unabhängig voneinander erforscht. In der letzten Zeit wurde in der Forschung nach Möglichkeiten gesucht, um diese beiden Domänen synergetisch zu verbinden. Die Vorteile dieser Synthese sind vielseitig. Aus semantischen Informationen lässt sich oft auf geometrische Details schliessen und umgekehrt. Weiss man zum Beispiel, dass eine Fläche Teil des Bodens ist, wird diese mit grosser Wahrscheinlichkeit horizontal und eben sein. Umgekehrt kann man von einer horizontalen Fläche darauf schliessen, dass es sich bei dem Objekt höchstwahrscheinlich um Boden und nicht um eine Wand handelt.

Ein entsprechender Algorithmus, der Geometrie und Semantik miteinander verknüpft, wurde kürzlich an der ETH Zürich entwickelt. Dieser berechnet, mithilfe von Tiefenkarten und klassifizierten Bildern, ein semantisches volumetrisches Modell. Bei den einzelnen Voxeln wird nun nicht nur zwischen Objekt und Luft unterschieden, die Klasse Objekt wird zusätzlich in verschiedene Kategorien unterteilt.

Mathematisch wird das Problem als Energiefunktion formuliert. Durch die Minimierung dieser Energie wird für jedes Voxel ausgerechnet, wie wahrscheinlich es zu einer bestimmten Klasse gehört. Das Voxel wird dann derjenigen Klasse mit der höchsten Wahrscheinlichkeit zugeordnet. Die Energiefunktion setzt sich als Summe aus zwei verschiedenen Teilen zusammen und lässt sich vereinfacht ausdrücken als:

$$E = E_{Datenterm} + E_{geometrischer\ Prior}$$

Der Datenterm berechnet sich aus den Input Daten. Er betrachtet zum einen die Differenzen zwischen der optimierten Oberfläche des Modells und der Tiefenbeobachtungen, zum anderen werden die Wahrscheinlichkeiten aus den klassifizierten Luftbildern berücksichtigt. Der geometrische Prior kann für jede Klasse individuell definiert werden und ist eine Kostenfunktion abhängig von einer bevorzugten Richtung für den Normalenvektor. Bei der Klasse Boden etwa geht man davon aus, dass der Normalenvektor parallel zur Z-Achse ist. Die Kosten fallen nun umso höher aus, je stärker der tatsächliche Normalenvektor vom erwarteten Vektor des Priors abweicht, wie in der folgenden Figur dargestellt ist.

Mit dieser Methode lässt sich beispielsweise vermeiden, dass es beim Übergang von Fassade zum Boden zu einer Glättung bzw. Abrundung der Kanten kommt.

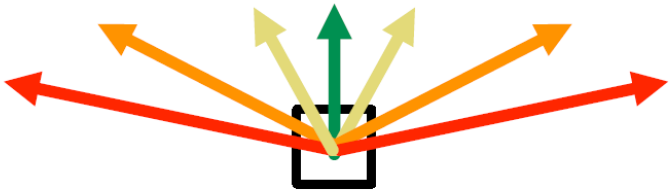


Abbildung 1: Prior der Klasse Boden [Blaha et al. 2016]

Der oben beschriebene Ansatz wurde erstmals von [Haene et al. 2013] publiziert. Als Datenstruktur wird hier ein reguläres Voxel-Grid mit gleich grossen Voxeln verwendet. Da in den meisten Fällen ein grosser Teil der modellierten Szene homogen ist (z.B. Luft), wurde der Fokus in [Blaha et al. 2016] darauf gesetzt, den Algorithmus auf einen sogenannten Octree umzuformulieren.

Durch die neue Datenstruktur entstehen viele Vorteile. Das Modell verbraucht deutlich weniger Speicherplatz, weil grosse Gebiete konstanter Klasse nun mit einem grossen anstatt vielen kleinen Voxeln dargestellt werden können. Dadurch kann man grössere Gebiete bei gleichem Speicherverbrauch modellieren, was bei der Rekonstruktion von Städten ein Vorteil ist. Weiter erlaubt dies auch eine deutliche schnellere Prozessierung des Ganzen. Zusammengefasst kann man das Prinzip der untenstehenden Abbildung entnehmen.

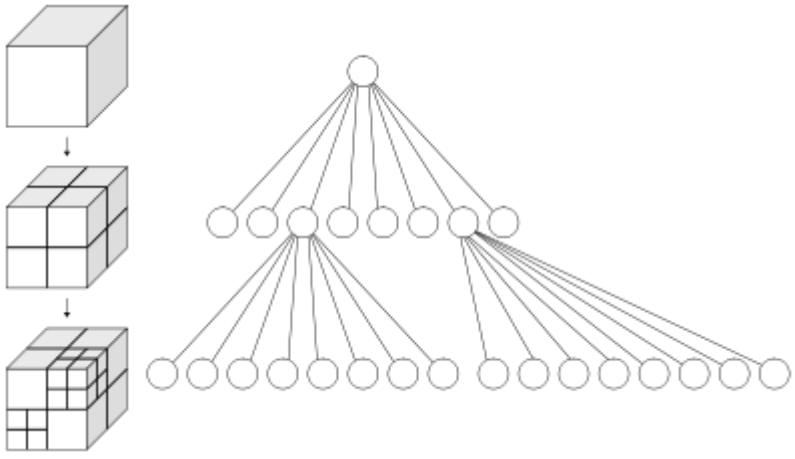


Abbildung 2: Visualisierung des Octree-Prinzips [https://en.wikipedia.org/wiki/Octree]

Da die Methoden [Haene at al. 2013] und [Blaha et al. 2016] nicht identisch sind, können die resultierenden Modelle geringe Diskrepanzen aufweisen. Wie gross die Abweichungen sind und ob bei gewissen Klassenübergängen systematische Fehler aufzufinden sind, wird im ersten Teil dieser Arbeit analysiert.

3D Modelle werden generell über ihre Oberfläche repräsentiert. Das gilt auch für volumetrische Modelle wie oben beschrieben, bei denen grundsätzlich die Oberfläche an den Klassengrenzen extrahiert wird (ausser im Fall Luft). Vereinfacht ausgedrückt werden Punkte an der Oberfläche als Eckpunkte für die Triangulierung benutzt. Je höher die Anzahl der einzelnen Dreiecke ist, desto höher auch der Detailgrad des Modells. Will man diesen adäquat reduzieren um Speicher zu gewinnen, muss man Dreiecke zusammenlegen und damit deren Anzahl zu reduzieren. Dies ist etwa bei Flächen angebracht, welche mit vielen kleinen Dreiecken repräsentiert sind. Ecken oder Kanten hingegen sollten erhalten werden, um nicht die Charakteristik des Modells zu verlieren.

Um diesen Aspekt zu untersuchen, werden im zweiten Teil dieser Arbeit Teile des Algorithmus „Surface Simplification Using Quadric Error Metrics“ von [Garland et al. 1997] implementiert. Dieser geht iterativ vor und bestimmt in jeder Iteration jeweils alle geeigneten Ecken für eine Zusammenlegung basierend auf einem Fehlerwert. Für die Ecke eines Hauses würde dieser beispielsweise sehr hoch ausfallen, für ein Stück mitten auf dem Boden umso niedriger. Ein bestimmter Anteil der Eckpunkte mit den niedrigsten Werten wird nun zusammengelegt, das Modell wird neu trianguliert und die nächste Iteration beginnt. Dies wird solange durchgeführt, bis man das Modell um eine gewünschte Anzahl von Dreiecken reduziert hat.

2.2. Ziele der Arbeit und Vorgehen

Die Ziele dieser Arbeit lassen sich in zwei Teilbereiche gliedern. Der erste Bereich analysiert die Qualität der Ergebnisse von [Blaha et al. 2016] und vergleicht sie mit denjenigen von [Haene et al. 2013]. Als Testumgebung dient eine synthetische Stadt von [Cabezas et al. 2015] für die ein sogenanntes Groundtruth Modell zur Verfügung steht. Mit dem Programm Geomagic Control lassen sich dazu 3D Abweichungen berechnen und grafisch darstellen.

Für den zweiten Teil werden die ersten Schritte des oben angesprochenen Algorithmus implementiert um die erste Iteration zu berechnen und anschliessend zu visualisieren. Dies beinhaltet vor allem die namensgebende Quadric Error Metric. Dies wird dann einerseits auf verschiedene Klassen eines kleinen Teilgebietes des semantischen 3D Modells, sowie andererseits auf weit verbreitete Modelle, wie dem Stanford Buddha [http://graphics.stanford.edu/data/3Dscanrep/], der 3D Modellierung angewendet. Auch werden gleiche Modelle mit verschiedener Auflösung verwendet. Anhand der grafischen Resultate sollte dann klar erkennbar sein an welchen Orten eine Glättung der Oberfläche Sinn macht. Erwartungsgemäss sollten dies glatte Flächen sein.

Eine Durchführung von weiteren Iterationen sowie die Anpassung des Programms, mit welcher die semantischen Informationen ausgenutzt werden würden, überschreitet den Rahmen dieser Bachelorarbeit.

Vorgehen:

- Durchführung der Vergleiche von [Blaha et al. 2016] und [Haene et al. 2013] im Geomagic Control
- Analysieren der Vergleiche
- Teilimplementation des Algorithmus von [Garland et al. 1997]
    - Berechnung der Quadric Error Metrics
    - Visualisierung der Resultate
- Testen des Algorithmus mit verschiedenen Objekten unterschiedlicher Struktur
- Analysieren der Resultate

3. Genauigkeitsanalyse

3.1. Generierung der Daten

Das Abschätzen der Genauigkeit von 3D Modellen gestaltet sich oft schwierig, da in den meisten Fällen keine Groundtruth zum Vergleich vorhanden ist. Dies trifft insbesondere bei grossen Modellen, wie etwa demjenigen einer Stadt zu. Um einen zutreffenden Vergleich für diese Arbeit zu ermöglichen dient als Vorbild nicht etwa eine reale Stadt, sondern ein künstliches Modell, von welchem man die Grössenverhältnisse genau kennt. Die Bilder und Tiefenkarten der Stadt erhielt man aus einem Algorithmus der einen Flug über dieser virtuellen Stadt simulierte und davon 90 Bilder erzeugt hat. Erstellt wurde die Stadt mit der Cityengine, ein Produkt von Esri. Leider ist so keine wirklich realistische Szenerie möglich und Probleme, welche in der Praxis auftauchen – wie etwa das Sensorrauschen der Kamera – müssen ignoriert werden. Im Rahmen dieser Arbeit sollte dies aber keine grosse Rolle spielen.

Weil die Koordinatensysteme der Groundtruth sowie der erzeugten Modelle nicht übereinstimmen müssen diese, bevor weitere Schritte unternommen werden können, zuerst in das gleiche System transformiert werden. Es wurde hier entschieden, dass das finale System, dasjenige der Modelle sein soll. Die Transformation erfolgt als 3D-Helmert Transformation und besteht mit 3 Translationen, 3 Rotationen und einer Skalierung also aus 7 zu bestimmenden Parametern. Die für diese Berechnung benötigten Punkte erhält man aus den 90 erzeugten Bildern. Die überflogenen Koordinaten zum Zeitpunkt der Generierung liefern direkt die Punkte für die Groundtruth. Da die Koordinaten der Groundtruth in einem unterschiedlich ausgerichteten System vorliegen müssen diese zuerst noch um die X-Achse rotiert werden. Dies erfolgt mit einer gewöhnlichen Matrix Multiplikation in Matlab.

$$\begin{matrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{matrix} * [Koordinaten\ der\ GT\ Punkte]^{T}$$

Die entsprechenden Punkte des Modellkoordinatensystems erhält man aus der relativen Orientierung der jeweiligen Kamerastandpunkte, welche jeweils aus einer Rotation R (3x3) und einer Translation t (3x1) bestehen. Die gesuchten Punkte berechnen sich dann wieder in Matlab wie folgt:

$$Punkte = -R' * t$$

Ist dies geschehen haben wir die benötigten Punkte um dank Überbestimmung per Allgemeiner Ausgleichung die gewünschten Helmert-Parameter zu bestimmen. Dafür wird aus der Internetbibliothek des Matlab Herstellers Mathworks das Programm Helmert3D verwendet. Diese Funktion erwartet als Input Daten zweimal eine n x 3 Punktematrix, sowie eine Approximation der gesuchten Rotationswinkel. Nötig ist dies damit das Programm nach einer bestimmten Anzahl Iterationen konvergiert und nicht nach einer gewissen Zeit mit einem falschen Ergebnis terminiert. Da keine genauen Angaben über die Rotationswinkel vorhanden sind, wurden diese einfach auf je π/100 geschätzt.

Die Berechnungen werden durchgeführt und man erhält die 7 ausgeglichenen Helmert-Parameter. Um die Genauigkeit der Berechnungen zu überprüfen werden die Punkte der Groundtruth mit den eben bestimmten Parametern in das Modellsystem transformiert und es werden die Residuen gebildet. Die Mittelwerte der Residuen liegen im Bereich des numerischen Fehlers und die Standardabweichungen betragen nur ein Bruchteil der Grösse eines Voxels - in diesem System etwa 0.04, also ungefähr Faktor 10. Die geschätzten Parameter liegen also durchaus im akzeptablen Genauigkeitsbereich.

Werte der Helmert-Parameter	
Translation x	0.52687
Translation y	0.88588
Translation z	-8.38691
Massstab	0.01305
Rotation x (Grad)	0
Rotation y (Grad)	0
Rotation z (Grad)	111.99609

Standardabweichungen der Residuen	
x	0.00386
y	0.00365
z	0.00155

Abbildung 3: Werte der Transformation und Standardabweichungen aus der Sicht des Groundtruth-Koordinatensystems – einheitenlos und auf 5 Nachkommastellen gerundet

Mittels des Freeware Programmes Meshlab wird nun die gesamte Groundtruth in das Koordinatensystem der Modelle transformiert. Weil die Groundtruth nicht klassifiziert ist und nur aus einem binären Gesamtmodell besteht, werden wiederum in Meshlab die Klassen der Modelle zu einem Gesamtmodell vereinigt. Erst so sind aussagekräftige Vergleiche mit der Groundtruth möglich. Schlussendlich werden alle Daten noch so zugeschnitten, dass sie nur einen kleineren Bereich in der Stadtmitte abbilden. In diesem Ausschnitt sind alle interessanten sowie relevanten Aspekte der Stadt abgebildet. Durch die kleinere Datenmenge reduziert sich zudem auch die Berechnungszeit der Vergleiche. Nachfolgend das komplette, unbearbeitete Octree-Modell.

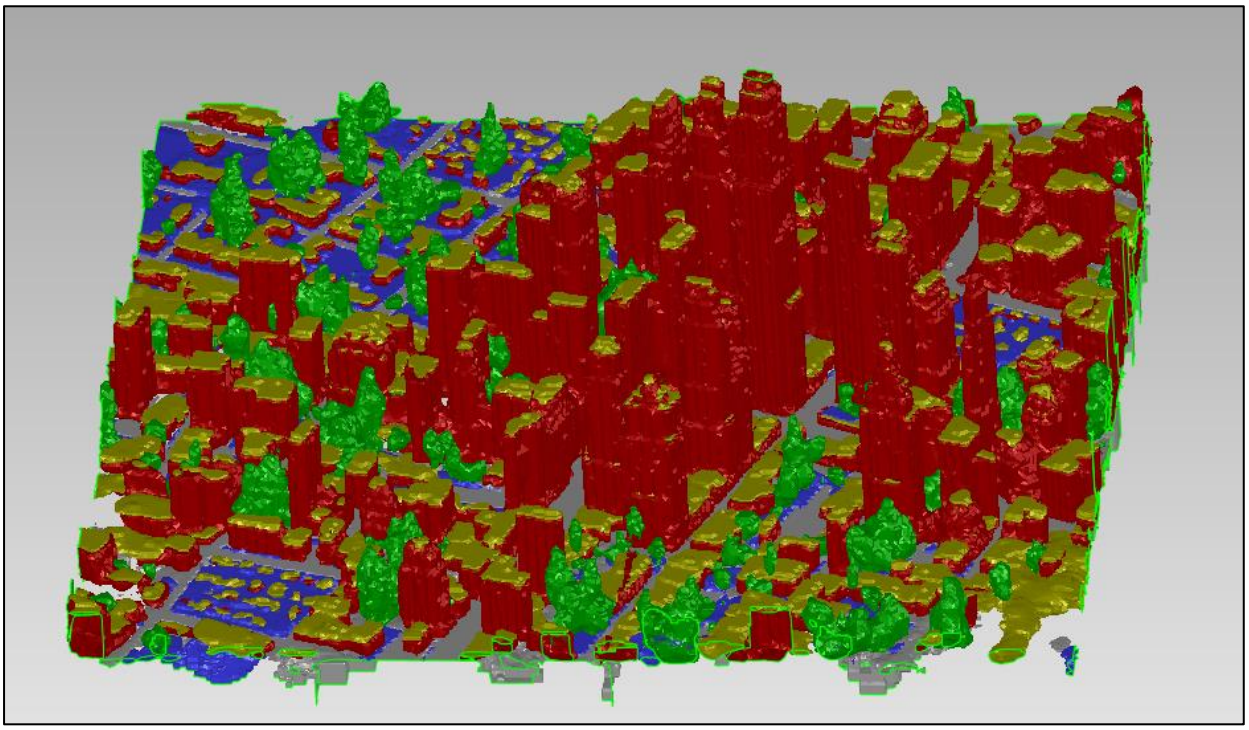


Abbildung 4: Zugeschnittenes Octree-Modell

3.2. Theoretische Grundlagen der Vergleiche

Die generierten Daten werden nun mit dem Programm Geomagic Control verglichen. Dieses Programm erlaubt die Bestimmung der 3D-Abweichung zwischen zwei binären 3D-Modellen, berechnet die Standardabweichung der Differenz und stellt die Unterschiede grafisch dar. Es wird ein Referenz- und ein Testmodell definiert. Als Referenz wird für diese Arbeit jeweils Octree bzw. die Groundtruth gewählt, Grid ist immer das Testmodell. Die Unterschiede werden nun je nach ihrem Ausmass farbig dargestellt. Liefert das Testmodell in einem bestimmten Gebiet nicht genügend Daten für einen sinnvollen Vergleich oder liegt die Differenz über einem bestimmten Grenzwert wird der Vergleich grau eingefärbt. Dieser Maximumsunterschied (wird im Vergleich rot bzw. dunkelblau dargestellt) dient zur Skalierung der Farbskala und wird vor der Durchführung des Vergleichs festgelegt. Die folgenden Vergleiche werden mit der einfachen Voxelgrösse, fünffachen Voxelgrösse und zehnfachen Voxelgrösse als Grenzwert durchgeführt.

Als Vergleichsmethode wurde „3D Deviation“ gewählt. Diese Methode sucht für die Bestimmung der Abweichung jeweils immer die kürzeste Distanz von einem Punkt im Testmodell zur Referenz. Durch die Natur der Voxelmodellierung, bei der kleinere Verschiebungen im Subvoxelbereich durchaus möglich sind, ist diese Methode einer reinen vertikalen Abweichung vorzuziehen. [Geomagic 2015]

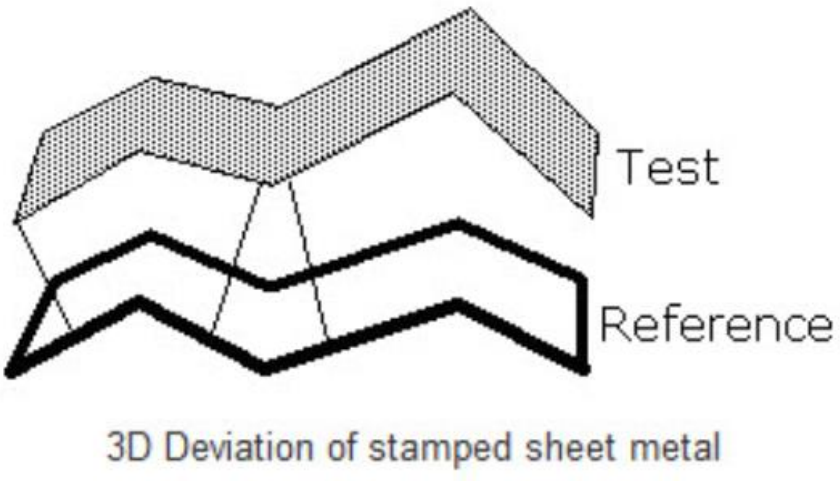


Abbildung 5: Prinzip der Differenzbestimmung [Geomagic 2015]

3.3. Octree vs. Grid

Wie schon angesprochen besteht das in dieser Arbeit verwendete, semantische Modell aus fünf Klassen. Gebäude, Boden, Dach, Vegetation und Clutter. Da Clutter eine Sammelklasse für Voxel ist, die keiner anderen Klasse zugeordnet werden konnten und so für beide Algorithmen recht unterschiedlich ausfallen kann, wird diese Klasse für sämtliche Vergleiche ignoriert.

Es werden nun also alle 4 verbleibenden Klassen im Octree und Grid Modell miteinander verglichen. Für jede Klasse werden drei Vergleiche durchgeführt. Diese drei Vergleiche unterscheiden sich voneinander insofern, dass der grösste grafisch dargestellte Unterschied anders gewählt wird. Und zwar wurden diese Beschränkung auf einen Voxel, fünf Voxel und zehn Voxel festgelegt. Sollten die Modelle sehr ähnlich zueinander sein wird man zwischen fünf und zehn Voxel wenige bis keine Unterschiede mehr sehen. Je nach dem sind dann weitere Vergleiche nicht mehr nötig und man kann zur Analyse nur den ersten Vergleich verwenden. Umgekehrt wird man bei eher unterschiedlichen Modellen starke Änderungen zwischen den Vergleichen feststellen. Natürlich wäre es vorzuziehen, für alle Klasse die gleichen Grenzwerte gebrauchen zu können. So wäre es möglich die Modellierung der einzelnen Klassen untereinander zu vergleichen.

Die folgenden Zahlenwerte sind jeweils aus der Sicht des relativen Modell-Koordinatensystems. Distanzmessungen in den Modellen und vergleichbare Messungen in der Realität haben ergeben, dass eine Voxelgrösse grob geschätzt etwa einem Meter entspricht. Da man im relativen Koordinatensystem eine Voxelgrösse von 0.04 hat, würde also eine Einheit im Modell-Koordinatensystem ungefähr 25 Meter in der Realität entsprechen.

3.3.1. Gebäude

Die Klasse Gebäude ist wohl die wichtigste Klasse in einem Stadtmodell. Nicht nur braucht sie wohl speichertechnisch den meisten Platz, sondern definiert wohl die Charakteristik des Gesamtmodells wie keine andere Klasse. Zudem besteht sie nicht nur, wie etwa Boden oder Dach, aus grösstenteils ebenen Flächen.

Umso erfreulicher ist es, dass das zu testende Octree-Modell hier im Vergleich zum Grid-Modell keine grossen Unterschiede aufzeigt.

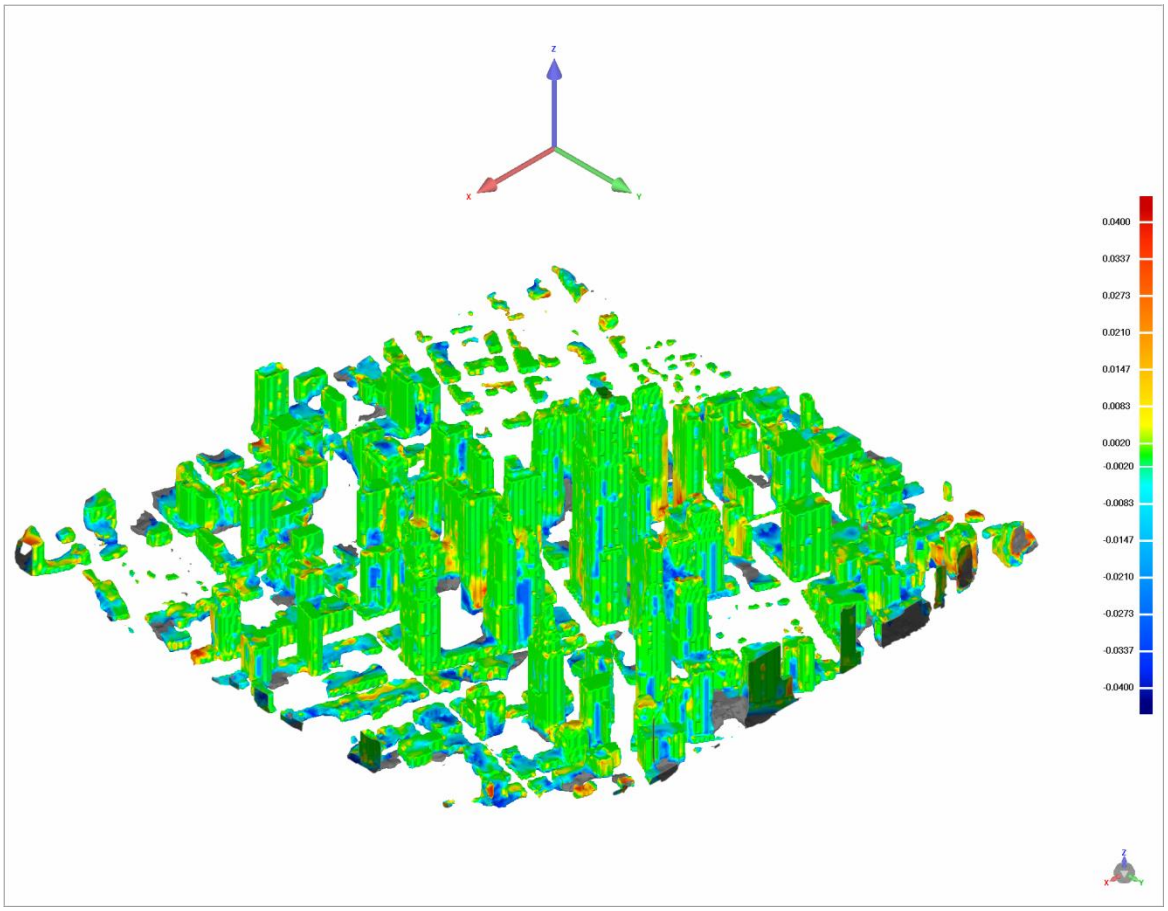


Abbildung 6: 3D-Compare der Klasse Gebäude mit grafischem Grenzwert auf eine Voxelgrösse festgelegt

Wie man der Abbildung entnehmen kann sind Octree und Grid bei der Klasse Gebäude sehr ähnlich. Der grösste Teil der Differenzen beläuft sich auf weniger als ein Zehntel der Voxelgrösse und nur bei 10 Prozent der Oberfläche liegen die beiden Modelle mehr als eine Voxelgrösse auseinander. Die Standardabweichung der Differenzen beläuft sich auf 0.0092, etwa einem Viertel eines Voxels. Einzig in Bodennähe scheint es nennenswerte Abweichungen zu geben. Grundsätzlich scheint es so, dass das Octree Modell hier häufig über dem Grid Modell liegt. Klare Systematiken sind aber nicht zu erkennen.

Auch ein Blick auf den Querschnitt unterstützt die Annahme, dass nennenswerte Abweichungen vor allem in Bodennähe auftreten. Aber auch hier kann man keine generellen Aussagen über die Natur der Abweichungen machen.

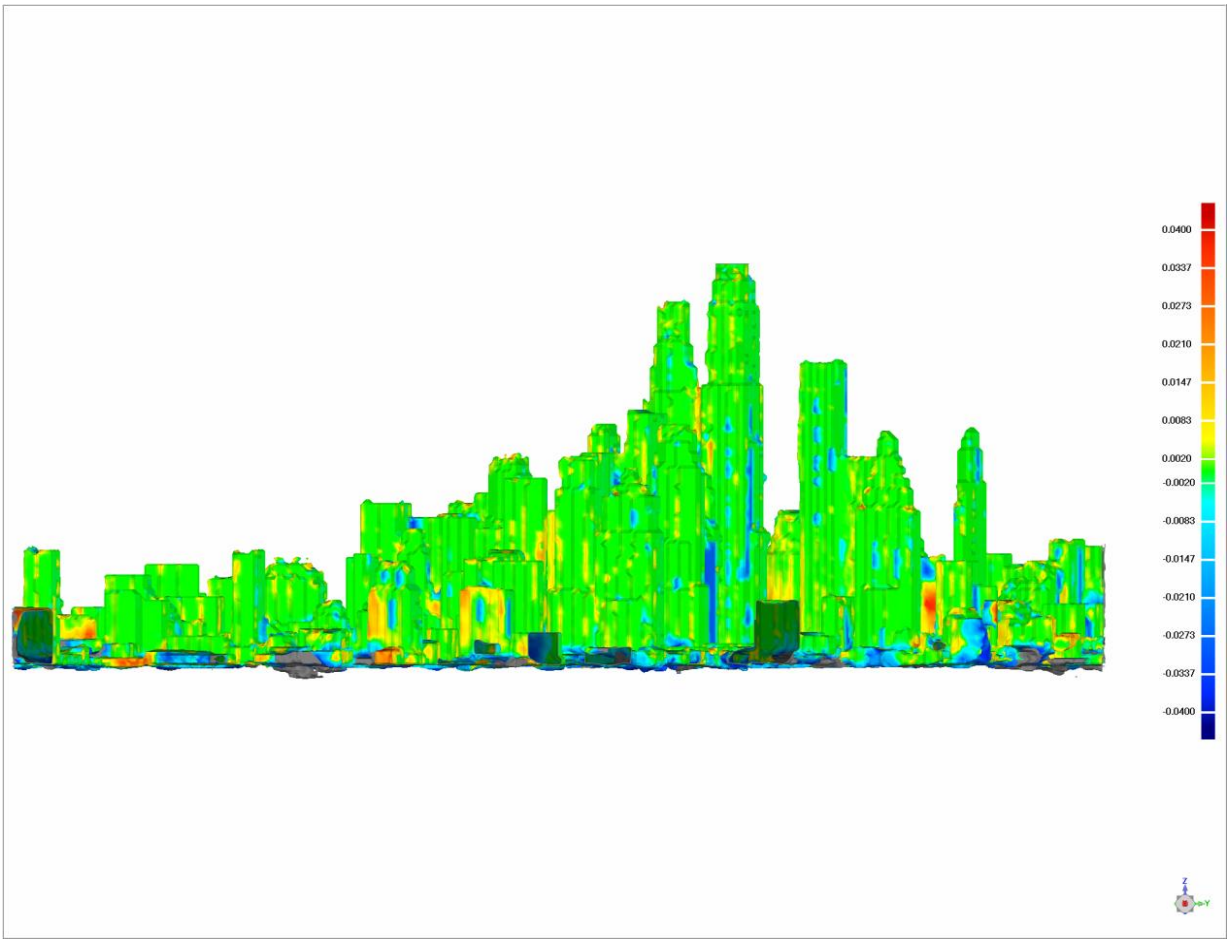


Abbildung 7: Querschnitt des Vergleichs der Klasse Gebäude

Da der Vergleich mit dem Grenzwert einer Voxelgrösse sehr aussagekräftig ist und der grösste Teil der Differenzen unterhalb dieses Wertes liegt, kann auf die Analyse weiterer Vergleiche hierzu verzichtet werden.

3.3.2. Boden

Der Vergleich zur Klasse Boden gestaltet sich nicht ganz einfach. Wenn man sich die beiden Modelle vor dem Vergleich anschaut, erkennt man, dass der Octree-Algorithmus den Boden offenbar deutlich umfangreicher modelliert. Bei keiner anderen Klasse ist der Grad der Modellierung auch nur ansatzweise so unterschiedlich. Daher wird bei jedem durchgeführten Vergleich der Anteil, der nicht verglichen werden kann, sehr gross sein.

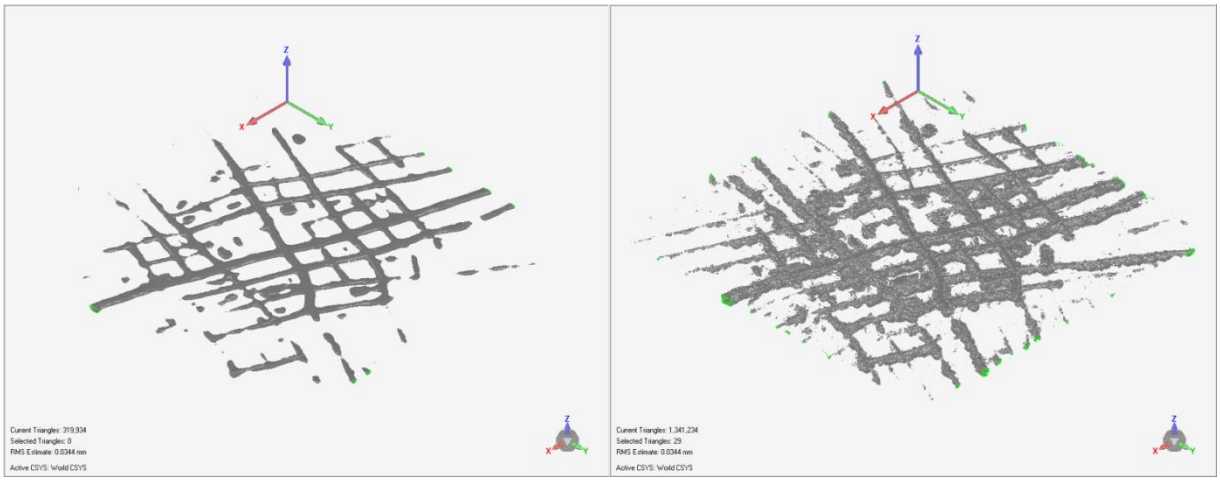


Abbildung 8: Differenz der Modellierung der Klasse Boden. Links Grid - rechts Octree

Führt man jetzt einen Vergleich durch hat man offensichtlich einen sehr hohen Anteil, welcher nicht verglichen werden kann. Abhängig vom Grenzwert sind dies zwischen 55 und 15 Prozent. Dementsprechend fällt auch die Standardabweichung mit 0.0123 relativ hoch aus. Da diese Daten in diesem Fall aber nicht wirklich signifikant sind, wurde der Grenzwert wieder auf 1 Voxelgrösse festgelegt.

Anhand der folgenden Abbildung erkennt man dann auch, dass der Teil, welcher von beiden Algorithmen modelliert wird, nahezu identisch ist. Die wenigen erkennbaren Unterschiede zeigen aber, dass das Octree-Modell an den meisten Stellen wieder leicht über dem Grid-Modell liegt.

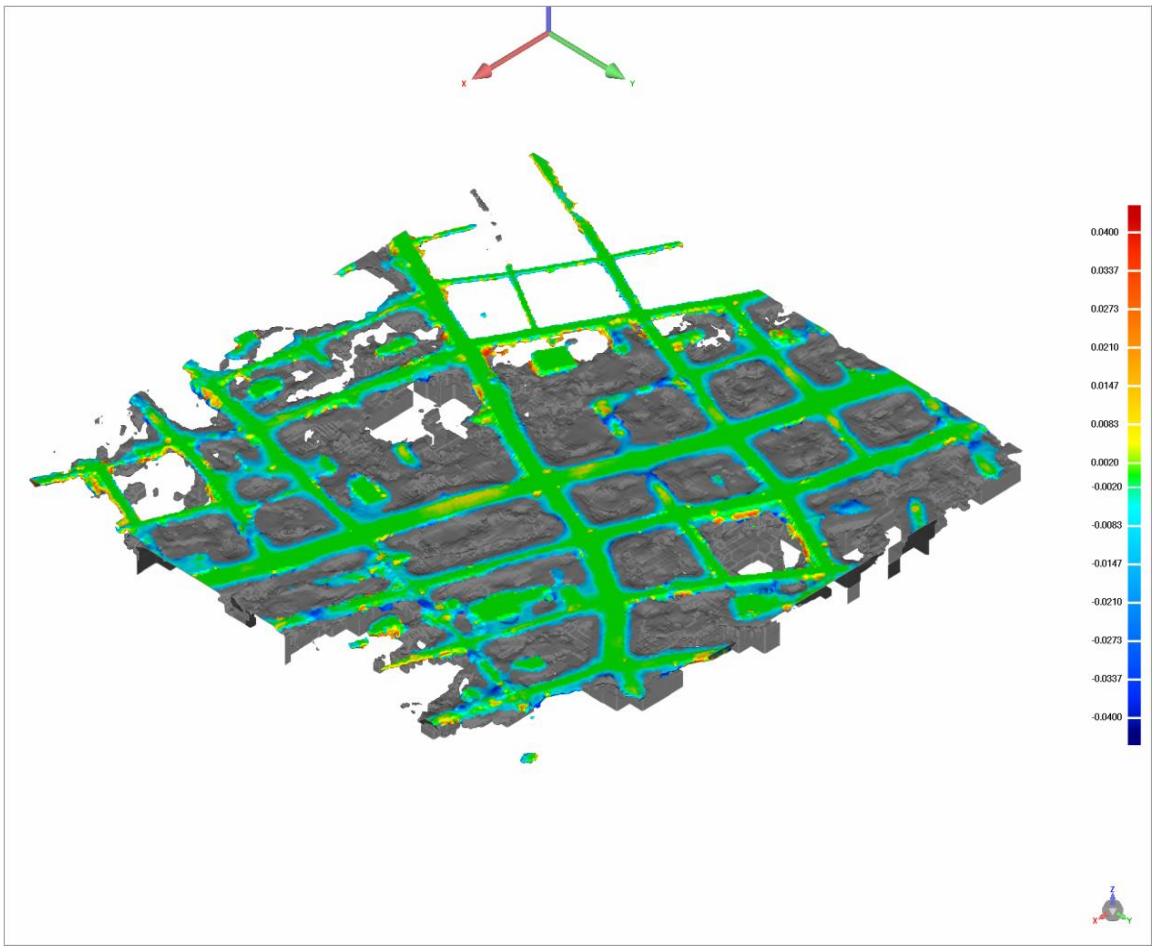


Abbildung 9: 3D-Compare der Klasse Boden mit grafischem Grenzwert auf eine Voxelgrösse festgelegt

3.3.3. Dach

Die Klasse Dach ist in ihrer Geometrie ähnlich simpel wie die Klasse Boden. Da der Grid – sowie Octree-Algorithmus die Klasse zum grössten Teil im gleichen Umfang modelliert ist es nicht überraschend, dass die Modelle sehr gut übereinstimmen. Auch der Anteil der Modelle, der wegen zu grosser Abweichungen nicht verglichen werden kann beträgt weniger als 10 Prozent. Zudem liegt die Standardabweichung der Abweichung mit 0.0086, weniger als einem Viertel der Voxelgrösse, relativ tief.

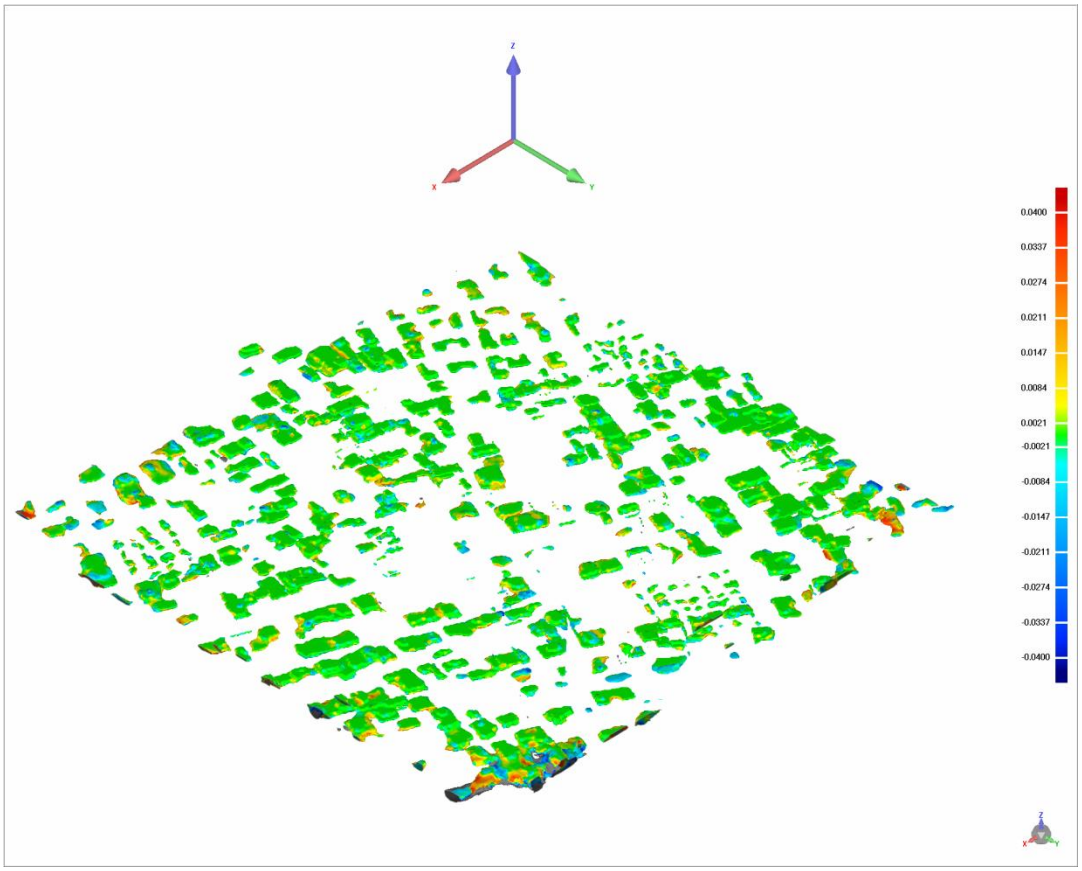


Abbildung 10: 3D-Compare der Klasse Dach mit grafischem Grenzwert auf eine Voxelgrösse festgelegt

Wie man erkennt ergibt der Vergleich ein sehr gutes Resultat. Wirkliche Unterschiede sind kaum auszumachen. Lediglich am Rand des Modells treten welche auf. Dies liegt aber wohl an der Modellierung, denn systematische Abweichungen sind keine zu erkennen. Auch hier sind wegen der guten Ergebnisse keine weiteren Vergleiche mehr nötig.

3.3.4. Vegetation

Eine Modellierung der Vegetation gestaltet sich im Vergleich zu den anderen Klassen offensichtlich deutlich schwieriger. Während man bei den anderen Klassen meistens auf einfache Geometrien zurückgreifen kann, gilt dies für die Vegetation nicht mehr. Die zu vergleichenden Algorithmen haben bei der Modellierung so einen grösseren Spielraum. Unter anderem sieht man dies daran, dass der Umfang der Modelle wieder recht unterschiedlich zu sein scheint. Der Octree-Algorithmus geht stark ins Detail, wo hingegen der Grid-Algorithmus kleinere Objekte gar nicht erst modelliert. Insofern ist zu erwarten, dass bei dieser Klasse die signifikantesten Unterschiede auftauchen werden.

Setzt man den Grenzwert, der gerade noch darzustellenden Unterschiede, auf eine Voxelgrösse, sind 13 Prozent der Unterschiede zwischen den Modellen nicht in diesem Rahmen. Wie aber schon erwähnt liegt dies vor allem daran, dass der Grad der Modellierung unterschiedlich ausfällt. Die Standardabweichung der Differenzen liegt mit 0.0119 im akzeptablen Bereich. Darum wird auch hier wieder der Vergleich mit dem kleinsten Grenzwert verwendet und auf die Analyse weiterer Vergleiche verzichtet.

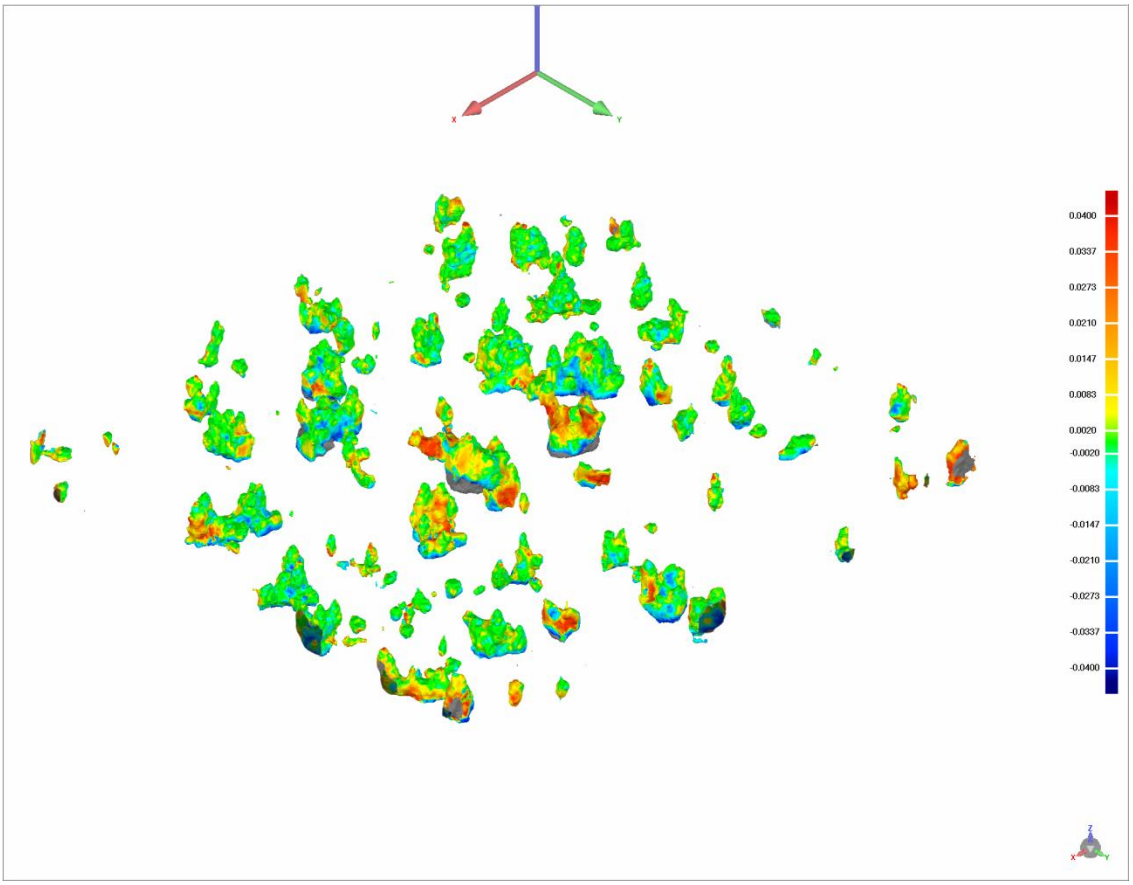


Abbildung 11: 3D-Compare der Klasse Vegetation mit grafischem Grenzwert auf eine Voxelgrösse festgelegt

Wie vermutet existieren hier grössere Differenzen als bei den vorherigen Klassen. Zwar stimmt vieles gut überein, doch gibt es auch etliche Unterschiede. Diese scheinen unabhängig von der Position zu sein. Sie tauchen sowohl in der Mitte als auch am Rand auf. Es fällt auf, dass das Octree-Modell im Allgemeinen ein bisschen tiefer als das Grid-Modell liegt. Bei den anderen Vergleichen schien die Tendenz eher umgekehrt zu sein. Eine genaue Systematik scheint aber wiederum nicht vorhanden zu sein.

Trotz dieser grösseren Differenzen ist auch dies ein gutes Resultat. Die Unterschiede lassen sich durch die verschiedenen Modellierungen der beiden Algorithmen erklären. Einerseits umfasst die Modellierung des Octree wieder schlicht mehr und andererseits liegt der relevante Teil der Differenzen unter einer Voxelgrösse, was auch an der Art der Modellierung und nicht einem systematischen Fehler auszumachen ist.

3.4. Octree vs. Groundtruth

Im vorherigen Schritt wurde festgestellt wie der Octree-Algorithmus im direkten Vergleich zu einem Algorithmus, der auf eine Grid Datenstruktur setzt, abschneidet. Die Resultate waren zwar durchwegs positiv, jedoch sagen sie ohne den Vergleich mit der Groundtruth noch nicht genug aus. Wie schon erwähnt haben wir hier den Vorteil eine sehr genaue Groundtruth zu haben.

Die zuvor vereinigten Klassen des Octree-Modells werden nun also als Gesamtmodell mit der Groundtruth verglichen. Ein Vergleich zwischen Grid und Groundtruth wird nicht durchgeführt. Die absolute Genauigkeit des Grid-Modelles ist für diese Arbeit nicht von sonderlichem Interesse. Vielmehr diente das Modell zur Verifizierung des Octree-Modelles durch die Analyse der relativen Genauigkeit zwischen den beiden Modellen. Da diese ja sehr positiv ausfällt, kann deshalb auf weitere Vergleiche mit dem Grid-Modell verzichtet werden. Im Gegensatz zu den Klassenvergleichen wird es hier wohl auch interessant sein, mehrere durchgeführte Vergleiche mit den verschiedenen Grenzwerten zu analysieren.

Es sollte erwähnt werden, dass dies jeweils ohne die Klasse Vegetation geschieht. Das kommt daher, da die verwendeten Daten aus verschiedenen Stadien der Generierung der Groundtruth stammen. Während die tatsächlich verwendete Groundtruth aus einem früheren Stadium stammt, in welchem die Vegetation noch kein Bestandteil davon war, wurde für die Generierung des Octree-Modells eine Groundtruth mit Vegetation benutzt.

Die Groundtruth wird hier immer als Referenzmodell und das Octree-Modell als Test verwendet.

Wegen der guten Ergebnisse im vorhergehenden Schritt, macht es wohl Sinn mit dem Vergleich mit dem niedrigsten Grenzwert zu beginnen.

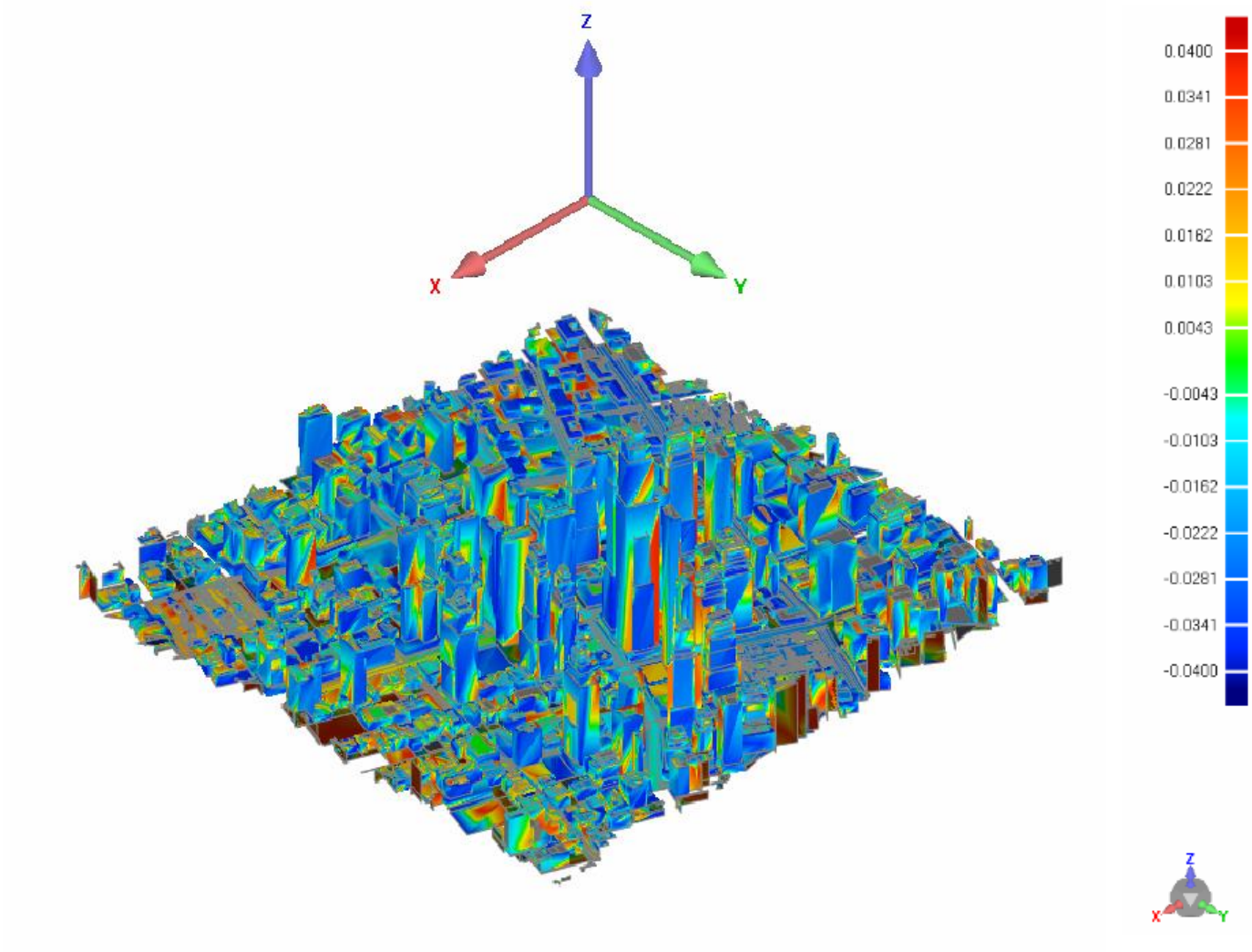


Abbildung 12: 3D-Compare Groundtruth vs. Octree mit grafischem Grenzwert auf eine Voxelgrösse festgelegt

Wie man sieht, treten hier wesentlich grössere Differenzen auf. Rund 55 Prozent aller Daten konnten mit diesem Grenzwert wegen zu grossen Abweichungen nicht verglichen werden und auch die Standardabweichung der Abweichungen liegt mit 0.0207 fast bei einer halben Voxelgrösse. Die Tendenz, dass das Octree Modell doch an den meisten Stellen unter der Groundtruth liegt, fällt dennoch auf.

Aufgrund dieser Zahlen ist es für genauere Analysen von Vorteil für den Vergleich den Grenzwert mit fünf Voxelgrössen zu verwenden. Hier sind es nur 23 Prozent, die nicht verglichen werden konnten.

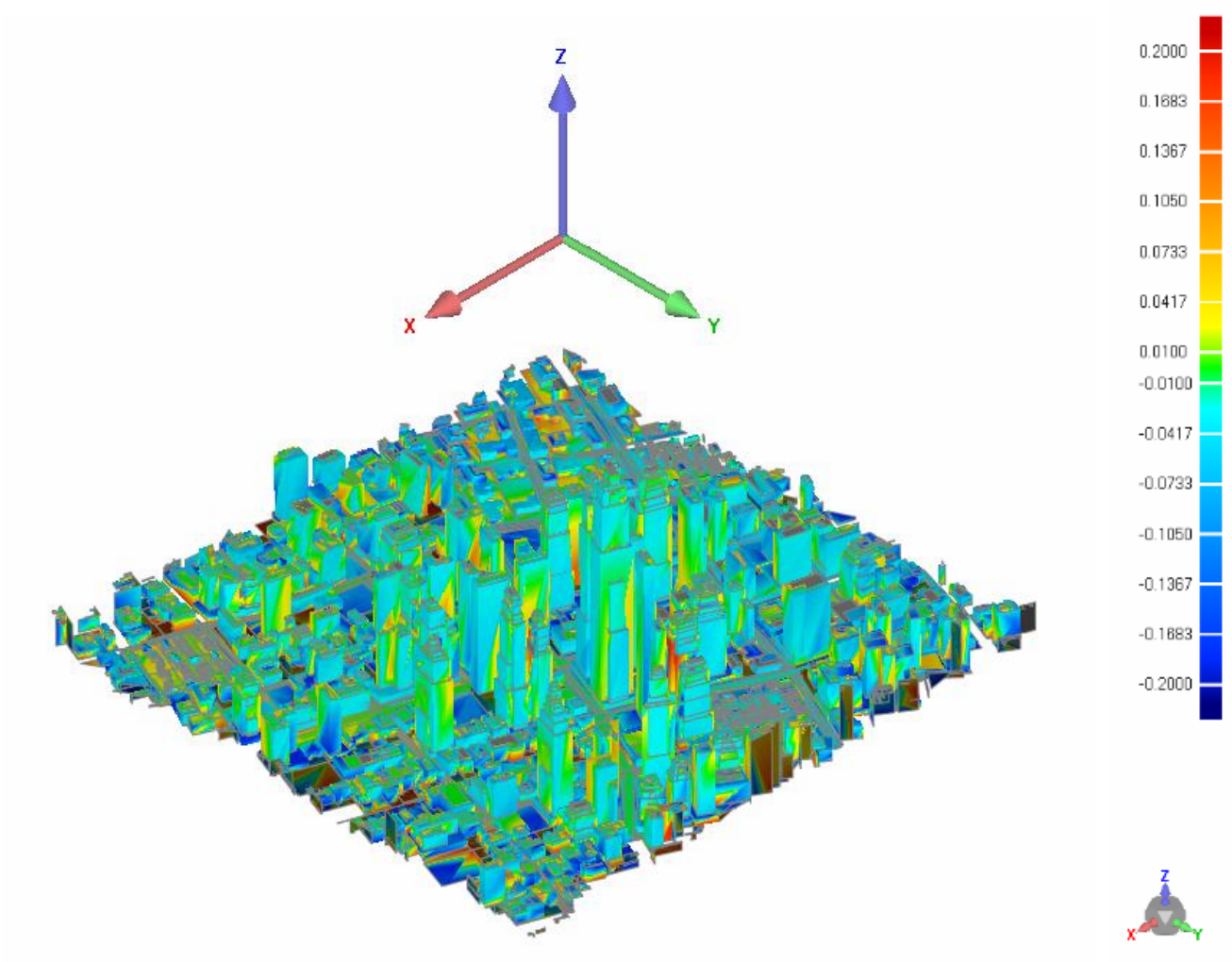


Abbildung 13: 3D-Compare Groundtruth vs. Octree mit grafischem Grenzwert auf fünf Voxelgrössen festgelegt

Es scheint als ob sich vorherigen Beobachtungen bewahrheitet haben. Das Octree-Modell liegt tatsächlich an den allermeisten Stellen unter der Groundtruth. Und zwar zum grössten Teil zwischen einem bis zwei Voxel. Auch fällt wieder auf, dass sich die Abweichungen je näher man dem Boden kommt vergrössern. In diesem Gebiet steigen die Abweichungen dann auf fünf Voxel und mehr an.

Eine interessante Erkenntnis findet man, wenn man sich Querschnitte durch den Vergleich von zwei gegenüberliegenden Seiten anschaut.

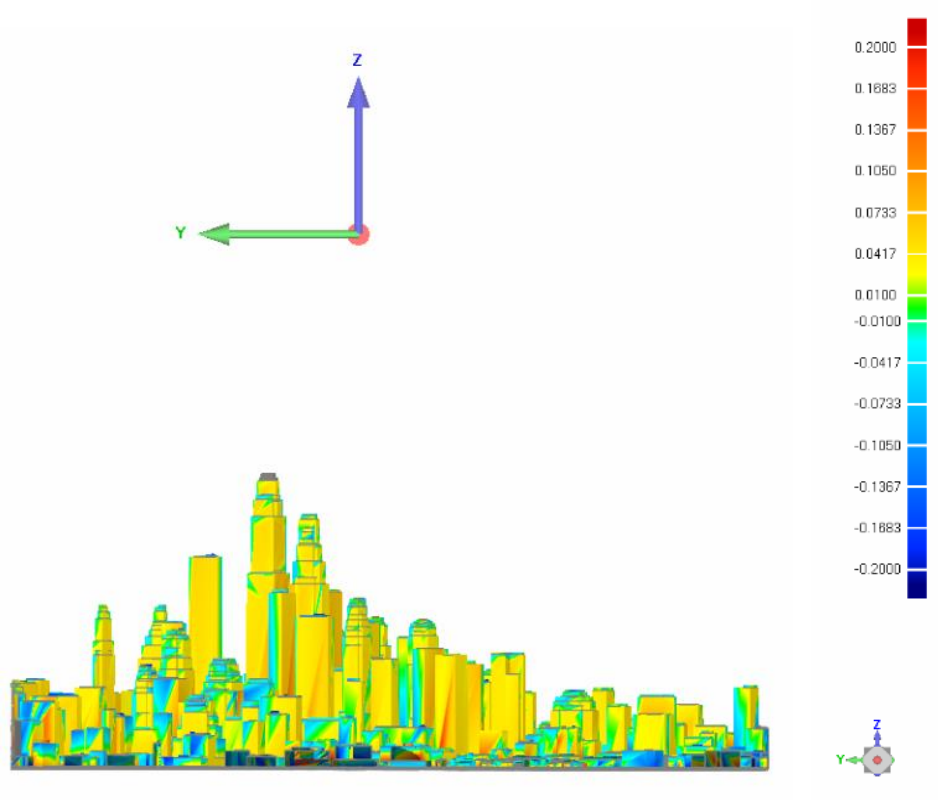


Abbildung 14: Querschnitt 1 durch den 3D-Compare Vergleich Groundtruth vs. Octree mit Grenzwert auf fünf Voxelgrössen festgelegt

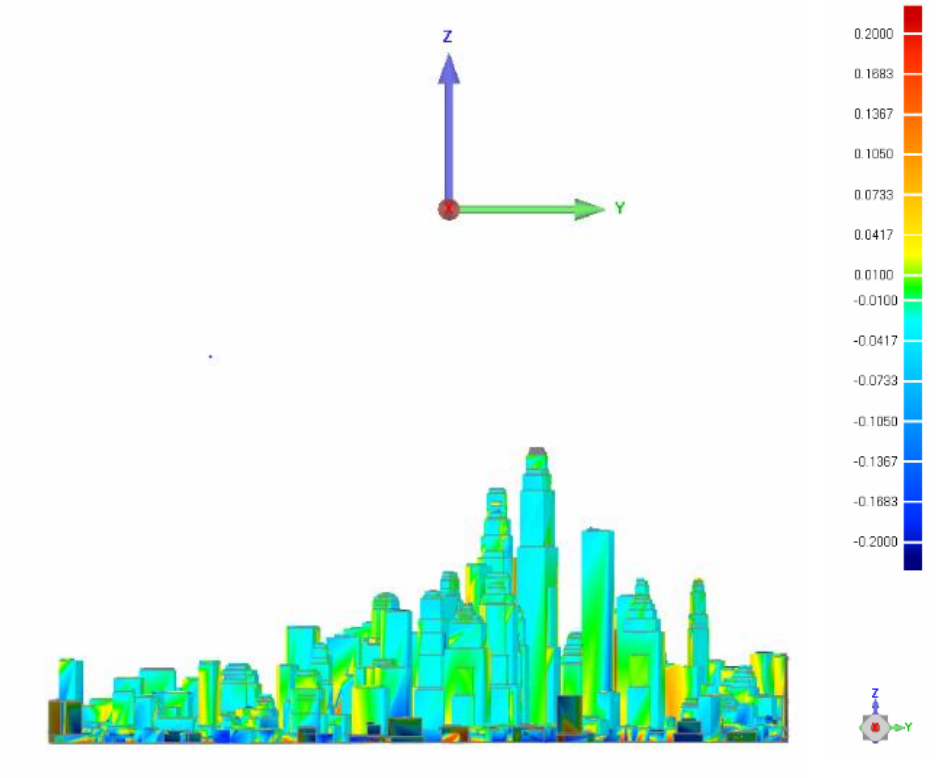


Abbildung 15: Querschnitt 2 durch den 3D-Compare Vergleich Groundtruth vs. Octree mit Grenzwert auf fünf Voxelgrössen festgelegt

Die Richtung der Abweichungen hängen also von der Seite ab, von der man das Modell ansieht. Schaut man in Richtung der x-Achse liegt das Octree-Modell vor der Groundtruth und schaut man gegen die x-Achse verhält es sich genau umgekehrt. Daraus kann man schliessen, dass das ganz Octree-Modell horizontal etwa um eine Voxelgrösse gegen die Richtung der x-Achse verschoben ist. In der vertikalen Richtung findet man hingegen eine solch klar definierbare systematische Abweichung nicht. Wie schon erwähnt, scheint das Ausmass der Abweichung mit der Nähe zum Boden in Verbindung zu stehen.

3.5. Schlussfolgerungen und Diskussion

Generell kann man sagen, dass die Ergebnisse aus den Vergleichen überwiegend positiv sind. Die relative Genauigkeit des Octree-Modells gegenüber dem Grid-Modell ist sehr gut. Bei den meisten Klassen belaufen sich diese Unterschiede im deutlichen Subvoxel-Bereich, was auf die unterschiedliche Modellierung zurückzuführen ist. Gerade bei Klassen, welche fast nur aus horizontalen Ebenen bestehen, gibt es wirklich so gut wie keine Differenzen. Lediglich in der Vertikale treten kleinere Unterschiede auf. Auffallend ist, dass diese Unterschiede meistens in Bodennähe zunehmen. Hinweise auf systematische Abweichungen zwischen den beiden Modellen sind aber keine aufzufinden. Einzig bei der Klasse Vegetation lassen sich vermehrt Differenzen, die über eine Voxelgrösse hinausgehen, finden. Dies sollte aber nicht überraschen, da diese Klasse die wohl mit Abstand am schwierigsten zu modellieren ist und den entsprechenden Algorithmen dazu am meisten Spielraum lässt. Eine deckungsgleiche Modellierung darf hier also nicht erwartet werden.

Zudem fällt auch auf, dass das Octree-Modell im Allgemeinen umfangreicher modelliert ist. Dies tritt oft am Rand der Modelle auf und trifft für das ganze Gebiet besonders bei der Klasse Boden zu. Dies könnte wohl auch mit Tatsache korrelieren, dass jeweilige Unterschiede zwischen den Modellen in Bodennähe zunehmen.

Die Vorteile hinsichtlich Speicherbedarf und Prozessierungsgeschwindigkeit im Vergleich zum Grid-Modell werden also nicht durch grosse Genauigkeitsverluste in Kauf genommen. Nicht nur ist der relative Genauigkeitsunterschied sehr klein, sondern modelliert der Octree-Algorithmus das Gebiet auch umfangreicher.

Wenn man den absoluten Vergleich mit der Groundtruth durchführt, erkennt man schon grössere Abweichungen. Es scheint in der horizontalen Richtung eine systematische Verschiebung von etwa einer Voxelgrösse vorzuliegen. In der Vertikale hingegen tauchen keine klar definierbaren systematischen Fehler auf. Auffallend ist lediglich, dass das Octree-Modell an den meisten Stellen unter der Groundtruth liegt. Während dies bei den Dächern der höchsten Häuser im Voxelbereich liegt, nimmt dies auf dem Boden doch deutlich grössere Ausmasse an. Abweichungen von fünf Voxel und sogar mehr treten durchaus oft auf.

Beide Vergleiche – absolut und relativ – haben die Nähe zum Boden als Indikator für grössere Abweichungen gemeinsam. Der Octree-Algorithmus scheint also mit Bodennähe Probleme zu haben.

Eventuell sind kleinere Abweichungen auch durch die Transformation der Groundtruth in das Koordinatensystem der Modelle zu erklären. Wenn man sich die Standardabweichungen der Transformationsparameter jedoch ansieht wird klar, dass dies nur einen sehr kleinen Einfluss haben kann und zum grössten Teil zu vernachlässigen ist.

4. Oberflächenreduzierung

Detaillierte 3D-Modell benötigen in der Regel viel Speicherplatz, was auch eine langsame Prozessierung mit sich bringt. Je nach Anwendungsgebiet ist aber ein solch hoher Detailgrad nicht zwangsläufig nötig. Oft kann es auch vorkommen, dass gewisse Teile eines Modelles komplexe Strukturen enthalten, wo hingegen andere Teile aus simplen Geometrien bestehen. Bei der Generierung des Modells wird dies aber in der Regel nicht berücksichtigt und das Modell wird auf der ganzen Oberfläche mit der gleichen Auflösung erzeugt. Wenn man dazu das Stadtmodell aus dem vorherigen Kapitel betrachtet, fällt auf, dass dies hier besonders zutrifft und im direkten Zusammenhang mit der semantischen Information steht. Für die Klassen Boden oder Dach etwa ist es nicht erforderlich die gleiche Auflösung wie die Klasse Gebäude zu haben. Dementsprechend würde es also Sinn machen die Auflösung dieser Klassen zu reduzieren, während andere Klassen eine hohe Auflösung beibehalten.

Um dies zu erreichen werden im Rahmen dieser Arbeit Teile des Algorithmus „Surface Simplification Using Quadric Error Metrics“ implementiert. Dieser Algorithmus kann auf alle Modelle, die durch eine Oberflächentriangulierung definiert sind, angewendet werden.

Die folgenden theoretischen Grundlagen stammen aus der oben erwähnten Arbeit. [Surface Simplification Using Quadric Error Metrics, Garland et al. 1997]

4.1. Theoretische Grundlagen

Der Algorithmus von [Garland et al.] hat sich als Grundstein und Standard für weitere Ansätze und Weiterentwicklungen im Bereich der Oberflächenvereinfachung in der 3D-Modellierung bewährt. Der Algorithmus hat Vorteile in Sachen Effizienz, Qualität und der Möglichkeit ihn generell auf alle Modelle anwenden zu können.

Die Grundidee besteht darin, durch die Zusammenlegung von einem Punktpaar der Triangulierung zu einem einzelnen Punkt und die folgende Neutriangulierung, die Gesamtzahl aller Dreiecke der Triangulierung des Modells um einen gewünschten Betrag zu reduzieren. Das Entscheidende ist nun, die Punkte zur Zusammenlegung so zu wählen, dass die Auswirkungen auf die Struktur des ursprünglichen Modells möglichst klein ausfallen. Durch die namensgebende Quadric Error Metric (QEM) kann numerisch festgestellt werden, welche Paare sich besonders dazu eignen. Dieser Prozess wird iterativ durchgeführt.

Zu Beginn jeder Iteration werden alle zur Zusammenlegung geeigneten Punktpaare eruiert. Geeignet sind Paare, die eine von den zwei folgenden Bedingungen erfüllen.

1.) Beide Punkte sind die Endpunkte der gleichen Dreieckskante im Modell.
2.) Der Abstand zwischen den beiden Punkten ist kleiner als ein bestimmter Grenzwert.

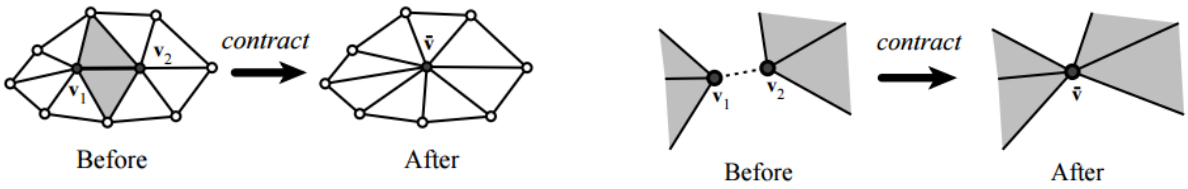


Abbildung 16: Visualisierung der beiden Bedingungen [Garland et al. 1997]

Geht man von einem Modell aus, welches durch die Modellierung Löcher, die man schliessen möchte, enthält, empfehlt es sich Bedingung 2 zu berücksichtigen. Hat man hingegen ein Modell, von dem man ausgehen kann, dass es keine oder nur sehr wenige Löcher enthält, kann man Bedingung 2 auch ignorieren.

Man hat nun also eine Übersicht über alle geeigneten Punktpaare zur Zusammenlegung. Der nächste Schritt ist festzulegen welche dieser Paare man zusammenlegen kann unter der Bedingung, die Charakteristik des ursprünglichen Modells so wenig wie möglich zu verändern. Zum Beispiel würden sich Punkte auf einer Fläche sehr gut, Punkte, welche sich entlang einer Ecke eines Objekts befinden, umso weniger eignen.

Dazu wird für jeden Punkt der Triangulierung die sogenannte Quadric Error Metric berechnet. Die Quadric Error Metric ist eine 4x4 Matrix und beschreibt den Fehler am Modell, der zu erwarten ist, wenn dieser bestimmte Punkt zur Zusammenlegung genutzt wird. Zur Berechnung dieser Matrix benötigt man zuerst für jedes Dreieck des Modells eine Fundamental Error Quadric K. Diese Matrix sieht wie folgt aus:

$$K = \begin{matrix} a^2 & ab & ac & ad \\ ab & b^2 & bc & bd \\ ac & bc & c^2 & cd \\ ad & bd & cd & d^2 \end{matrix}$$

Die Koeffizienten a,b,c und d erhält man aus der normierten Normalengleichung, welche die Ebene, in der das jeweilige Dreieck liegt, beschreibt:

$$a * x + b * y + c * z + d = 0$$

Unter der Bedingung der Normierung:

$$a^2 + b^2 + c^2 = 1$$

Die QEM-Matrix eines Punktes ist nun nichts anderes als die Summe der K-Matrizen der Dreiecke, die sich in diesem Punkt treffen.

$$QEM = \sum Ki$$

Diese Matrix ist ein Indikator für den Abstand des betrachteten Punktes zu den umliegenden Dreiecken. Durch diese Quadric Error Metrics lässt sich nun für jede Vereinigung eines Punktpaares ein bestimmter Kostenwert berechnen. Dieser Wert ist dann auch das entscheidende Indiz dafür, ob sich dieses Paar zur Zusammenlegung empfiehlt. Dies geschieht durch das folgende Vektor-Matrixprodukt:

$$Kostenwert = v' * (Q1 + Q2) * v$$

Wobei v den Koordinaten des theoretischen Neupunktes und Q1 und Q2 den QEM des Punktepaares entspricht. Dieser Neupunkt wird oft als Mittelpunkt zwischen dem Punktpaar gewählt. Man könnte ihn prinzipiell beliebig wählen, jedoch sollte er auf besagter Kante liegen, da sonst die Kosten sehr schnell in die Höhe schiessen würden und der zu testende Punkt schnell verworfen würde. Würde man den obigen Kostenwert für einen Eckpunkt mit nur seiner QEM berechnen, ergäbe sich als Resultat folgenderweise 0. Dies macht ja insofern Sinn, als dass der Punkt zu den umliegenden Dreiecken keinen Abstand hat.

Man berechnet nun also für jedes eruierte Punktepaar diesen Kostenwert. Wie erwähnt können dies auch mehrere Werte für mehrere potenzielle Neupunkte entlang der Kante sein. Diese Werte werden geordnet und die Zusammenlegung wird für einen im Vorhinein bestimmten Anteil mit den niedrigsten Kosten durchgeführt. Das Modell wird neu trianguliert und die nächste Iteration beginnt. Dies wird solange durchgeführt bis man das originale Modell um eine gewünschte Anzahl an Dreiecke reduziert hat.

Für mehr Informationen und genauere mathematische Hintergründe wird auf die Arbeit von [Garland et al. 1997] verwiesen.

4.2. Implementation

Für die Implementation des Algorithmus wurde das Programm Matlab verwendet. Da sämtliche 3D-Modelle sich per Text-Editor als Koordinatenliste der Punkte mit dazugehörender Dreiecksindizierung öffnen lassen, fällt so auch die Importierung leicht. Bei der Importierung ist aufgefallen, dass die Klasse Gebäude vier degenerierte Dreiecke enthielt. Also eine Indizierung, die nur aus zwei statt der benötigten drei Punkten besteht. Dies hängt wohl mit der Modellierung des Octree-Algorithmus zusammen. Da es aber nur bei dieser Klasse auftrat und zudem auch nur in insignifikanter Anzahl kann dies getrost ignoriert werden.

Man hat nun also eine Liste der Koordinaten aller Punkte im Modell, sowie eine Liste der Indizes aller Dreiecke, welche für die Triangulierung gebraucht wird. Als nächster Schritt wird nun eine Funktion gesucht, die eine Liste aller geeigneten Punktpaare zur Vereinigung generiert. Da man bei den Octree-Stadtmodellen eigentlich davon ausgehen kann, dass das Modell keine oder nur sehr wenige Löcher enthält, wird auch nur Bedingung 1 zur Vereinigung implementiert. Man braucht also eine Liste aller Kanten im Modell. Dazu wurde die Funktion „compute_edges“ [Wasmeier 2004], welche man in der Mathworks File Exchange Bibliothek findet, verwendet. Diese braucht als Input die schon vorhandenen Daten und liefert genau die gesuchte Liste aller Kanten im Modell. Zur Vereinfachung wurde dann entschieden, den jeweiligen Neupunkt genau in die Mitte der Kante zu setzen.

Als nächstes wird eine Funktion implementiert, die aus den drei Eckpunkten eines Dreiecks die gesuchte K-Matrix berechnet. Dazu müssen zwei Richtungsvektoren gebildet werden, die die Ebene des jeweiligen Dreiecks aufspannen. Mittels des Kreuzproduktes wird aus diesen Richtungsvektoren der Normalenrichtungsvektor des Dreiecks berechnet. Dieser beinhaltet schon drei der gesuchten Koeffizienten und eingesetzt in die Normalengleichung hat man dann alle Komponenten der K-Matrix. Diese müssen dann noch normalisiert werden.

Summiert man alle relevanten Matrizen dann für jeden Punkt auf, ergibt sich die gesuchte Quadric Error Metric. Wie im theoretischen Teil beschrieben, kann nun für jeden Mittelpunkt aller Kanten berechnet werden, wie sich dieser Punkt zur Vereinigung der Eckpunkte dieser Kante eignet. Damit ist man auch schon beim Ende einer Iteration angelangt.

Da eine Iteration das Prinzip des Algorithmus schon gut veranschaulicht und die Durchführung mehrerer Iterationen den zeitlichen Rahmen dieser Arbeit überschreitet, wurde entschieden nur eine Iteration zu implementieren.

Die erhaltenen Kostenwerte für jeden potenziellen Neupunkt werden jetzt als Endresultat der Implementation visualisiert. Dazu werden diese Werte erst mit der folgenden Gleichung in das Intervall [0..1] transformiert:

$$Kostenwert = \left(\frac{1}{max - \min}\right) * (Wertalt - \min)$$

Wobei max und min dem grössten bzw. kleinsten aller nicht normalisierten Kostenwerte entsprechen. Diese Normalisierung ist nötig um die Kostenwerte einfach auf eine RGB-Farbskala transformieren zu können. Die am schlechtesten geeigneten Punkte sollen rot dargestellt werden und umgekehrt die am besten geeigneten blau. In der RGB-Farbskala sieht die Darstellung für rot und blau jeweils so aus:

$$Rot = 1 \quad 0 \quad 0 \qquad\qquad Blau = 0 \quad 0 \quad 1$$

Deshalb wird die folgende nicht lineare Mapping-Funktion verwendet:

$$Farbe = Kostenwert^{Parameter} \quad 0 \quad 1 - (Kostenwert^{Parameter})$$

Der Potenzparameter wird zur besseren Visualisierung verwendet und meistens im Bereich von 0.2 bis 0.5 angesetzt. Ohne ihn wären die grafischen Resultate unklar, da die Kostenwerte in der ersten Iteration bis auf wenige Ausnahmen alle sehr klein sind und ungleich verteilt sind. Der grösste Teil des Modells würde bei einer linearen Mapping-Funktion so einfach in blau erscheinen und Unterschiede wären kaum auszumachen. Dies ist beispielsweise auch an dem Histogramm der Kostenwerte der Klasse Gebäude zu erkennen. Es ist anzunehmen, dass sich dies nach mehreren Iterationen ändern würde und die Kostenwerte regelmässiger verteilt wären.

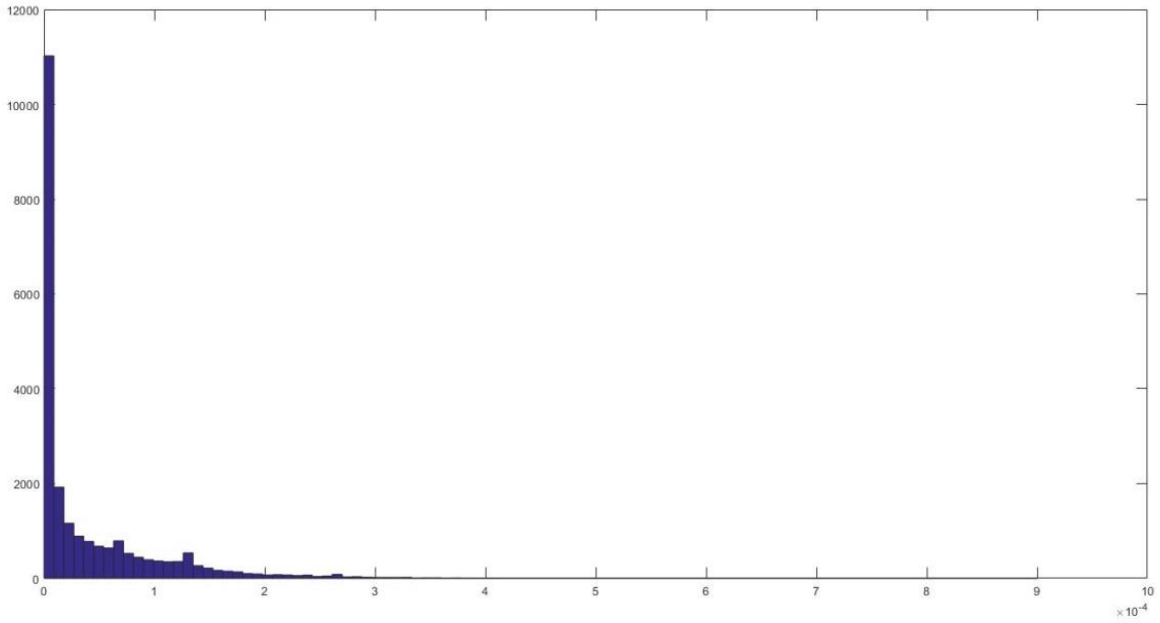


Abbildung 17: Histogramm der Kostenwerte der Klasse Gebäude ohne einen Potenzparameter [Matlab 2016]

Die berechneten potenziellen Neupunkte und ihre dazugehörenden Farben werden nun als Punktwolke im ply-Format exportiert und zur besseren Darstellung in das Programm Meshlab geladen. Damit ist die Visualisierung der ersten Iteration des Algorithmus vollständig und man kann entsprechend erkennen an welchen Stellen sich das Modell zur Vereinfachung eignet.

4.3. Resultate

Da die Importierung der benötigten Modelle in Matlab relativ zeit–und rechenaufwändig ist, wurde zum Testen des Algorithmus nur ein kleiner und relativ niedrig aufgelöster Ausschnitt eines ganzen Stadtmodells verwendet. Der Algorithmus wurde auf jede einzelne Klasse – mit der Ausnahme von Clutter - angewandt und liefert gute Ergebnisse. Bei jeder Klasse ist klar erkennbar, dass gerade Flächen sich gut zur Glättung eignen und Ecken sowie Kanten eher nicht.

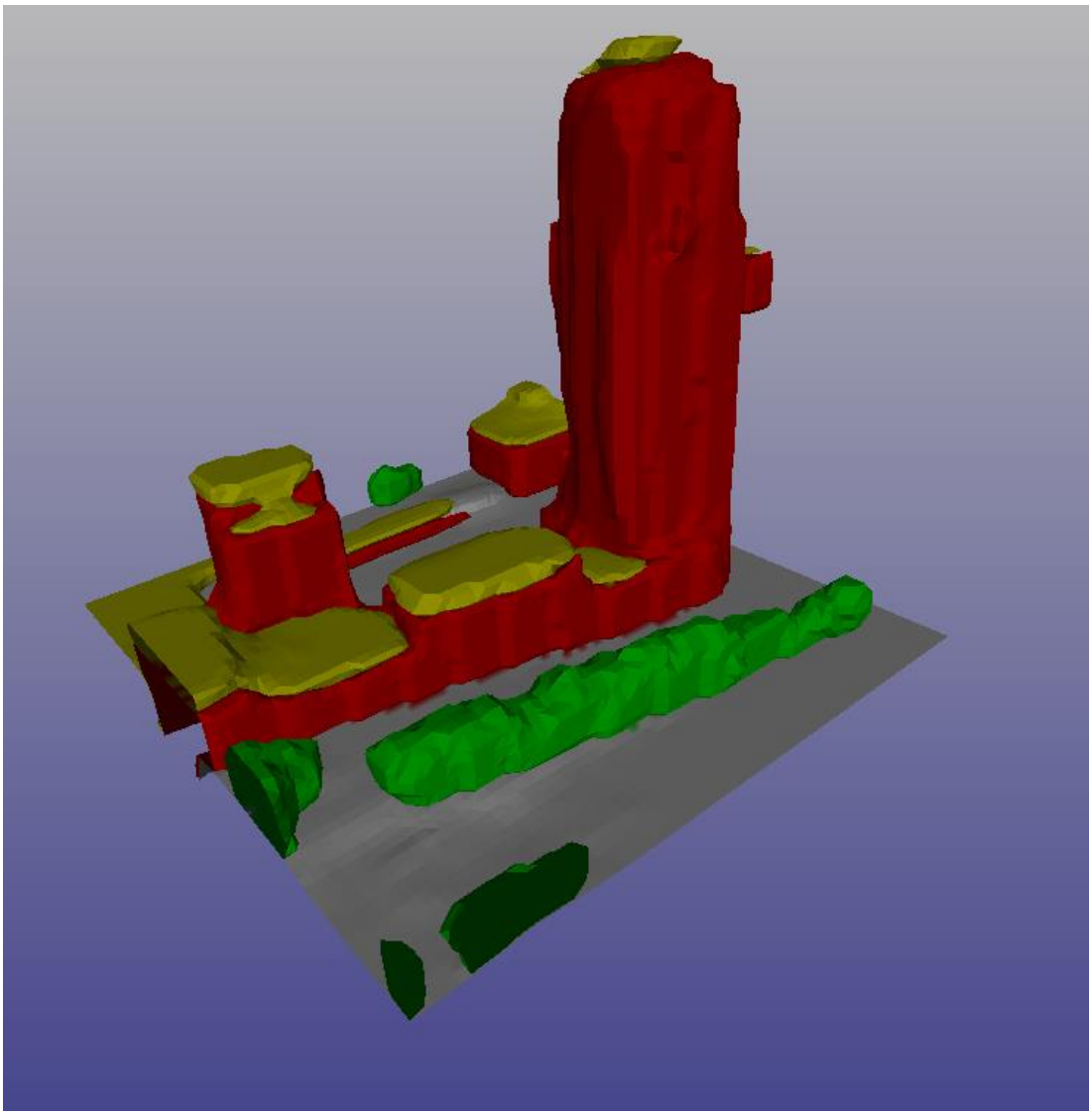


Abbildung 18: Verwendeter Teilbereich

Für die Klassen Boden und Dach ist also zu erwarten, dass sich ein grosser Teil des Modells zur Vereinfachung eignen würde. Tatsächlich erhält man auch diese Resultate wie man erkennen kann.

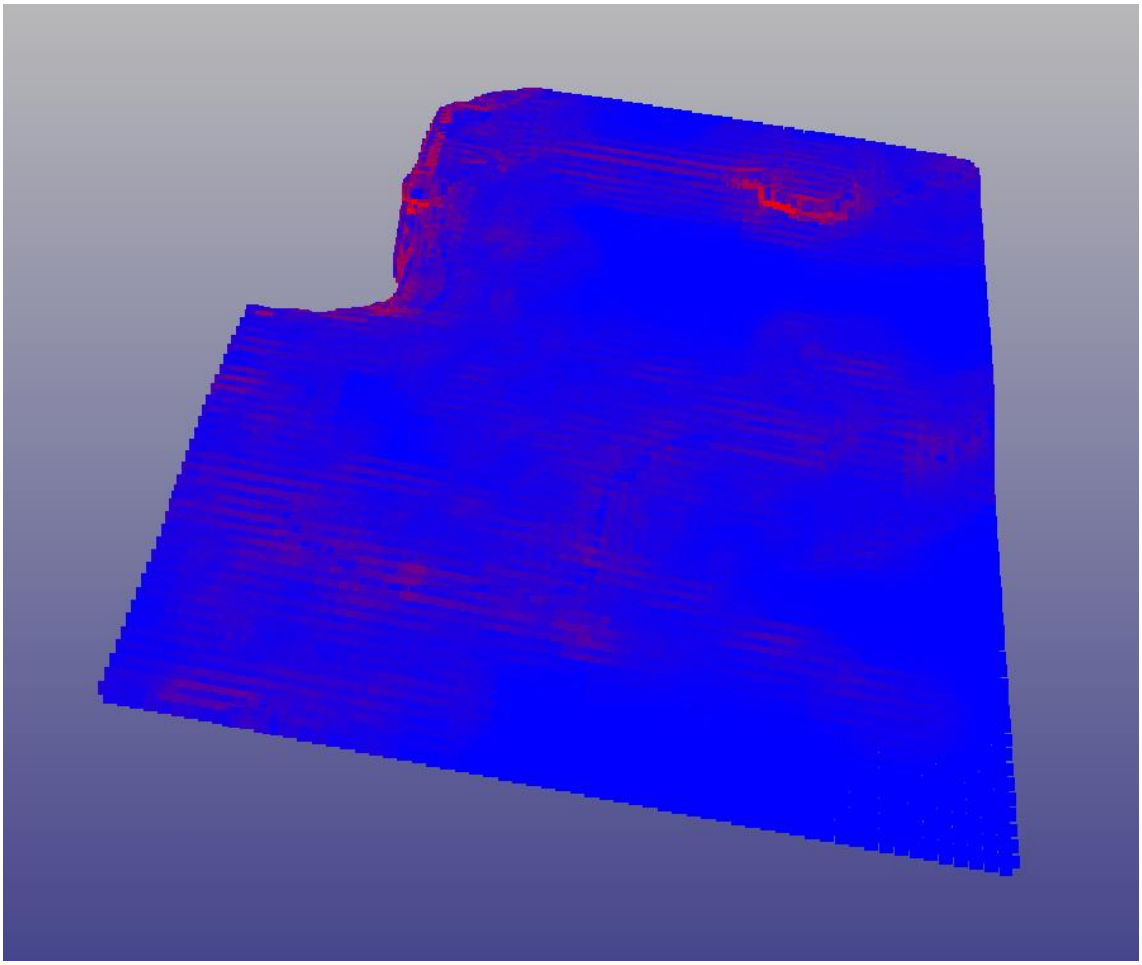


Abbildung 19: Resultat des Algorithmus auf die Klasse Boden angewandt.

Die einzigen ungeeigneten Stellen findet man bei der Klasse Boden eigentlich nur bei leichten Erhebungen. Eine Ausnahme ist die Stelle oben links, bei der die Klasse Boden in eine andere Klasse übergeht.

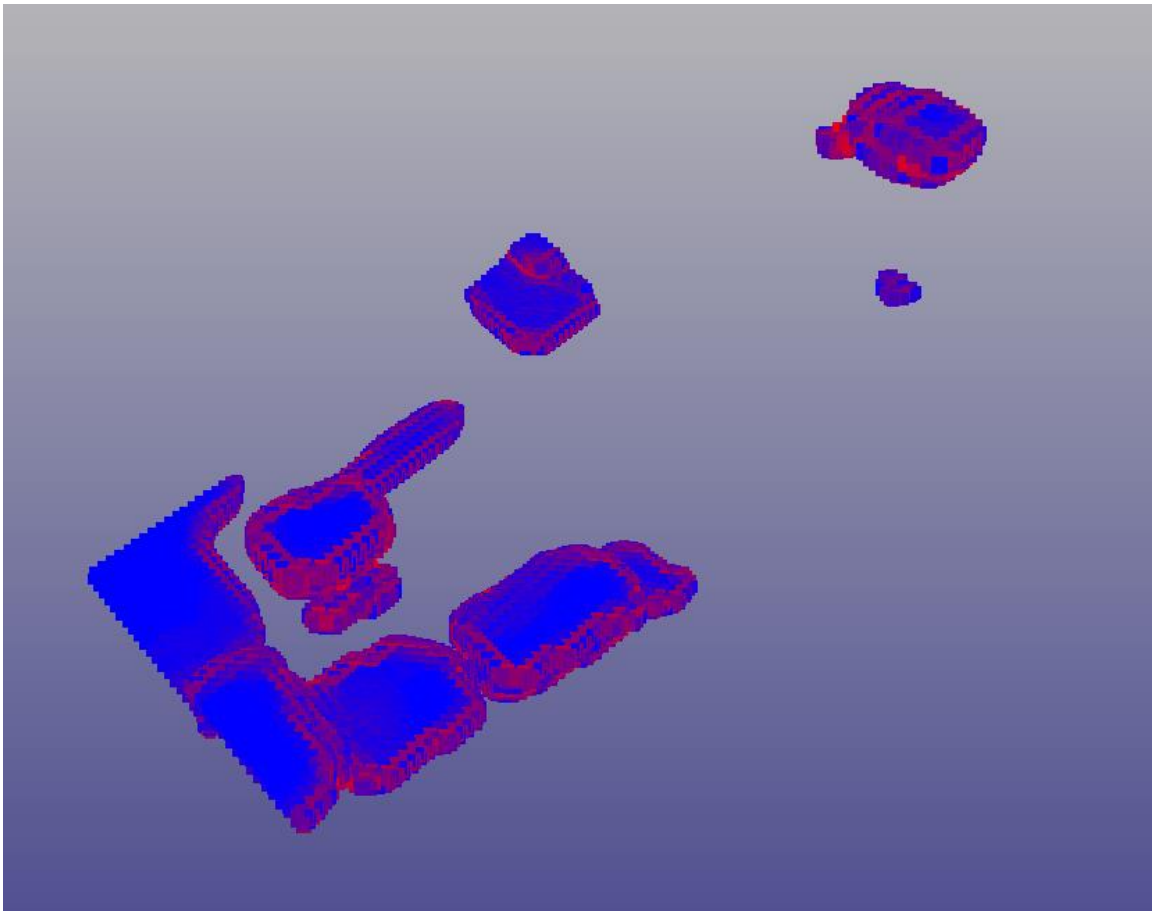


Abbildung 20: Resultat des Algorithmus auf die Klasse Dach angewandt.

Auch bei der Klasse Dach sieht es ähnlich aus. Die ebenen Flächen dieser Klasse sind gut geeignet, während die Abrundungen des Dachs sich nicht eignen. Das sind genau jene Stellen, in denen diese Klasse in die Klasse Gebäude übergeht. Interessant ist es nun zu sehen, ob sich diese Tendenz genauso bei der Klasse Gebäude fortsetzt.

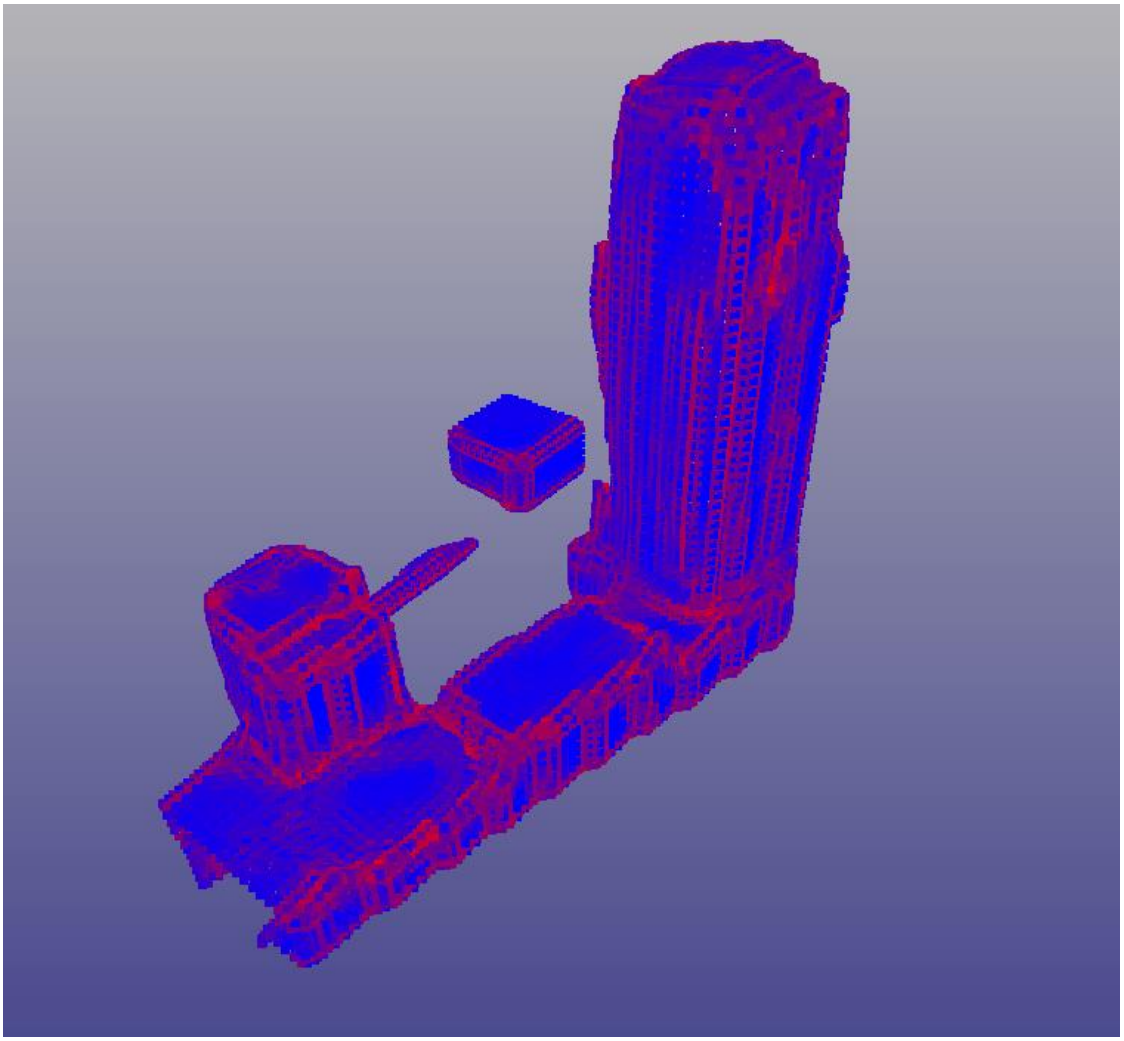


Abbildung 21: Resultat des Algorithmus auf die Klasse Gebäude angewandt.

In der Tat ist dies auch so. Gut zu erkennen ist die Untauglichkeit der Kanten zur Glättung. Dass die Kanten auch blaue Stellen enthalten, ist zu erklären wenn man sich das Originalmodell ansieht. Da dieses nicht den höchsten Detailgrad enthält, sind schon zum Teil geglättete Kanten vorhanden und damit auch schon ebene Flächen. Ein Test mit einem höher aufgelösten Modell ist leider aus leistungstechnischen Gründen nicht möglich.

Ergänzt man alle Klassen zu einem Gesamtmodell ergibt sich ein schönes Bild des Ausschnitts und man kann die jeweiligen Klassen auch schon gut an der Tauglichkeit zur Vereinfachung erkennen.

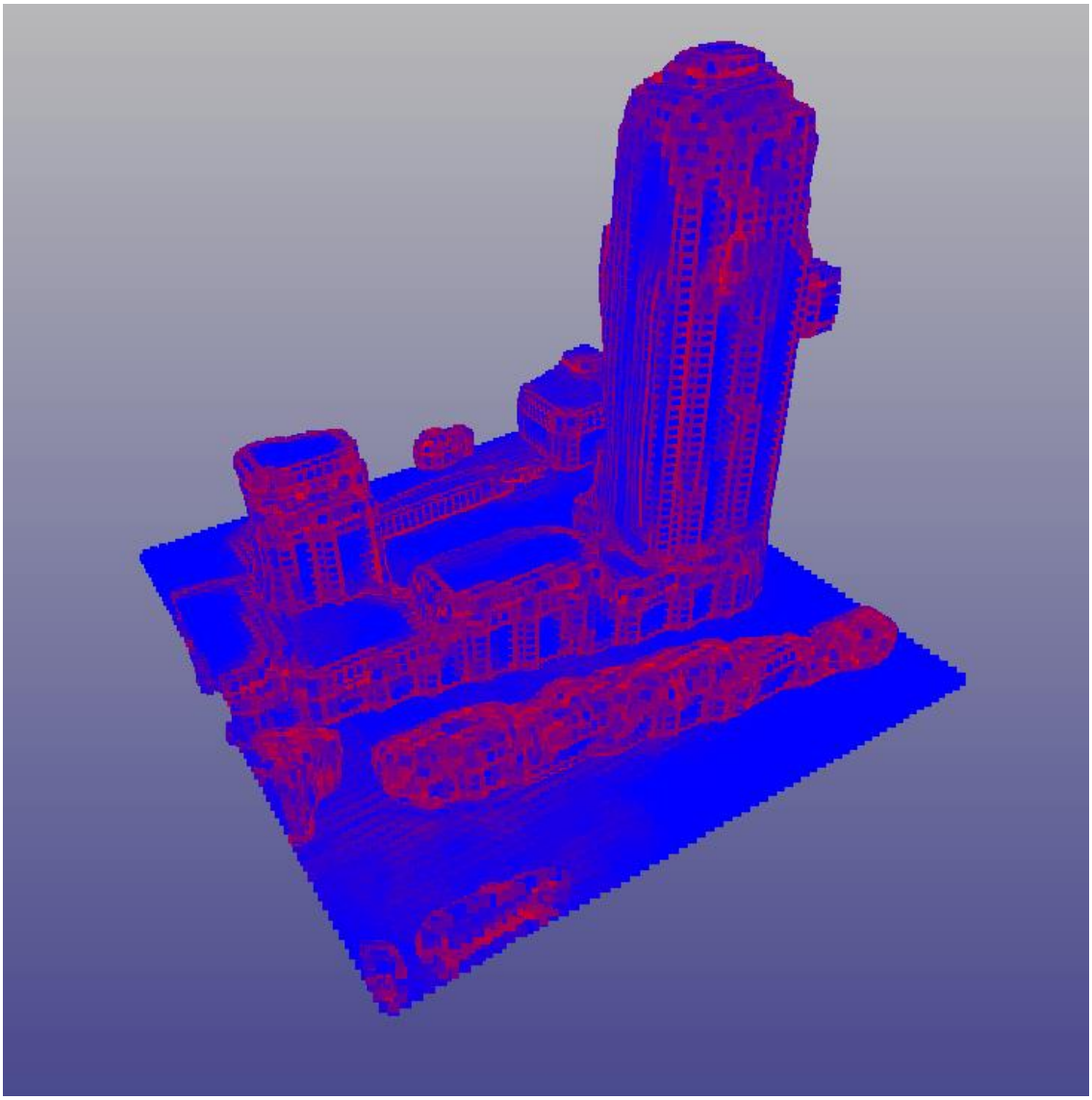


Abbildung 22: Zum Gesamtmodell zusammengefügte Resultate

Auf die Diskussion der Anwendung des Algorithmus auf die Klasse Vegetation wird hier verzichtet. Wegen der Schwierigkeit diese Klasse zu modellieren, hat sie wohl im Vergleich zu den anderen den grössten Grad an Abstraktion. Da das Modell der Vegetation daher auch nicht der Realität entspricht, ist die Vereinfachung ihrer relativ komplexen Geometrie, bzw. das Resultat des Algorithmus auf ihr Modell angewandt, eher uninteressant.

Um die Tauglichkeit des Algorithmus für kompliziertere Geometrien zu prüfen, wird er auch mit weiteren Modellen getestet. Als Beispiel wird hier der Buddha aus der Stanford Bibliothek von 3D-Modellen gezeigt. Dieses Modell sieht im Original wie folgt aus:
[http://graphics.stanford.edu/data/3Dscanrep/]

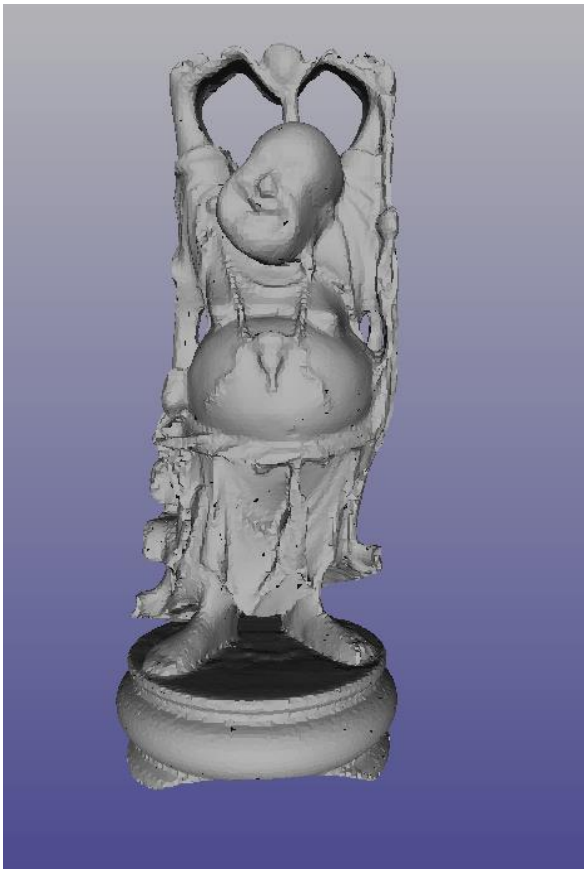


Abbildung 24: Originalmodell des Buddha [http://graphics.stanford.edu/data/3Dscanrep/]

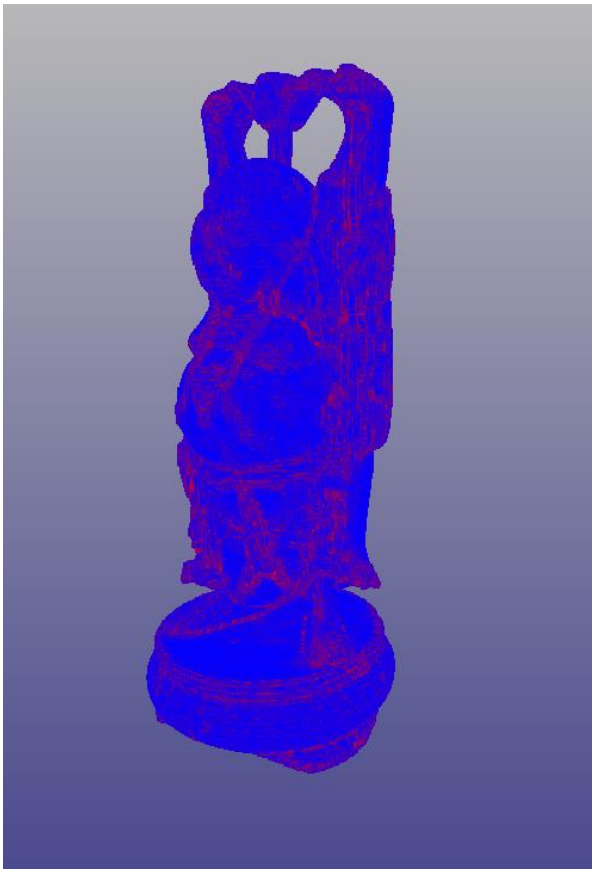


Abbildung 23: Resultat des Algorithmus auf das Buddha-Modell angewandt.

Auch hier sehen die Resultate wie erwartet aus. Jedoch scheint es so, dass die Implementation des Algorithmus an ihre Grenzen angelangt ist. Zwar sieht man die gewünschten geeigneten Stellen zur Glättung, jedoch nicht mehr so klar wie bei den vorherigen Resultaten. Es scheint also für deutlich komplexere Geometrien nicht mehr möglich sein nur die erste Iteration durchzuführen um eindeutige Resultate zu erhalten.

Für die vergleichsweise simplen Geometrien des Stadtmodells scheint die Implementation aber durchaus auszureichen. Zwar geht es hier vorrangig um die Visualisierung, doch könnte es durchaus sein, dass auch zur Vereinfachung der Oberfläche eine oder nur wenige Iterationen ausreichen könnten. Natürlich hängt das auch noch davon ab wie hoch der Anteil der Punktpaare, die bei jeder Iteration vereinigt werden, festgelegt wird. Dies gilt natürlich wie erwähnt nur für eher einfache Geometrien. Schlussendlich bleibt festzustellen, dass eine Implementierung des QEM-Algorithmus bzw. dessen Feintuning immer von der Charakteristik der zu bearbeitenden Modelle abhängt.

Auch eine Anwendung auf höher aufgelöste Modelle wäre interessant gewesen. Leider ist dafür im Rahmen dieser Arbeit nicht die benötigte Rechenleistung verfügbar. Beziehungsweise müsste dafür die Implementation der Importierung deutlich optimiert werden.

5. Schlusswort und Ausblick

Allgemein kann man sagen, dass die Resultate dieser Arbeit zufriedenstellend sind. Im ersten Teil wurde die erfolgreich die gute Genauigkeit des Octree-Modells nachgewiesen. Während man gegenüber der Groundtruth eine systematische, horizontale Verschiebung von ungefähr einem Voxel hat, fällt der relative Unterschied zu dem Grid-Modell klein aus und ist auf Unterschiede in der Modellierung zurückzuführen. Systematische Abweichungen lassen sich hier keine finden.

Im zweiten Teil wurde ein Algorithmus implementiert, der die Eignung von Modellen zur Oberflächenvereinfachung aufzeigen soll. Auch hier wurden klare Ergebnisse erzielt und die Erwartungen wurden bestätigt. Der Algorithmus zeigt klar auf, dass sich gerade Flächen dazu eignen, Ecken und Kanten hingegen nicht.

Diese Implementation könnte nun noch weiter ausgebaut und abgeändert werden. Der erste Schritt dazu wäre den Algorithmus so zu ändern, dass er alle nötigen Iterationen selbständig durchführt und das Modell jeweils wieder neu trianguliert. Weiter wäre es möglich die semantische Information der Teilmodelle verstärkt auszunutzen. Beispielsweise könnte man bei Klassenübergängen die Teilmodelle so reduzieren, dass diese sich nicht mehr gegenseitig überlappen, was einer weiteren Speicherplatzreduzierung gleichkommt. Oder man könnte die Quadric Error Metrics je nach Klasse auch unterschiedlich gewichten. Dies würde dazu führen, dass bestimmte Klassen noch stärker vereinfacht werden, während andere zum grossen Teil im Originalzustand blieben. Bestimmt lassen sich viele Ansätze finden diesen Basisalgorithmus noch weiter auszubauen.

Denn in Zukunft wird gerade die Abwägung zwischen Detailgrad und Speicherbedarf wegen immer grösser und genauer werdenden Modellen noch wichtiger und die Anwendungsmöglichkeiten sowie der Bedarf von semantischen Informationen in 3D-Modellen werden stark wachsen. Die Forschungsbemühungen in diesem Gebiet werden sicherlich weiter zunehmen und der verbreite Einsatz in der Praxis ist nur eine Frage der Zeit.

6. Danksagung

Ich möchte mich bei folgenden Personen für die erfolgreiche Durchführung meiner Arbeit bedanken.

- Maros Blaha für die gute Betreuung meiner Arbeit während des ganzen Semesters und die vielen hilfreichen Tipps.
- Prof. Dr. Konrad Schindler für das Ausschreiben und Ermöglichen dieser Arbeit.
- Emanuel Meier für das Gegenlesen und Korrigieren.

7. Eigenständigkeitserklärung

ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Eigenständigkeitserklärung

Die unterzeichnete Eigenständigkeitserklärung ist Bestandteil jeder während des Studiums verfassten Semester-, Bachelor- und Master-Arbeit oder anderen Abschlussarbeit (auch der jeweils elektronischen Version).

Die Dozentinnen und Dozenten können auch für andere bei ihnen verfasste schriftliche Arbeiten eine Eigenständigkeitserklärung verlangen.

Ich bestätige, die vorliegende Arbeit selbständig und in eigenen Worten verfasst zu haben. Davon ausgenommen sind sprachliche und inhaltliche Korrekturvorschläge durch die Betreuer und Betreuerinnen der Arbeit.

Titel der Arbeit (in Druckschrift):

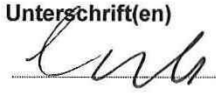
Analyse von semantischen 3D Stadt-Modellen

Verfasst von (in Druckschrift):
Bei Gruppenarbeiten sind die Namen aller Verfasserinnen und Verfasser erforderlich.

Name(n):	**Vorname(n):**
Miotti	Luca

Ich bestätige mit meiner Unterschrift:
- Ich habe keine im Merkblatt „Zitier-Knigge" beschriebene Form des Plagiats begangen.
- Ich habe alle Methoden, Daten und Arbeitsabläufe wahrheitsgetreu dokumentiert.
- Ich habe keine Daten manipuliert.
- Ich habe alle Personen erwähnt, welche die Arbeit wesentlich unterstützt haben.

Ich nehme zur Kenntnis, dass die Arbeit mit elektronischen Hilfsmitteln auf Plagiate überprüft werden kann.

Ort, Datum	**Unterschrift(en)**
Baden, 30.05.2016	

Bei Gruppenarbeiten sind die Namen aller Verfasserinnen und Verfasser erforderlich. Durch die Unterschriften bürgen sie gemeinsam für den gesamten Inhalt dieser schriftlichen Arbeit.

8. Quellenverzeichnis

- Surface Simplification Using Quadric Error Metric [Garland et al. 1997]
- Large-Scale Semantic 3D Reconstruction: an Adaptive Multi-Resolution Model for Multi-Class Volumetric Labeling [Blaha et al. 2016]
- A Comparison and Evaluation of Multi-View Stereo Reconstruction Algorithmus [Seitz et al. 2006]
- Stanford Buddha [http://graphics.stanford.edu/data/3Dscanrep/]
- [Geomagic Control 2015]
- [Meshlab 2016]
- [wikipedia.org]
- Mathworks File Exchange [http://www.mathworks.com/matlabcentral/fileexchange/]
    - compute_edges [Wasmeier 2004]
    - helmert3d [Peyr 2004]

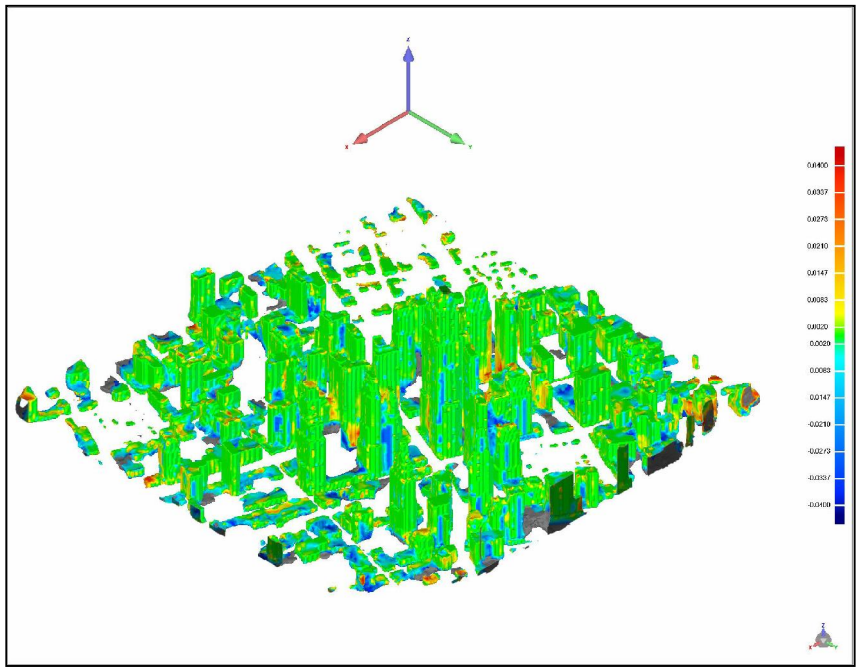
9. Anhang

Folgend sind die automatisch generierten Geomagic-Dokumentationen der Vergleiche angefügt. Für die Vergleiche der Klassen von Octree und Grid jeweils mit einer Voxelgrösse bzw. für den Vergleich mit der Groundtruth mit fünf Voxelgrössen als grafischem Grenzwert.

Geomagic Control

Geomagic Control Report

Date Inspected: 3/23/2016
Date Generated: 3/23/2016, 2:54 pm



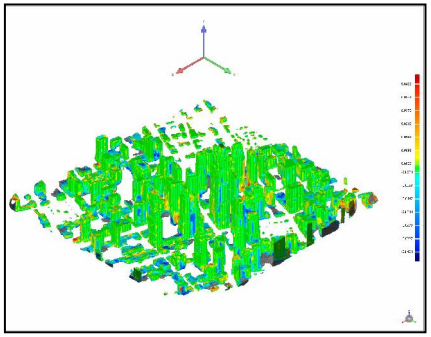
Author: miottil:PF-PC02
Client Name: 3D Systems, Inc.
Reference Model: oc_optModel_building_loop3
Test Model: optModel_building

3D Comparison Results

Reference Model	oc_optModel_building_loop3
Test Model	optModel_building
No. of Data Points	241634
# Outliers	28794

Tolerance Type	3D Deviation
Units	mm
Max. Critical	0.0400
Max. Nominal	0.0020
Min. Nominal	-0.0020
Min. Critical	-0.0400

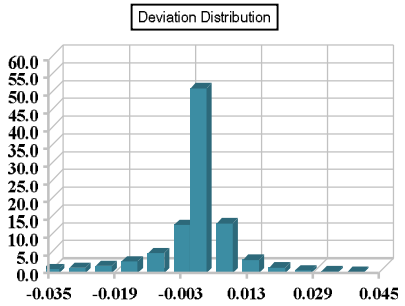
Deviation	
Max. Upper Deviation	0.0400
Max. Lower Deviation	-0.0400
Average Deviation	0.0034 / -0.0073
Standard Deviation	0.0092



Deviation Distribution

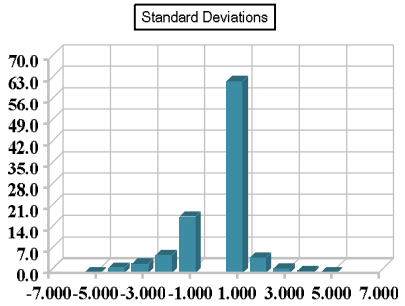
>=Min	<Max	# Points	%
-0.0400	-0.0337	2607	1.0789
-0.0337	-0.0273	3516	1.4551
-0.0273	-0.0210	4814	1.9923
-0.0210	-0.0147	7927	3.2806
-0.0147	-0.0083	13561	5.6122
-0.0083	-0.0020	32924	13.6256
-0.0020	0.0020	125789	52.0577
0.0020	0.0083	33886	14.0237
0.0083	0.0147	9077	3.7565
0.0147	0.0210	3768	1.5594
0.0210	0.0273	1814	0.7507
0.0273	0.0337	1178	0.4875
0.0337	0.0400	773	0.3199

Out of Upper Critical	0	0.0000
Out of Lower Critical	0	0.0000

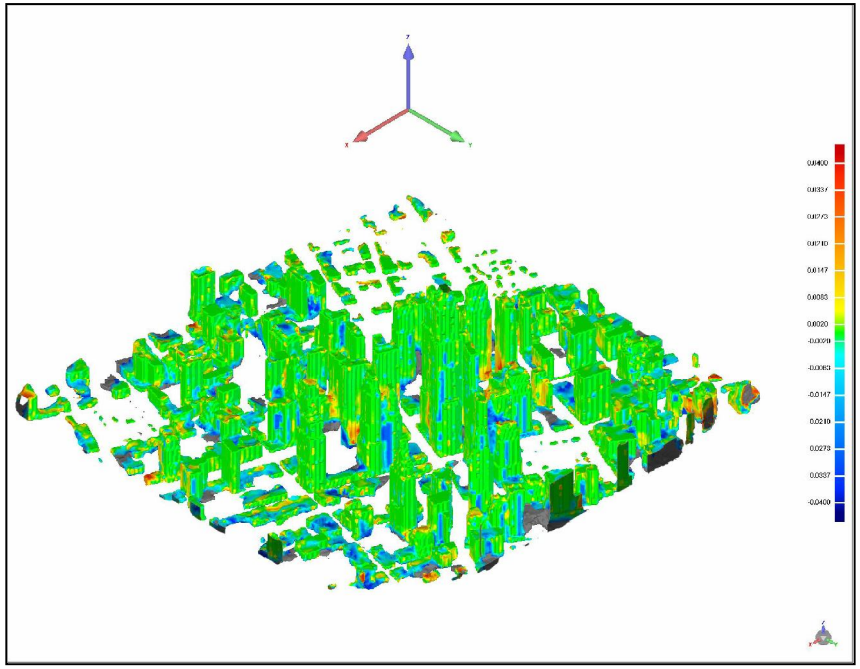


Standard Deviations

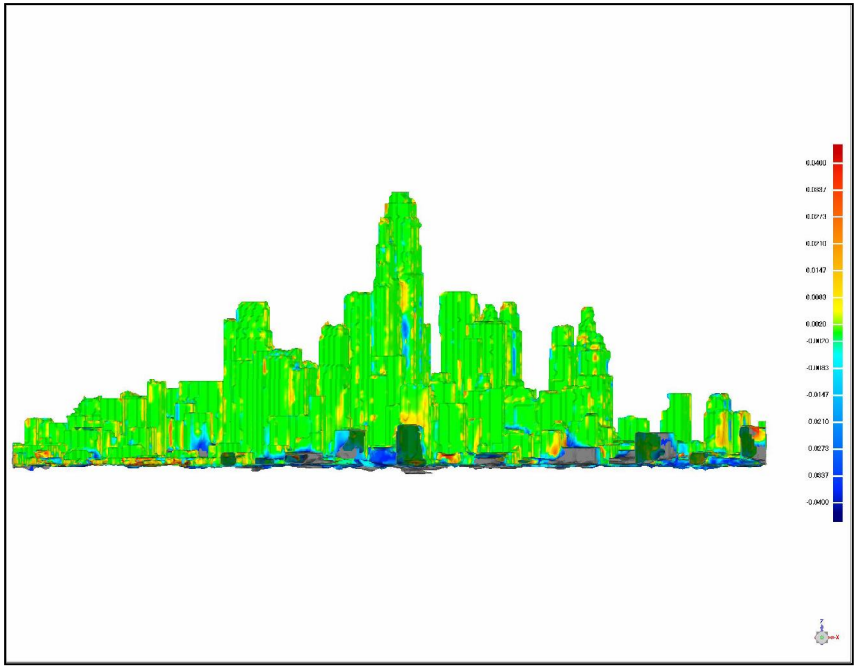
Distribution (+/-)	# Points	%
-6 * Std. Dev.	0	0.0000
-5 * Std. Dev.	694	0.2872
-4 * Std. Dev.	4170	1.7258
-3 * Std. Dev.	7527	3.1150
-2 * Std. Dev.	14261	5.9019
-1 * Std. Dev.	44685	18.4928
1 * Std. Dev.	152274	63.0184
2 * Std. Dev.	12143	5.0254
3 * Std. Dev.	3653	1.5118
4 * Std. Dev.	1692	0.7002
5 * Std. Dev.	535	0.2214
6 * Std. Dev.	0	0.0000



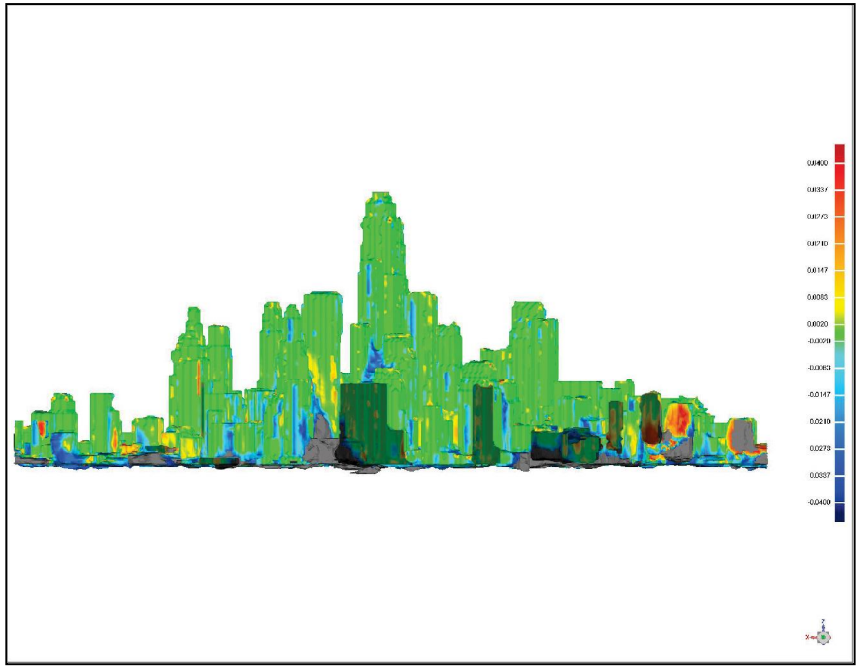
Predefined: Isometric



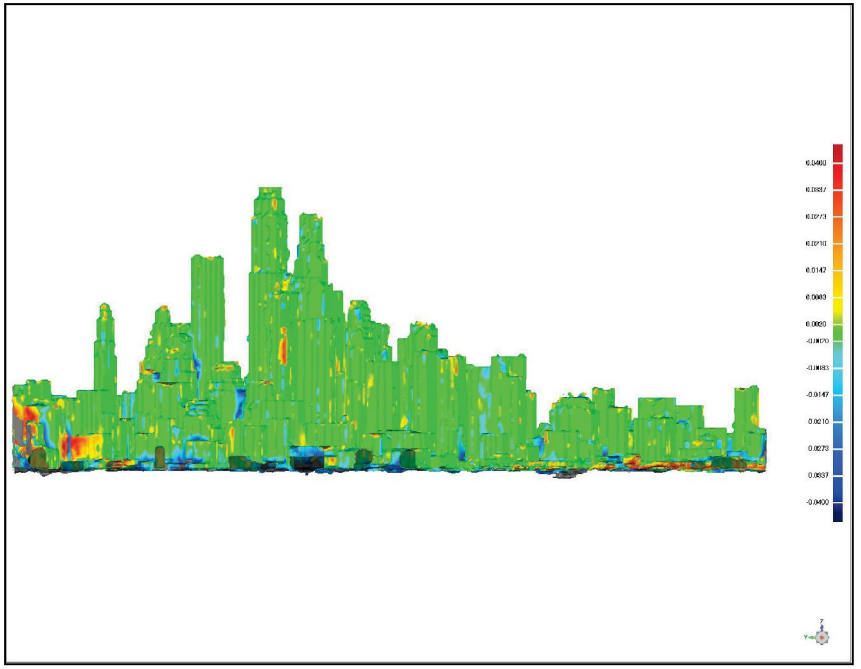
Predefined: Front



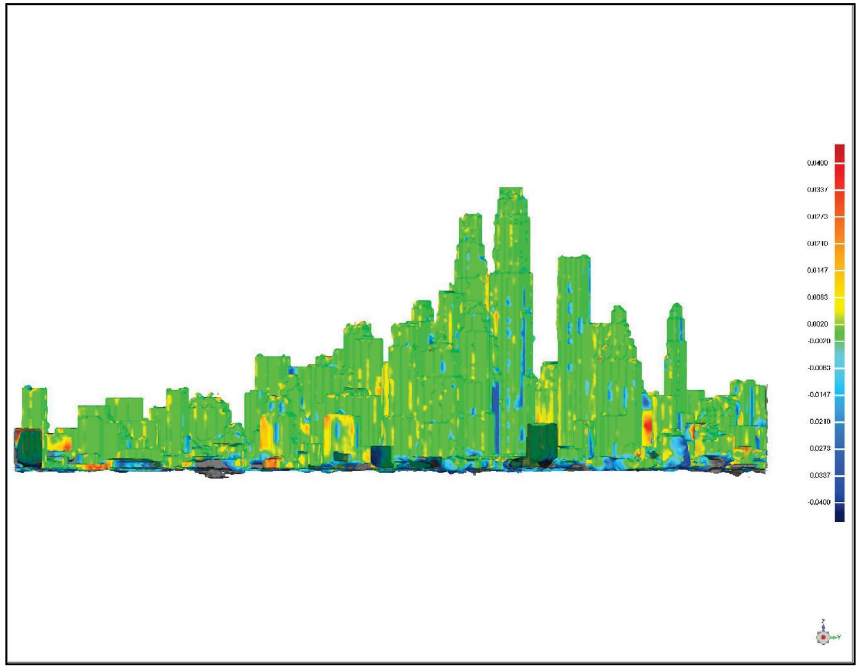
Predefined: Back



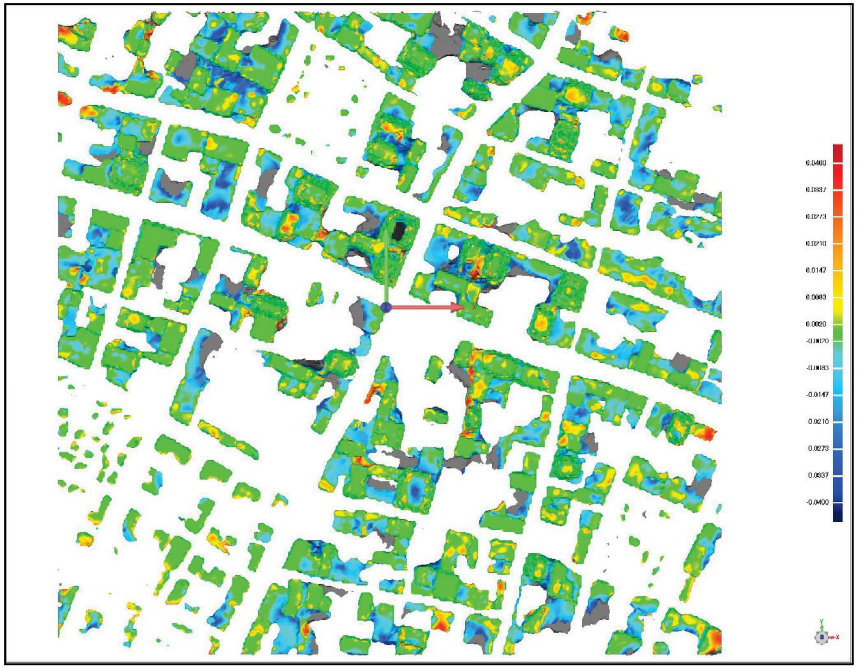
Predefined: Left



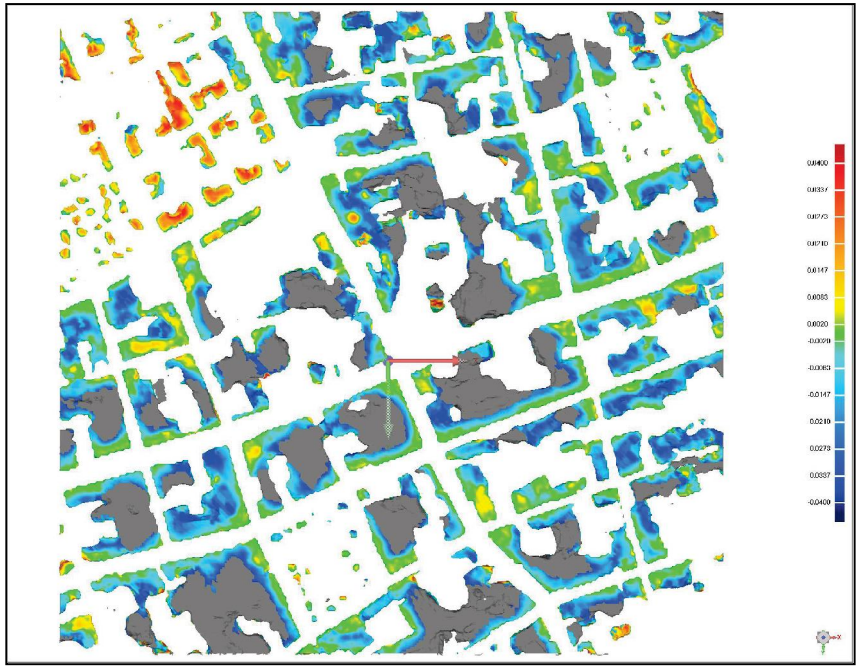
Predefined: Right



Predefined: Top



Predefined: Bottom



Location Set: Upper and Lower Deviations

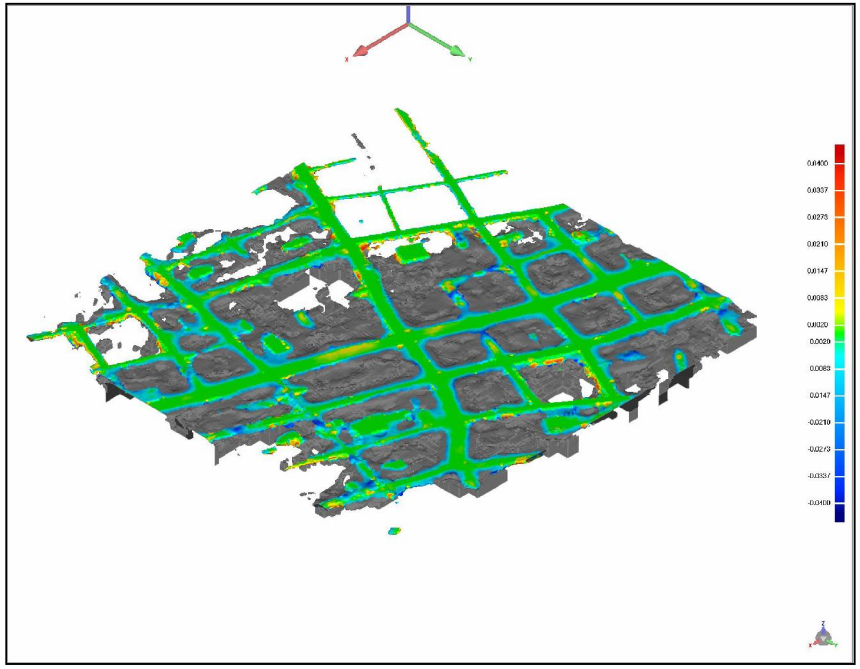
Units: mm

Name	Dev	Status	Upper Tol	Lower Tol	Ref X	Ref Y	Ref Z	Radius	Dev X	Dev Y	Dev Z	Measured X	Measured Y	Measured Z	Proj. Dir X	Proj. Dir Y	Proj. Dir Z
Lower Dev.	-0.0400				2.0413	-2.9137	-8.4022	n/a	0.0290	0.0047	0.0272	2.0703	-2.9089	-8.3750	-0.7246	-0.1177	-0.6790
Upper Dev.	0.0400				1.6193	0.5406	-7.0139	n/a	0.0213	-0.0333	0.0061	1.6406	0.5073	-7.0078	0.5330	-0.8323	0.1524

Geomagic Control

Geomagic Control Report

Date Inspected: 3/23/2016
Date Generated: 3/23/2016, 3:13 pm



Author: miottil:PF-PC02
Client Name: 3D Systems, Inc.
Reference Model: oc_optModel_ground_loop3
Test Model: optModel_ground

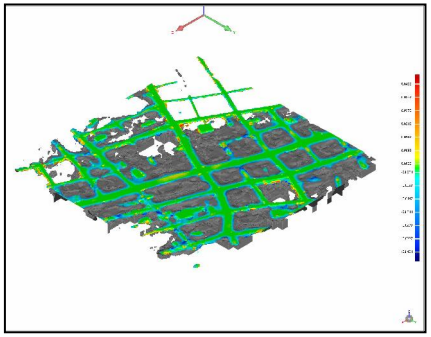
Geomagic Control

3D Comparison Results

Reference Model	oc_optModel_ground_loop3
Test Model	optModel_ground
No. of Data Points	48094
# Outliers	61209

Tolerance Type	3D Deviation
Units	mm
Max. Critical	0.0400
Max. Nominal	0.0020
Min. Nominal	-0.0020
Min. Critical	-0.0400

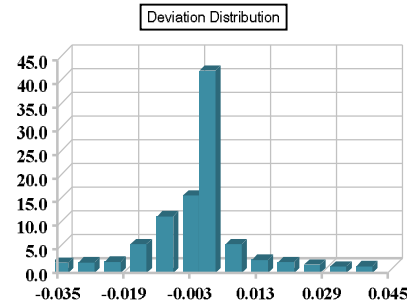
Deviation	
Max. Upper Deviation	0.0400
Max. Lower Deviation	-0.0400
Average Deviation	0.0052 / -0.0102
Standard Deviation	0.0123



Deviation Distribution

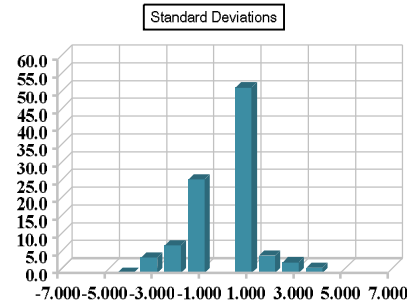
>=Min	<Max	# Points	%
-0.0400	-0.0337	1050	2.1832
-0.0337	-0.0273	1088	2.2622
-0.0273	-0.0210	1160	2.4119
-0.0210	-0.0147	2937	6.1068
-0.0147	-0.0083	5779	12.0161
-0.0083	-0.0020	7905	16.4366
-0.0020	0.0020	20554	42.7371
0.0020	0.0083	2926	6.0839
0.0083	0.0147	1343	2.7924
0.0147	0.0210	1132	2.3537
0.0210	0.0273	855	1.7778
0.0273	0.0337	665	1.3827
0.0337	0.0400	699	1.4534

Out of Upper Critical	0	0.0000
Out of Lower Critical	1	0.0021

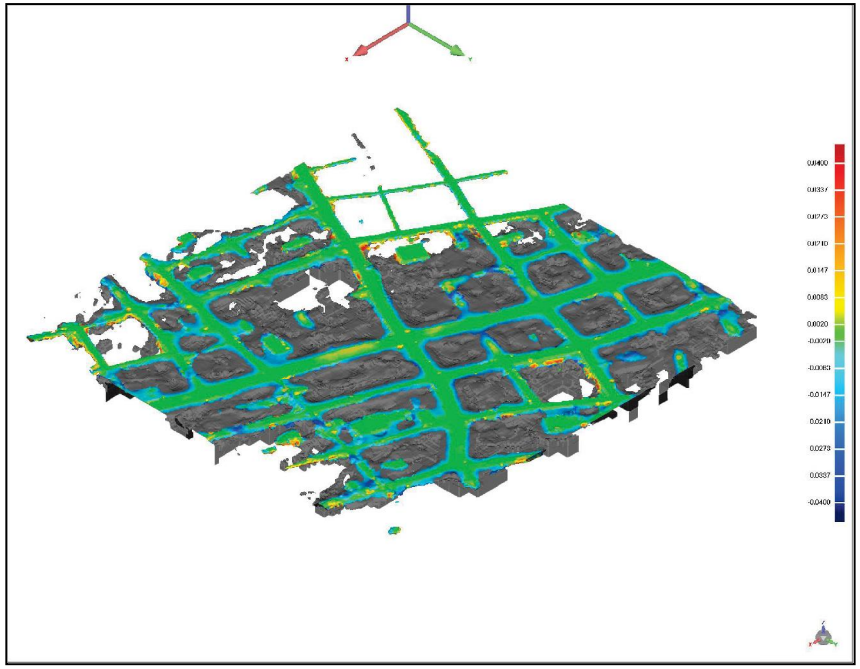


Standard Deviations

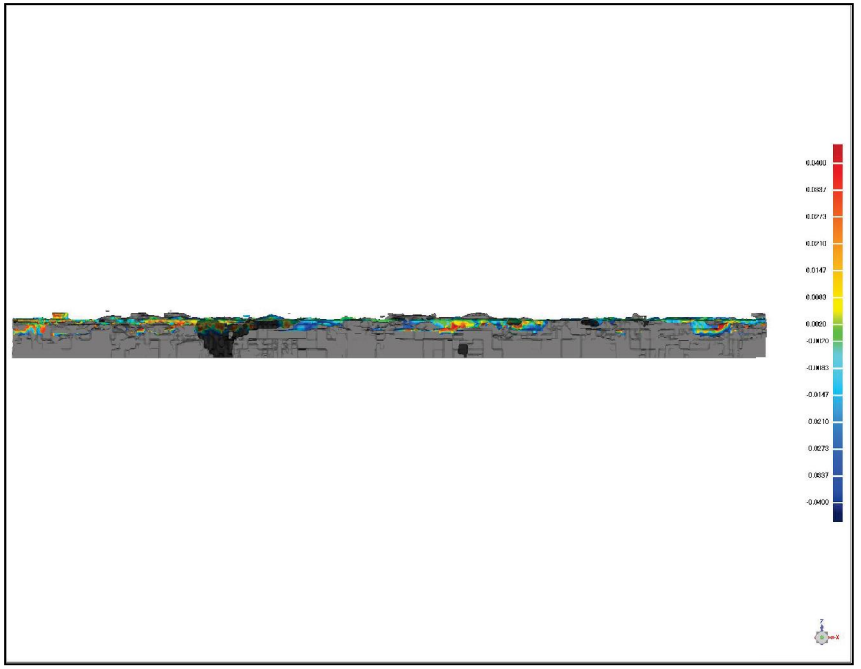
Distribution (+/-)	# Points	%
-6 * Std. Dev.	0	0.0000
-5 * Std. Dev.	0	0.0000
-4 * Std. Dev.	19	0.0395
-3 * Std. Dev.	2084	4.3332
-2 * Std. Dev.	3720	7.7349
-1 * Std. Dev.	12687	26.3796
1 * Std. Dev.	25120	52.2310
2 * Std. Dev.	2332	4.8488
3 * Std. Dev.	1448	3.0108
4 * Std. Dev.	684	1.4222
5 * Std. Dev.	0	0.0000
6 * Std. Dev.	0	0.0000



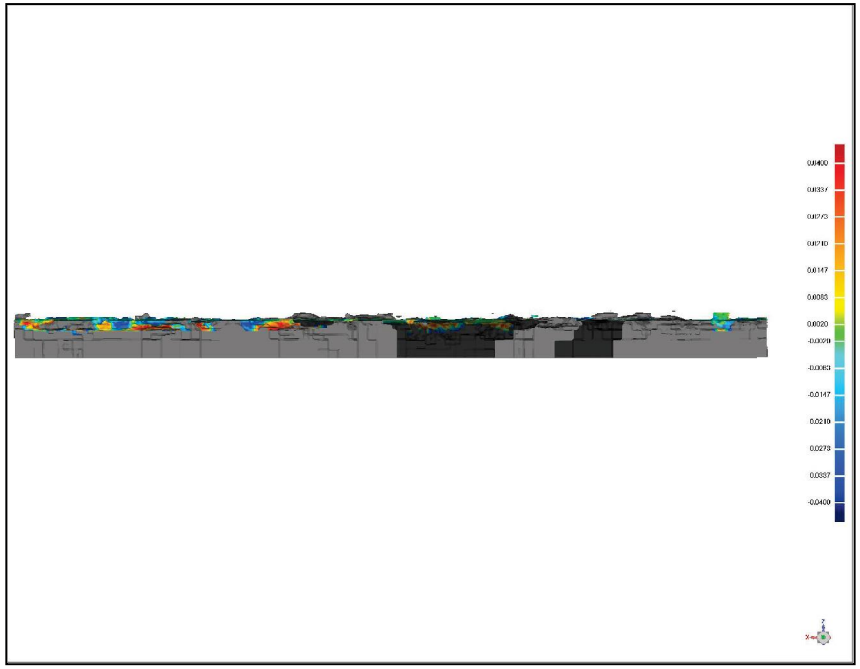
Predefined: Isometric



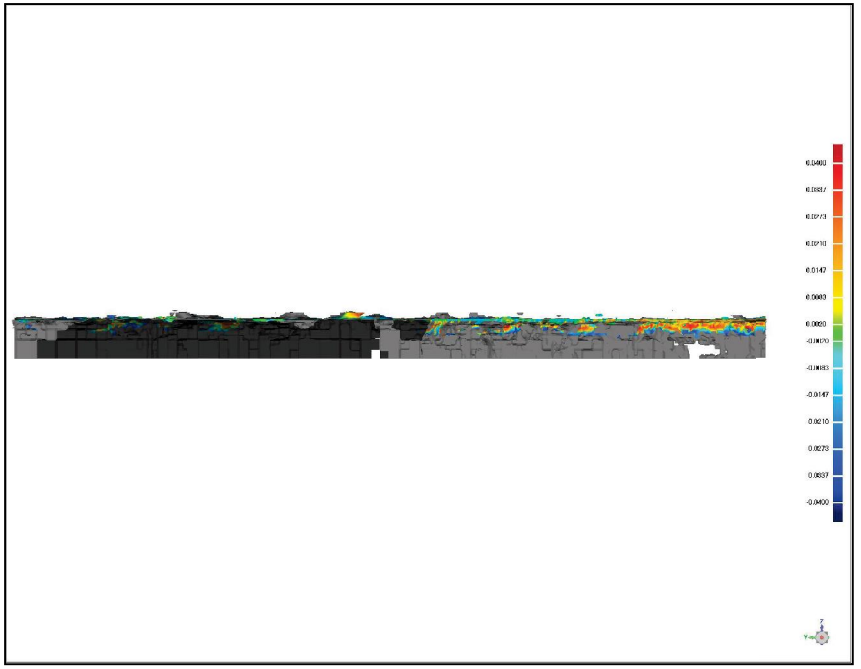
Predefined: Front



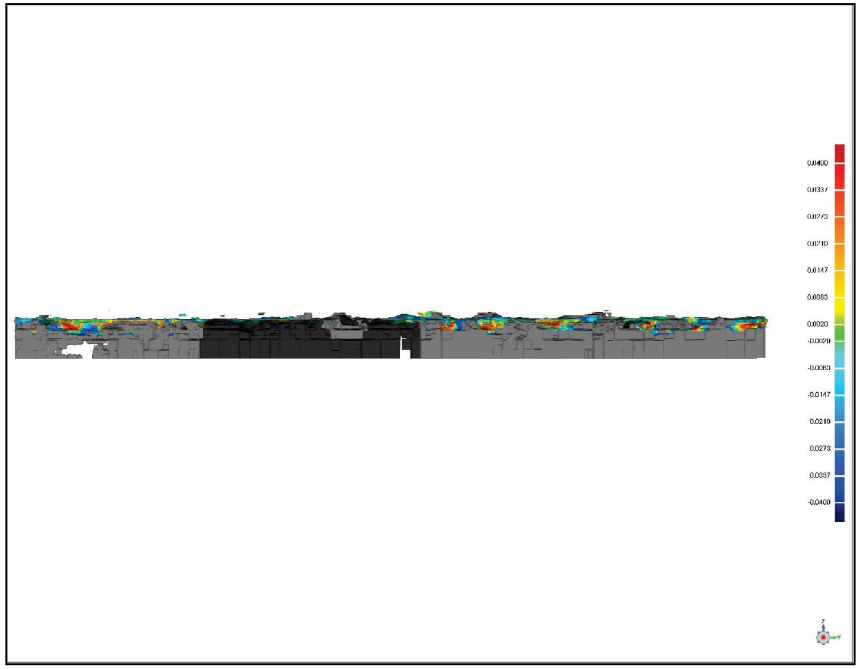
Predefined: Back



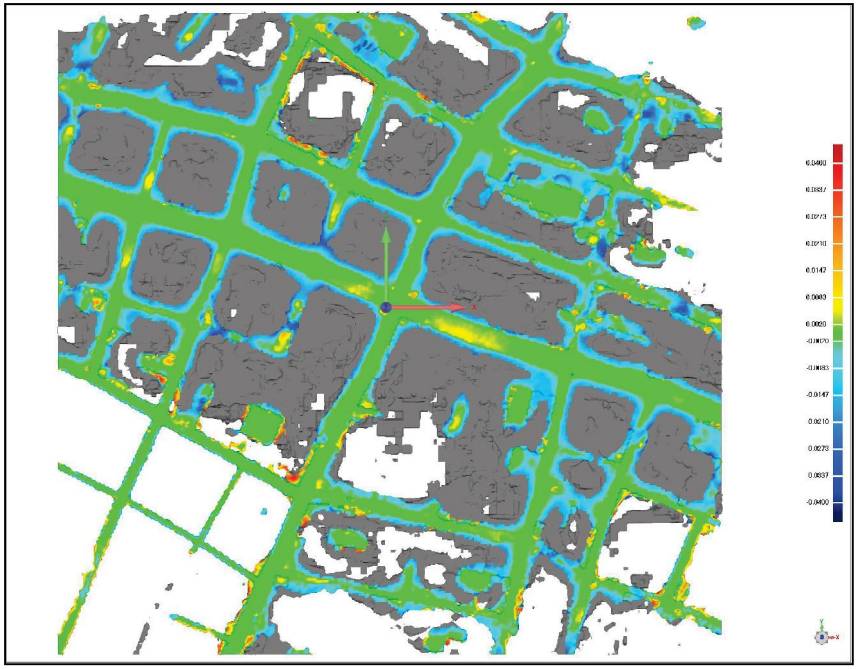
Predefined: Left



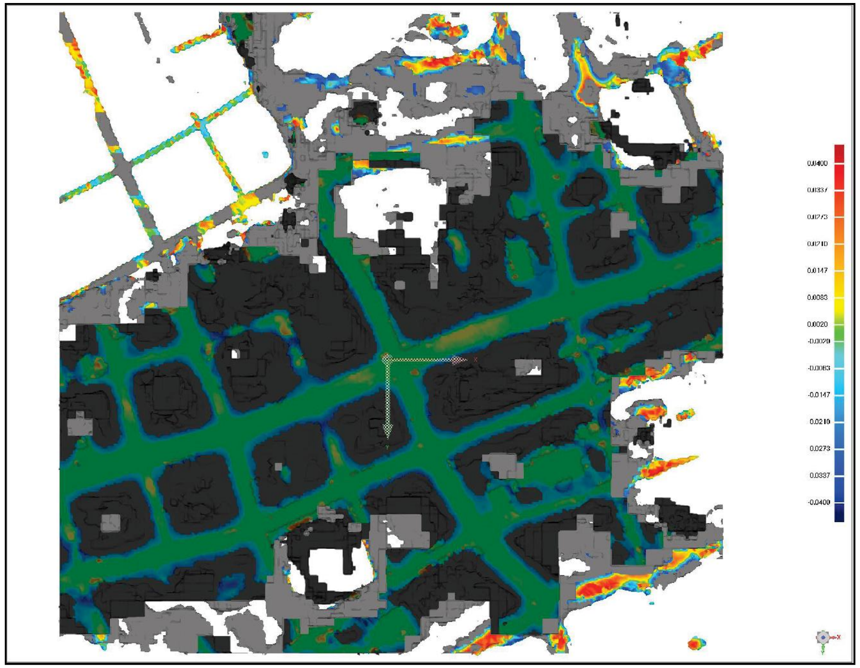
Predefined: Right



Predefined: Top



Predefined: Bottom



Location Set: Upper and Lower Deviations

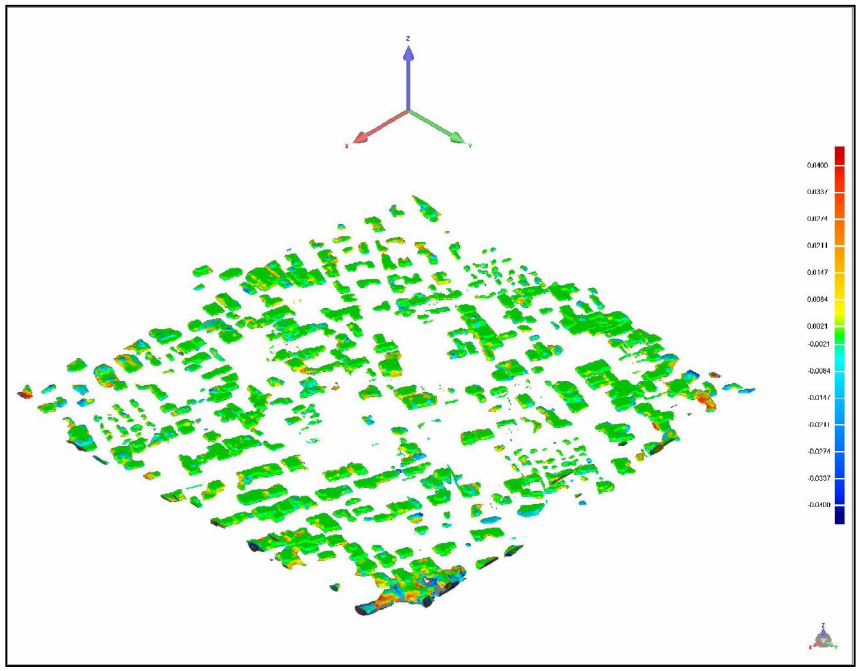
Units: mm

Name	Dev	Status	Upper Tol	Lower Tol	Ref X	Ref Y	Ref Z	Radius	Dev X	Dev Y	Dev Z	Measure d X	Measure d Y	Measure d Z	Proj. Dir X	Proj. Dir Y	Proj. Dir Z
Lower Dev.	-0.0400				-5.5452	3.4687	-8.3738	n/a	-0.0017	0.0078	-0.0392	-5.5469	3.4766	-8.4130	0.0416	-0.1957	0.9798
Upper Dev.	0.0400				-1.4916	-4.0203	-8.5703	n/a	0.0073	-0.0032	0.0392	-1.4844	-4.0234	-8.5311	0.1813	-0.0797	0.9802

Geomagic Control

Geomagic Control Report

Date Inspected: 3/23/2016
Date Generated: 3/23/2016, 3:20 pm



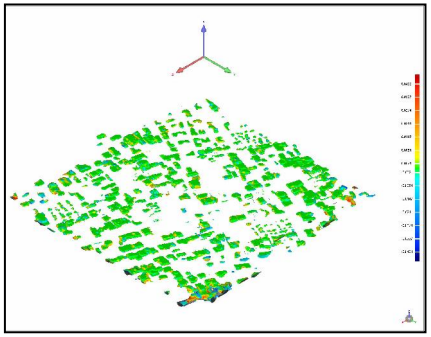
Author: miottil:PF-PC02
Client Name: 3D Systems, Inc.
Reference Model: oc_optModel_roof_loop3
Test Model: optModel_roof

3D Comparison Results

Reference Model	oc_optModel_roof_loop3
Test Model	optModel_roof
No. of Data Points	96403
# Outliers	5111

Tolerance Type	3D Deviation
Units	mm
Max. Critical	0.0400
Max. Nominal	0.0021
Min. Nominal	-0.0021
Min. Critical	-0.0400

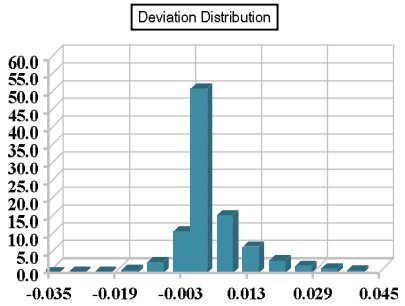
Deviation	
Max. Upper Deviation	0.0400
Max. Lower Deviation	-0.0399
Average Deviation	0.0053 / -0.0042
Standard Deviation	0.0086



Deviation Distribution

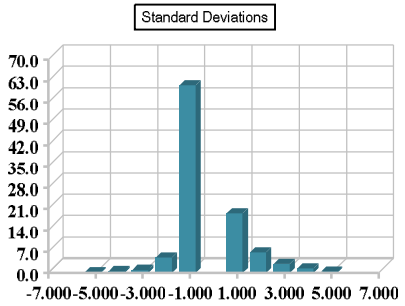
>=Min	<Max	# Points	%
-0.0400	-0.0337	198	0.2054
-0.0337	-0.0274	344	0.3568
-0.0274	-0.0211	411	0.4263
-0.0211	-0.0147	859	0.8911
-0.0147	-0.0084	2959	3.0694
-0.0084	-0.0021	11326	11.7486
-0.0021	0.0021	50047	51.9144
0.0021	0.0084	15649	16.2329
0.0084	0.0147	7153	7.4199
0.0147	0.0211	3453	3.5818
0.0211	0.0274	1934	2.0062
0.0274	0.0337	1274	1.3215
0.0337	0.0400	796	0.8257

Out of Upper Critical	0	0.0000
Out of Lower Critical	0	0.0000



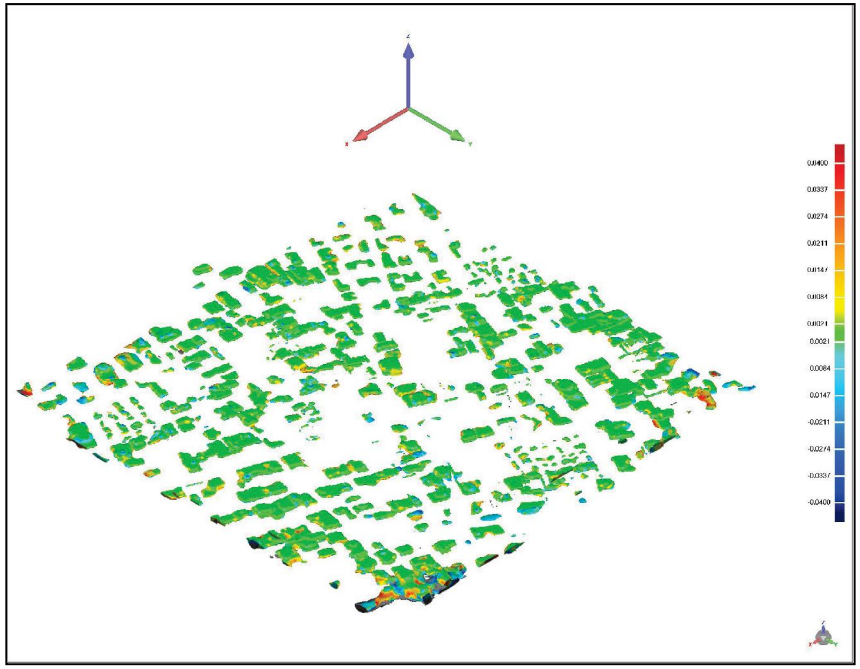
Standard Deviations

Distribution (+/-)	# Points	%
-6 * Std. Dev.	0	0.0000
-5 * Std. Dev.	273	0.2832
-4 * Std. Dev.	522	0.5415
-3 * Std. Dev.	968	1.0041
-2 * Std. Dev.	4921	5.1046
-1 * Std. Dev.	59460	61.6786
1 * Std. Dev.	18883	19.5876
2 * Std. Dev.	6579	6.8245
3 * Std. Dev.	2872	2.9792
4 * Std. Dev.	1481	1.5363
5 * Std. Dev.	444	0.4606
6 * Std. Dev.	0	0.0000



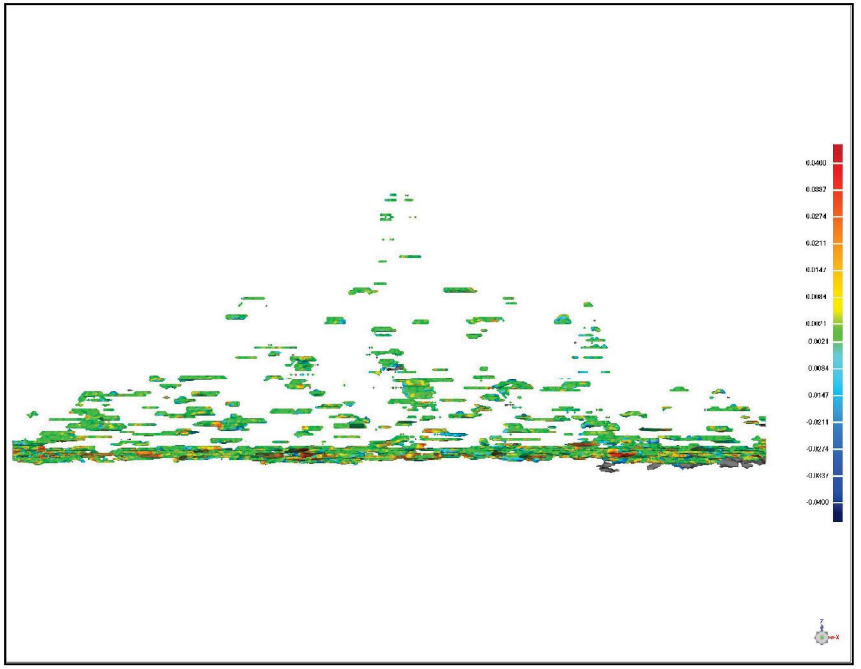
Date: 3/23/2016, 3:20 pm

Predefined: Isometric

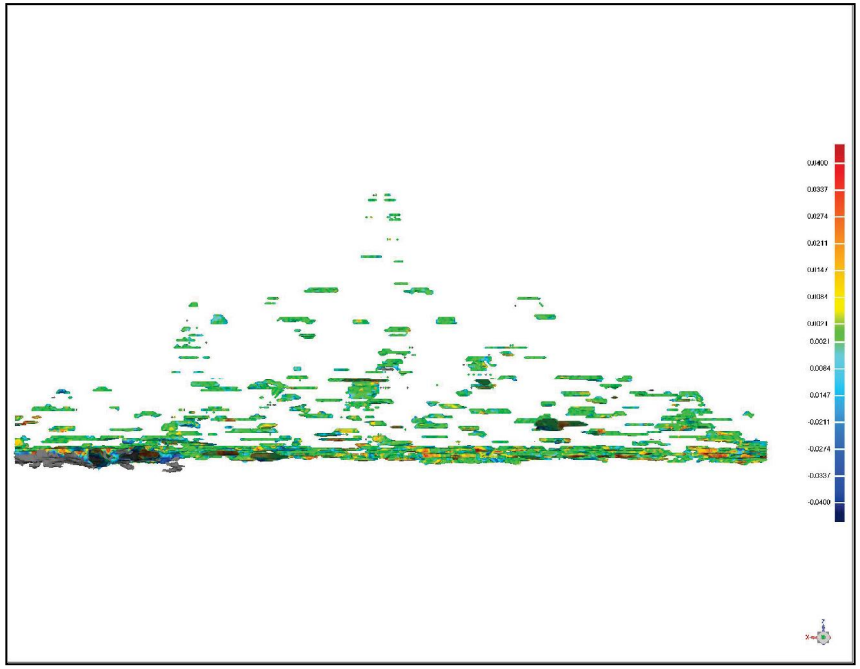


Date: 3/23/2016, 3:20 pm

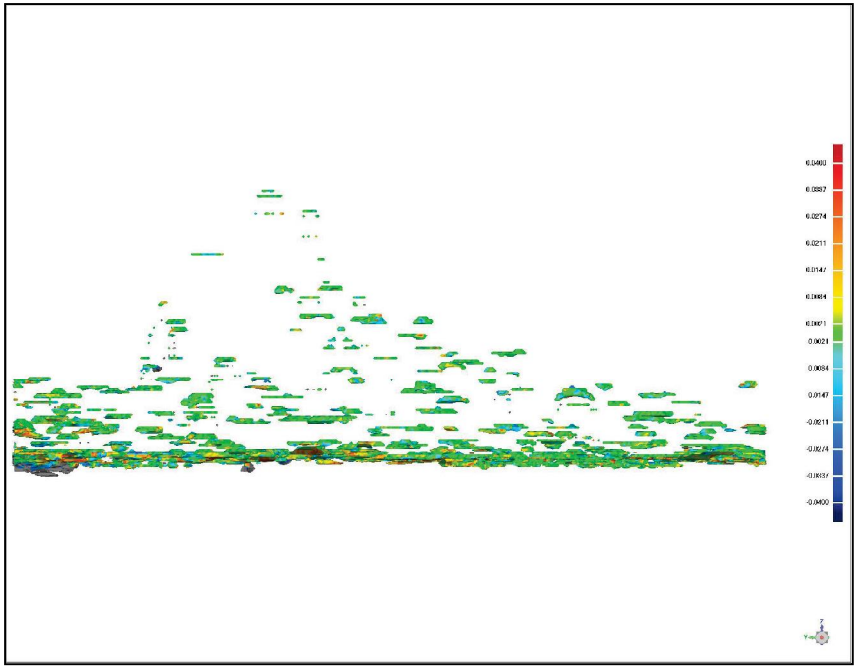
Predefined: Front



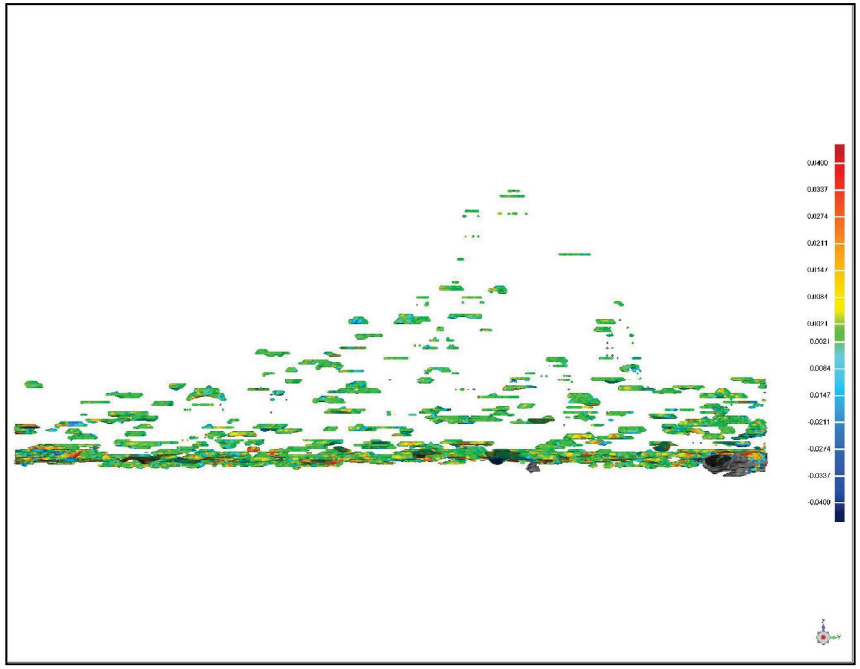
Predefined: Back



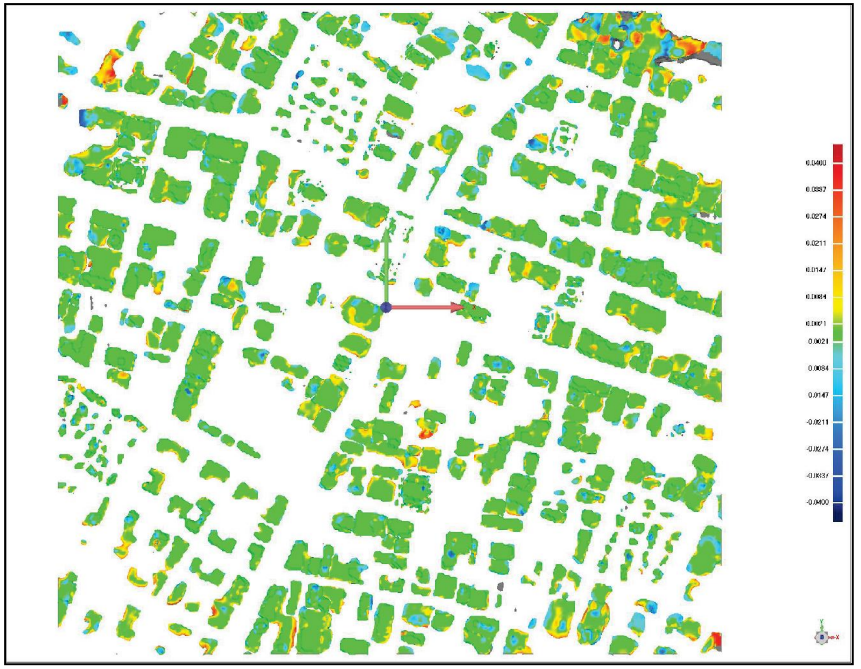
Predefined: Left



Predefined: Right



Predefined: Top



Predefined: Bottom

Location Set: Upper and Lower Deviations

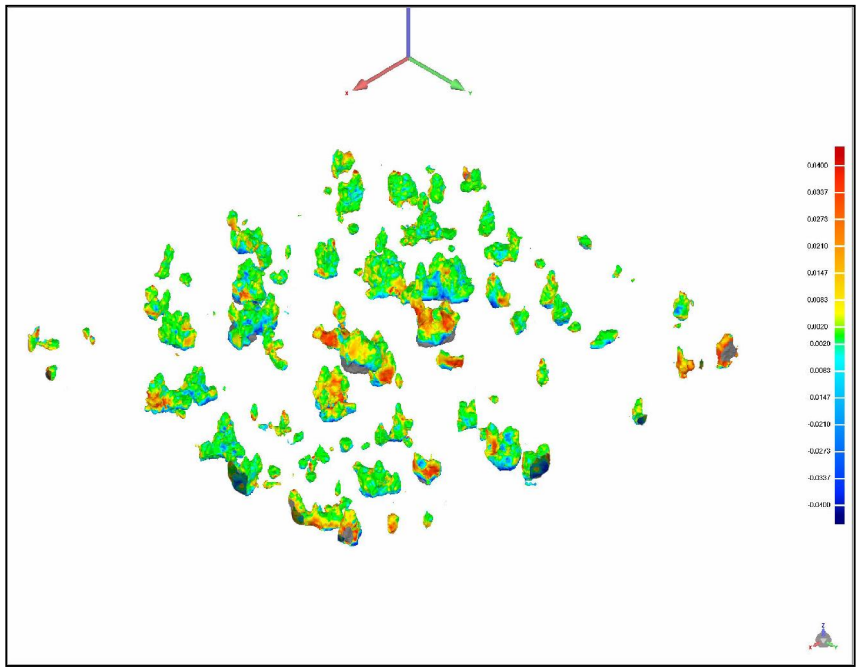
Units: mm

Name	Dev	Status	Upper Tol	Lower Tol	Ref X	Ref Y	Ref Z	Radius	Dev X	Dev Y	Dev Z	Measured X	Measured Y	Measured Z	Proj. Dir X	Proj. Dir Y	Proj. Dir Z
Lower Dev.	-0.0399				-1.8052	1.8863	-8.3359	n/a	0.0083	-0.0005	0.0391	-1.7969	1.8858	-8.2969	-0.2086	0.0127	-0.9779
Upper Dev.	0.0400				-0.2039	-4.1016	-8.2188	n/a	0.0086	-0.0006	-0.0391	-0.1953	-4.1022	-8.2578	0.2139	-0.0150	-0.9767

Geomagic Control

Geomagic Control Report

Date Inspected: 3/23/2016
Date Generated: 3/23/2016, 3:27 pm



Author: miottil:PF-PC02
Client Name: 3D Systems, Inc.
Reference Model: oc_optModel_vegetation_loop3
Test Model: optModel_vegetation

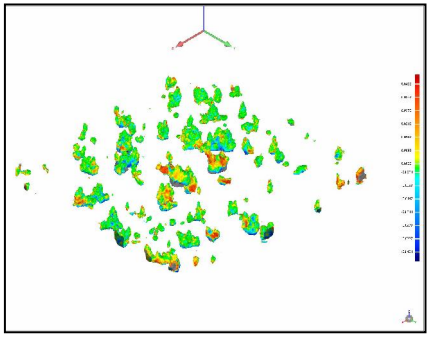
Geomagic Control

3D Comparison Results

Reference Model	oc_optModel_vegetation_loop3
Test Model	optModel_vegetation
No. of Data Points	77143
# Outliers	11980

Tolerance Type	3D Deviation
Units	mm
Max. Critical	0.0400
Max. Nominal	0.0020
Min. Nominal	-0.0020
Min. Critical	-0.0400

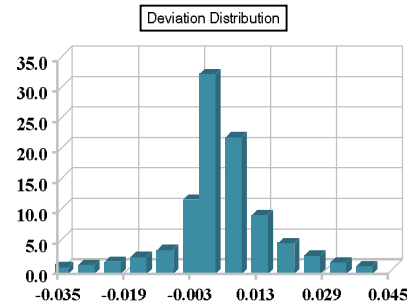
Deviation	
Max. Upper Deviation	0.0400
Max. Lower Deviation	-0.0400
Average Deviation	0.0077 / -0.0082
Standard Deviation	0.0119



Deviation Distribution

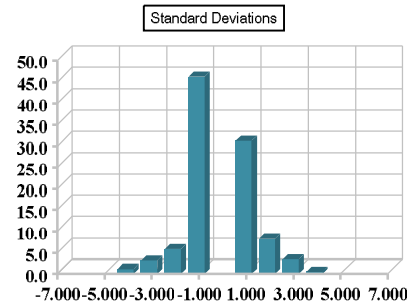
>=Min	<Max	# Points	%
-0.0400	-0.0337	878	1.1381
-0.0337	-0.0273	1140	1.4778
-0.0273	-0.0210	1549	2.0080
-0.0210	-0.0147	2168	2.8104
-0.0147	-0.0083	3063	3.9705
-0.0083	-0.0020	9426	12.2189
-0.0020	0.0020	25347	32.8572
0.0020	0.0083	17346	22.4855
0.0083	0.0147	7465	9.6768
0.0147	0.0210	3935	5.1009
0.0210	0.0273	2325	3.0139
0.0273	0.0337	1489	1.9302
0.0337	0.0400	1012	1.3118

Out of Upper Critical	0	0.0000
Out of Lower Critical	0	0.0000

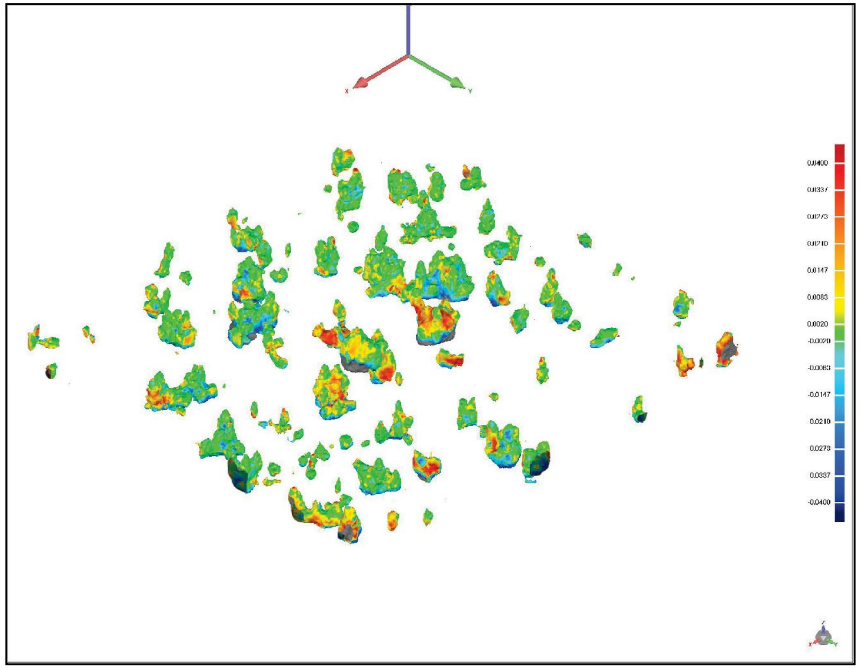


Standard Deviations

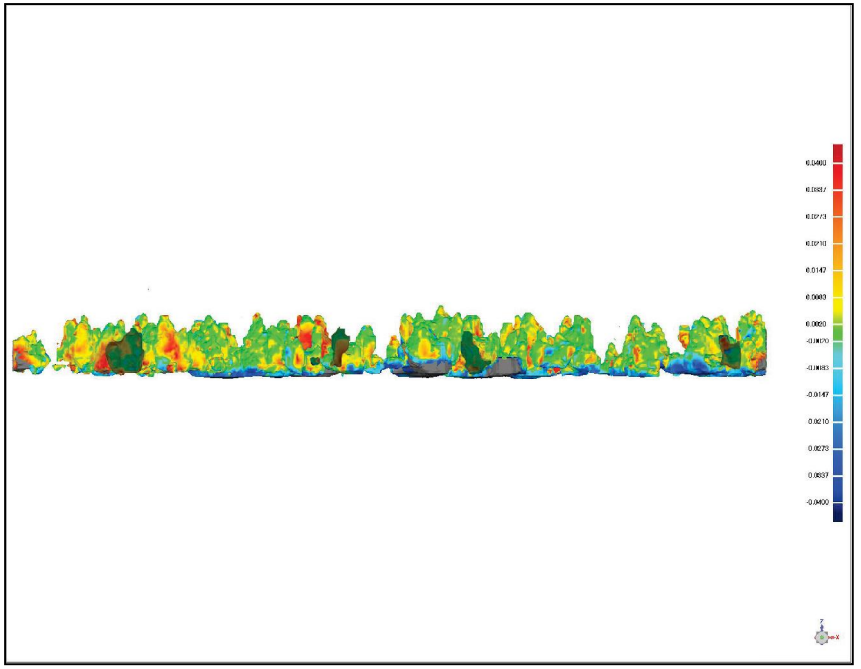
Distribution (+/-)	# Points	%
-6 * Std. Dev.	0	0.0000
-5 * Std. Dev.	0	0.0000
-4 * Std. Dev.	894	1.1589
-3 * Std. Dev.	2473	3.2057
-2 * Std. Dev.	4528	5.8696
-1 * Std. Dev.	35676	46.2466
1 * Std. Dev.	24141	31.2938
2 * Std. Dev.	6386	8.2781
3 * Std. Dev.	2648	3.4326
4 * Std. Dev.	397	0.5146
5 * Std. Dev.	0	0.0000
6 * Std. Dev.	0	0.0000



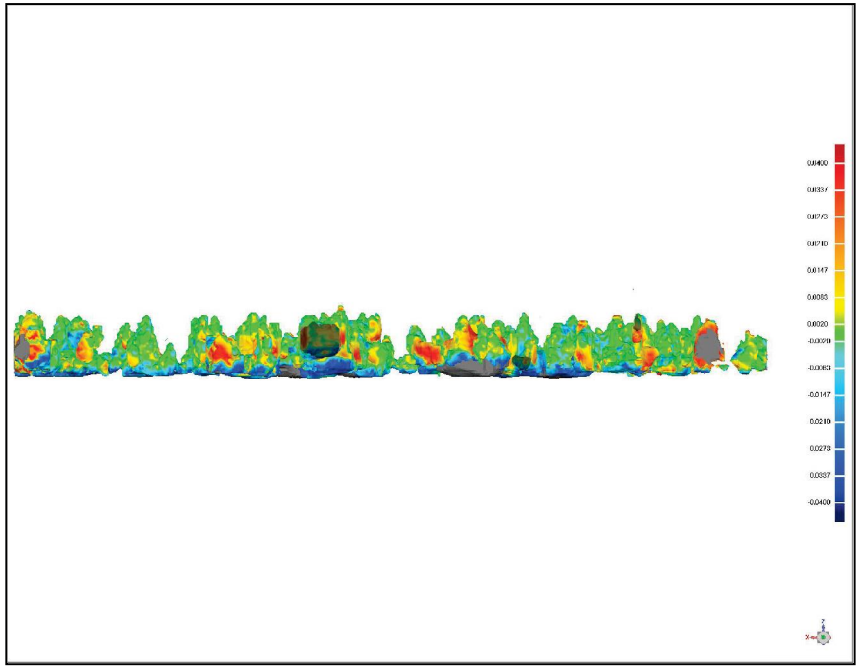
Predefined: Isometric



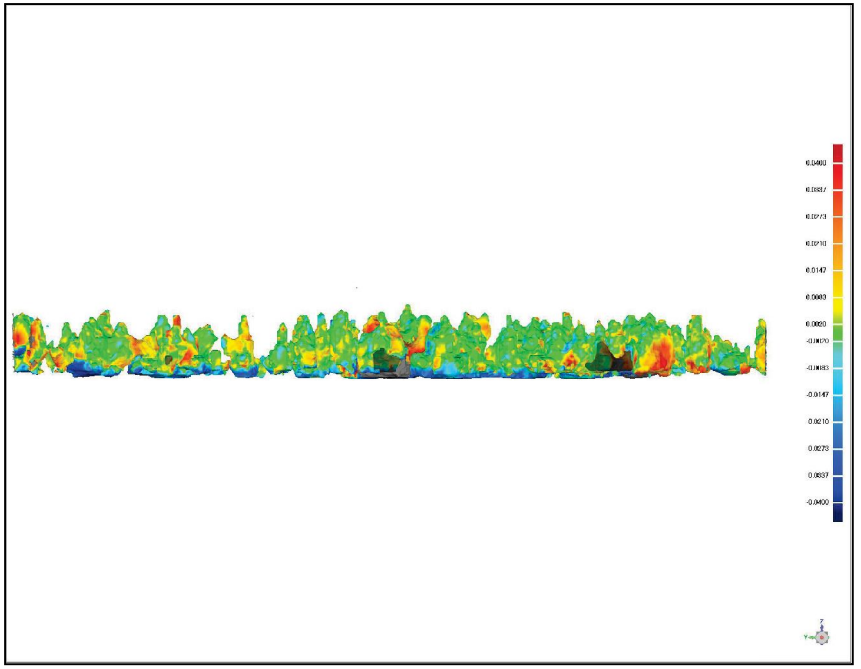
Predefined: Front



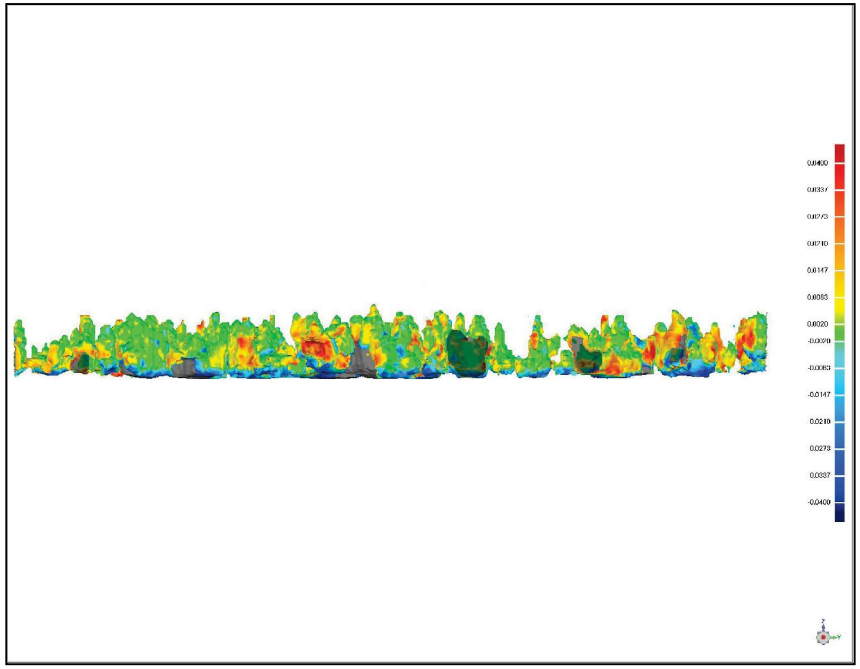
Predefined: Back



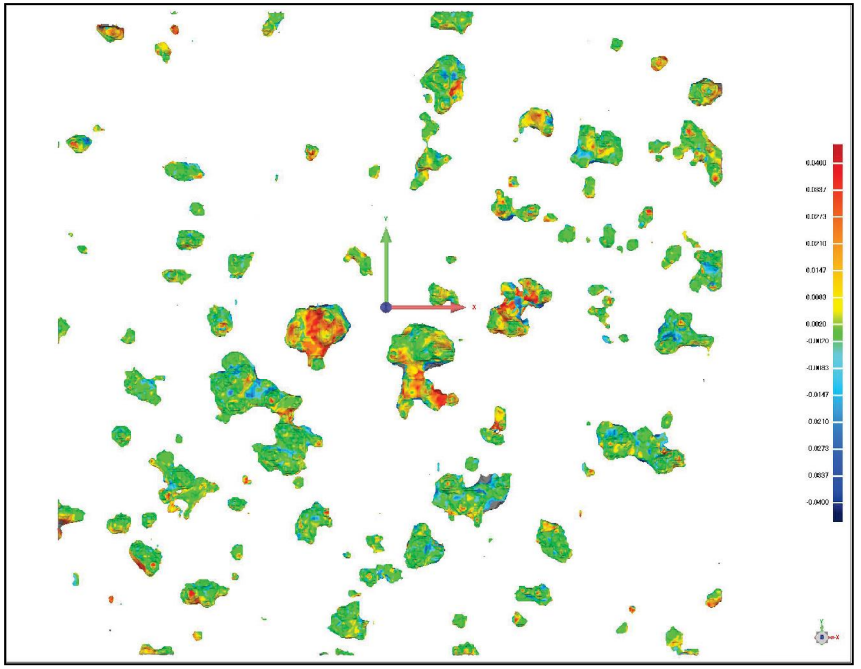
Predefined: Left



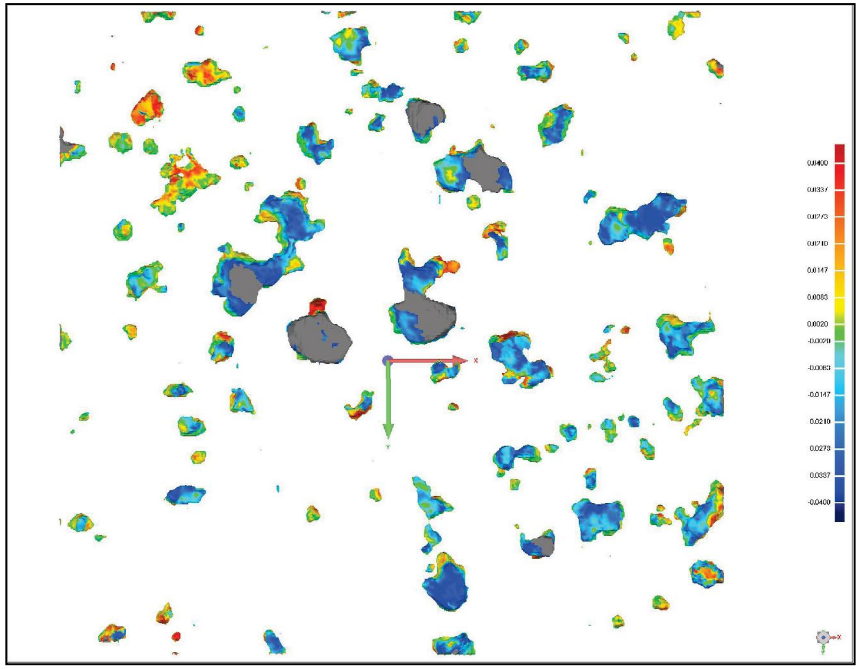
Predefined: Right



Predefined: Top



Predefined: Bottom



Location Set: Upper and Lower Deviations

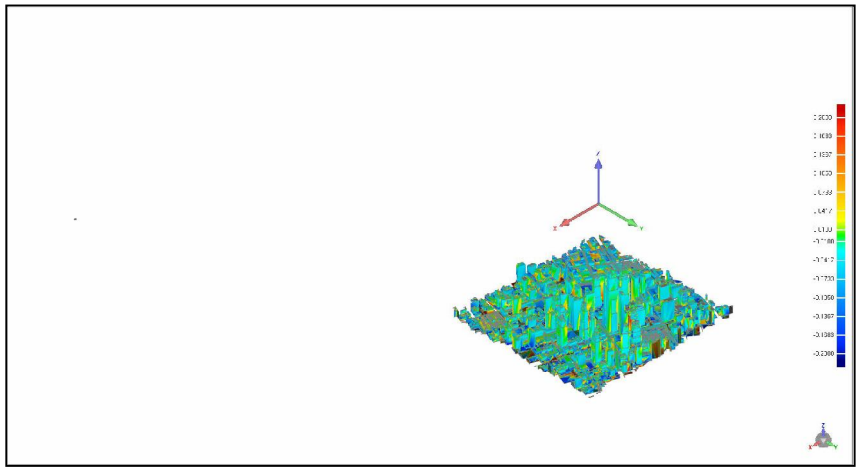
Units: mm

Name	Dev	Status	Upper Tol	Lower Tol	Ref X	Ref Y	Ref Z	Radius	Dev X	Dev Y	Dev Z	Measured X	Measured Y	Measured Z	Proj. Dir X	Proj. Dir Y	Proj. Dir Z
Lower Dev.	-0.0400				5.0771	-2.9071	-8.3716	n/a	0.0074	0.0165	0.0357	5.0845	-2.8906	-8.3359	-0.1843	-0.4115	-0.8926
Upper Dev.	0.0400				-3.5902	-3.6719	-8.3629	n/a	-0.0036	0.0000	-0.0398	-3.5938	-3.6719	-8.4027	-0.0890	0.0000	-0.9960

Geomagic Control

Geomagic Control Report

Date Inspected: 3/24/2016
Date Generated: 3/24/2016, 2:51 pm



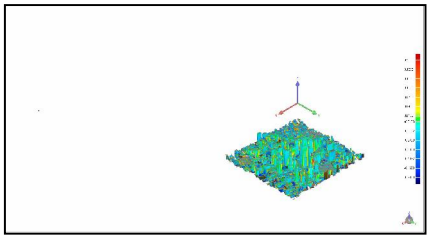
Author: miottil:PF-LAB10
Client Name: 3D Systems, Inc.
Reference Model: gt_transformed
Test Model: oc_whole_new

3D Comparison Results

Reference Model	gt_transformed
Test Model	oc_whole_new
No. of Data Points	452621
# Outliers	142288

Tolerance Type	3D Deviation
Units	mm
Max. Critical	0.2000
Max. Nominal	0.0100
Min. Nominal	-0.0100
Min. Critical	-0.2000

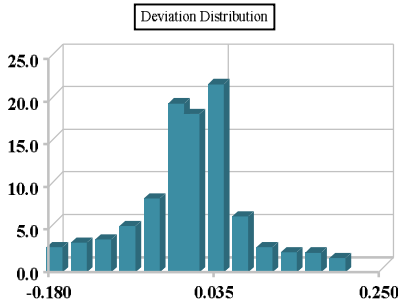
Deviation	
Max. Upper Deviation	0.2000
Max. Lower Deviation	-0.2000
Average Deviation	0.0456 / -0.0553
Standard Deviation	0.0711



Deviation Distribution

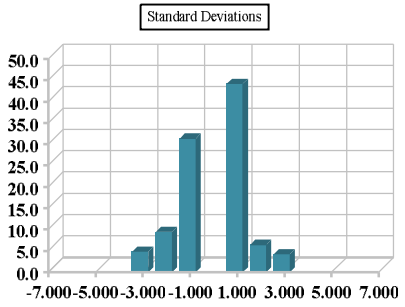
>=Min	<Max	# Points	%
-0.2000	-0.1683	13070	2.8876
-0.1683	-0.1367	15527	3.4305
-0.1367	-0.1050	17203	3.8008
-0.1050	-0.0733	24266	5.3612
-0.0733	-0.0417	39035	8.6242
-0.0417	-0.0100	89417	19.7554
-0.0100	0.0100	83832	18.5215
0.0100	0.0417	99629	22.0116
0.0417	0.0733	29429	6.5019
0.0733	0.1050	13205	2.9175
0.1050	0.1367	10382	2.2938
0.1367	0.1683	10248	2.2641
0.1683	0.2000	7377	1.6298

Out of Upper Critical	1	0.0002
Out of Lower Critical	0	0.0000

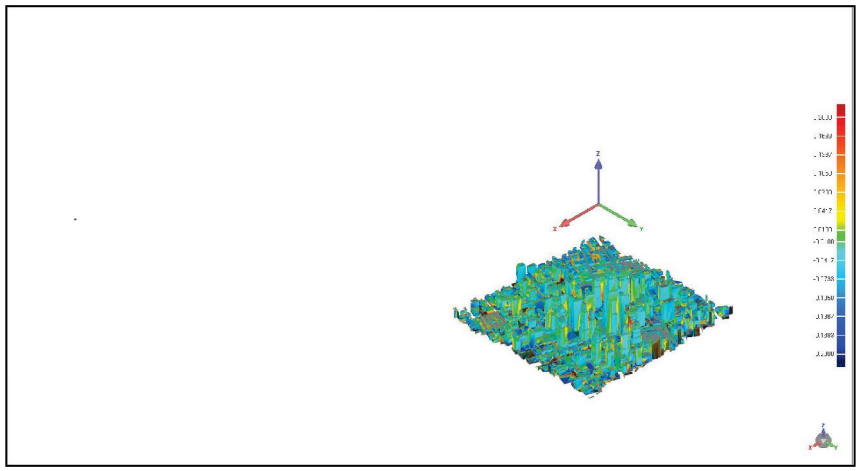


Standard Deviations

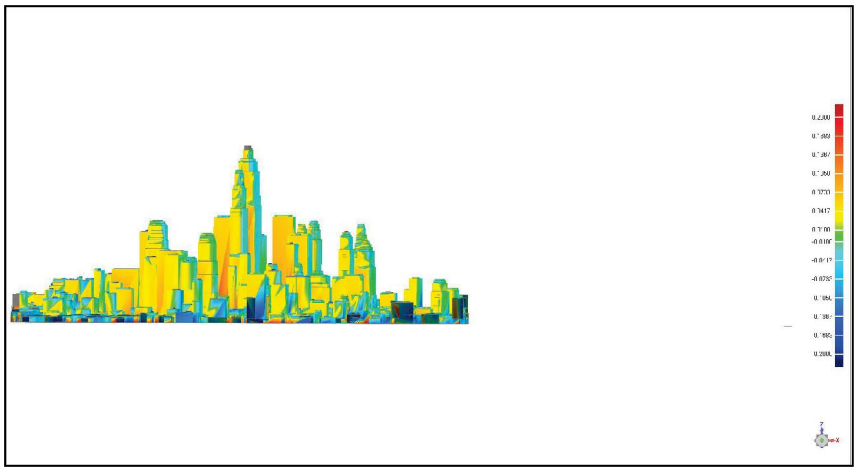
Distribution (+/-)	# Points	%
-6 * Std. Dev.	0	0.0000
-5 * Std. Dev.	0	0.0000
-4 * Std. Dev.	0	0.0000
-3 * Std. Dev.	21575	4.7667
-2 * Std. Dev.	42427	9.3736
-1 * Std. Dev.	141440	31.2491
1 * Std. Dev.	199833	44.1502
2 * Std. Dev.	28684	6.3373
3 * Std. Dev.	18662	4.1231
4 * Std. Dev.	0	0.0000
5 * Std. Dev.	0	0.0000
6 * Std. Dev.	0	0.0000



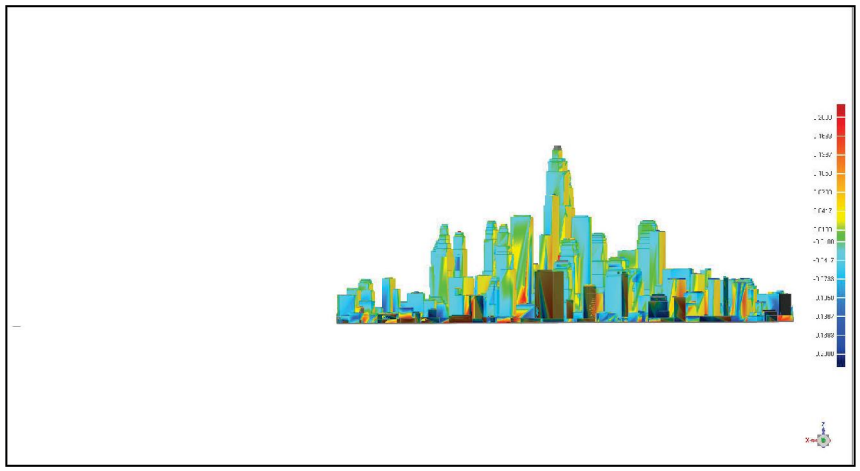
Predefined: Isometric



Predefined: Front

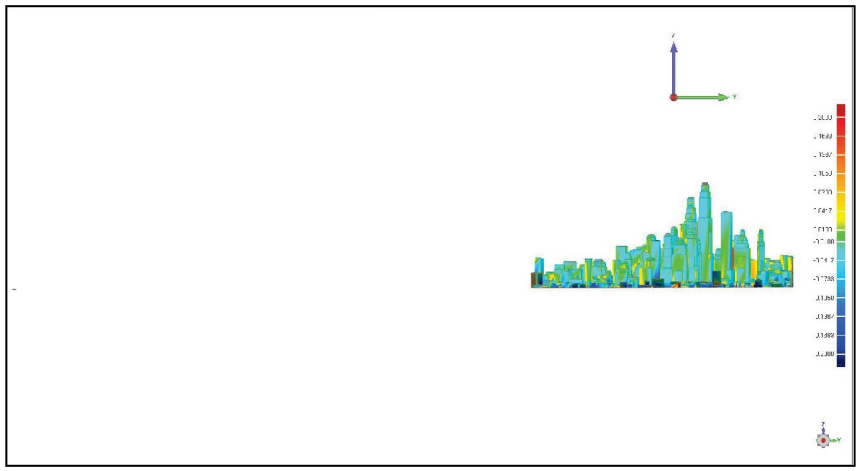


Predefined: Back

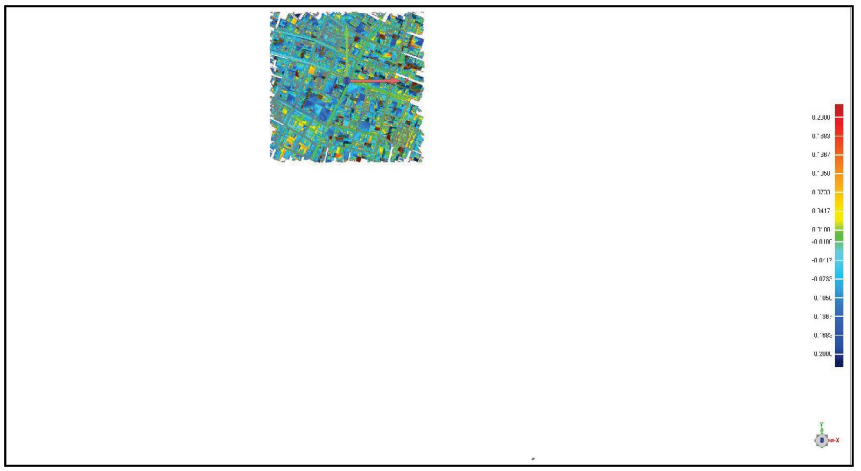


Predefined: Left

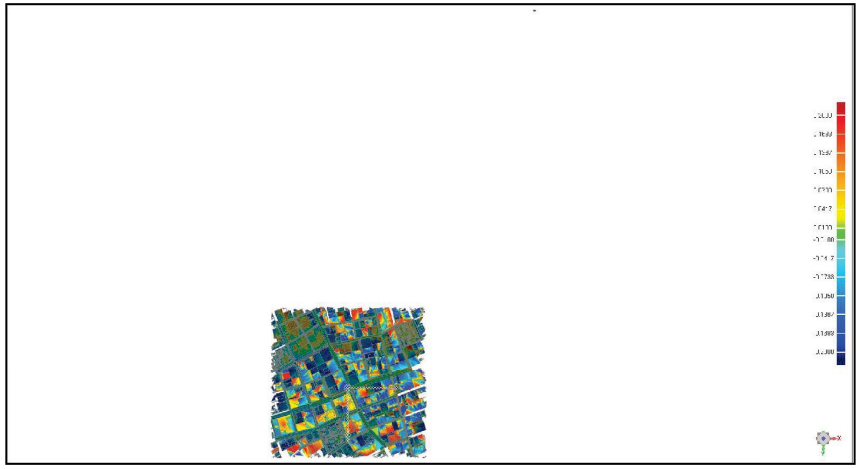
Predefined: Right



Predefined: Top



Predefined: Bottom



Location Set: Upper and Lower Deviations

Units: mm

Name	Dev	Status	Upper Tol	Lower Tol	Ref X	Ref Y	Ref Z	Radius	Dev X	Dev Y	Dev Z	Measured X	Measured Y	Measured Z	Proj. Dir X	Proj. Dir Y	Proj. Dir Z
Lower Dev.	-0.2000				3.0461	0.5081	-8.3945	n/a	0.0008	-0.0003	0.2000	3.0469	0.5078	-8.1945	-0.0038	0.0015	-1.0000
Upper Dev.	0.2000				-4.1346	4.1945	-7.9247	n/a	-0.0646	-0.1320	0.1356	-4.1992	4.0625	-7.7891	-0.3231	-0.6601	0.6782