Hochschule Neubrandenburg
University of Applied Sciences

Fachbereich Landschaftsarchitektur, Geoinformatik,
Geodäsie und Bauingenieurwesen

Freie Software zur Herstellung semiautomatischer 3D Stadtmodelle

---

Masterarbeit

Tomasz Kogut

Master-Studiengang Geoinformatik und Geodäsie

Betreuer:
Prof. Dr.-Ing. Wolfgang Kresse
Prof. Dr.-Ing. Andreas Wehrenpfennig

urn:nbn:de:gbv:519-thesis 2011-0063-4

2011

Danksagung

An dieser Stelle möchte ich mich bei Herrn Prof. Dr.- Ing. Wolfgang Kresse und Herrn Prof. Dr.-Ing. Andreas Wehrenpfennig für die Betreuung meiner Masterarbeit bedanken.

Des Weiteren möchte ich mich bei allen bedanken, die mich bei der Erstellung meiner Masterarbeit unterstützt haben.

Ich möchte diese Masterarbeit meinen Eltern und Freunden widmen.

Kurzfassung

Diese Masterarbeit beschreibt die Möglichkeiten der (semi-) automatischen Modellierung eines Gebäudes aus Urlaubsfotos. Die Arbeit ist grundlegend in einen theoretischen und praktischen Teil gegliedert. In Kapitel 2 wurden die Ziele der Arbeit dargestellt. Im nächsten Kapitel wurden theoretischen Grundlagen, Standards und Anwendungen für 3D Stadtmodelle beschrieben. In Kapitel 4 wurde der Stand der Technik vorgestellt. Im 5. Kapitel wurde die verwendete Software mit ihren theoretischen Ideen für die Erstellung eines virtuellen Modells im dreidimensionalen Raum näher erläutert. Mit dem 6. Kapitel schließt sich der praktische Teil dieser Masterarbeit an. Dort werden der Ablauf des Algorithmus für die Herstellung eines automatischen 3D Stadtmodells und Beispiele der Ergebnisse dargestellt. In Kapitel 7 „Diskussion" wurden Probleme, Quellen und Anwendungen für die vorgestellten Verfahren beschrieben. Abschließend wurde in Kapitel 8 eine Zusammenfassung gegeben.

Abstract

This thesis describes the possibilities of the (semi-) automatic creation of virtual buildings from holiday photos. The work is divided mainly into theoretical and practical parts. In chapter two, the target of the work is presented. In the next chapter there are included descriptions of theoretical fundamental standards and applications for 3D city models. In Chapter four is the state of the art presented. Chapter five presents the software that is used with its theoretical ideas to create virtual model in the three- dimensional space. However chapter number six begins with a description of the practical part of the mentioned thesis, where both the flow of the algorithm to produce an automatic 3D city models and examples of results were shown. In chapter 7 named "Talk" then are presented the problems, sources and applications of method. Chapter 8 consists of conclusions.

Inhaltsverzeichnis

1. Einleitung

Die Bedeutung von 3D Stadtmodellen nimmt in der heutigen Zeit sehr schnell zu. Sehr viele „benutzerfreundliche Firmen“ wie beispielweise Google oder Microsoft erstellen 3D Stadtmodelle für die freie Benutzung. Diese sind aber oft nicht genau genug oder verzerrt. Gleichzeitig werden neue Anwendungen von 3D Modellen in verschiedenen Bereichen erzeugt. Man kann also sagen, dass der Trend für das Vorhandensein virtueller Städte stetig steigt, aber die Herstellung der heutigen Modelle sehr kostenintensiv ist, weil sie fast nur durch die Auswertung von Luftbildern und Laserdaten möglich ist. Es ist wünschenswert, ein neues Verfahren zur Minimierung der Kosten zu suchen. Ziel dieser Masterarbeit ist nicht nur die Herstellung eines (semi-) automatischen virtuellen Stadtmodells im dreidimensionalen Raum, sondern auch das Aufzeigen, wie kostengünstig ein 3D Modell erstellt werden kann, welche Daten hierzu benötigt werden und wie die Daten zu bearbeiten sind. In den folgenden Kapiteln wurden nicht nur theoretische Grundlagen zur Erzeugung eines (semi-) automatischen 3D Stadtmodell beschrieben, sondern auch „ein Algorithmus” entwickelt und ein Beispiel erstellt, wie Urlaubfotos als Quelle zur Erstellung eines virtuellen Modells benutzt werden können.

2. Ziel der Arbeit

Ziel der Arbeit ist:

- Die Ausnutzung ungeordneter Urlaubsfotos um eine Punktwolke herzustellen.
- Die Untersuchung der Datenquellen, um eine Punktwolke zu erzeugen.
- Die Suche der Daten zum virtuellen Modell, welche die Kosten minimieren können.
- Die Entwicklung eines Verfahrens zur Errichtung eines (semi-) automatischen 3D Stadtmodells.

3. Grundlagen

Der folgende Abschnitt befasst sich mit allgemeinen Informationen über 3D Stadtmodelle, welche als Grundlage für die beschreibenden Verfahren in der Arbeit benutzt werden.

3.1. Was ist ein 3D Stadtmodell?

Unter einem 3D Stadtmodell ist laut Lorber G. „ein dreidimensionales Computermodell einer Stadt zu verstehen. In diesem Modell sollen die realen Objekte der Stadt möglichst realitätsnah abgebildet werden. Zu jedem Objekt werden die Informationen gespeichert, die für eine räumliche Rekonstruktion erforderlich sind.“ [Lorber 1996].
Diese Definition stammt aus dem vorherigen Jahrzehnt und beschreibt nur die Grundvoraussetzungen an ein 3D Stadtmodell. Heutzutage beinhaltet es mehr als nur räumliche Informationen zur Rekonstruktion. Moderne Modelle enthalten semantisch-geometrische Daten, mit denen es ermöglicht wird, immer mehr Aufgabenbereiche abzudecken. Ein 3D Stadtmodell wird in Zukunft nicht nur zur Visualisierung von Städten dienen – immer mehr ingenieurtechnische Anwendungen werden auf Grundlage eines 3D Modells arbeiten. Die Daten können z.B. in dem modernen Datenaustauschformat CityGML beschrieben werden.

3.1.1. CityGML als Standard für heutige 3D Stadtmodelle

Das neuste und auch am besten geeignete Modell zur Beschreibung von 3D Stadtmodellen ist CityGML. Es wurde als Standardformat von einem Konsortium GDI NRW (Geodata Infrastructure North Rhine-Westphalia), in dem mehr als 100 Institutionen beteiligt sind, seit 2002 entwickelt. Das Ziel der Gruppe ist eine universelle Lösung zur Verarbeitung, Visualisierung und zum Austausch von 3D Geodaten. CityGML beruht auf ISO (International Standardisierungsorganisation) Normen & OGC (Open Geospatial Consortium).Standards.

Thomas Kolbe schreibt, dass „unter CityGML ein geometrisch-semantisches Informationsmodell für 3D Stadtmodelle zu verstehen ist. Semantische 3D Stadtmodelle zeichnen sich dadurch aus, dass Informationen über den städtischen Raum in Form von klassifizierten Objekten mit räumlichen und nicht-räumlichen Eigenschaften strukturiert sind und damit auch die Bedeutung, Einordnung sowie physikalische und funktionale Eigenschaften repräsentiert werden. Auch die räumlichen und logischen Beziehungen zwischen Objekten werden ausdrücklich beschrieben.“ [KOLBE 2008].

Um die enormen Datenmengen eines 3D Stadtmodells besser verarbeiten zu können, integrierten die Entwickler das Konzept des Level of Detail (Detaillierungsgrade).

LoD sollen die Prozesse der unabhängigen Datenerfassung für die unterschiedlichen Anwendungen vorstellen. Darüber hinaus erleichtern LoD eine effiziente Visualisierung und Datenanalyse. In einem CityGML-Datensatz können die gleichen Objekte in verschiedenen Detaillierungsgraden gleichzeitig dargestellt werden, so dass die Analyse und Visualisierung des gleichen Objekts im Hinblick auf verschiedene Grade der Auflösung ermöglicht wird [13].

Es lassen sich folgende fünf LoD unterscheiden:

- LoD 0 ist eine 2,5D Ansicht des Geländes - DGM
- LoD 1- Das Modell wird als „Klötzchen-Modell“ bezeichnet. Die Gebäude werden als Quader ohne Dachformen dargestellt. Diese niedrigste Stufe des Detaillierungsgrades wird im Allgemeinen als ausreichend für die Darstellung in den Maßstäben 1:250.000 bis 1:10.000 angesehen.
- LoD 2: In diesem Modell werden den Gebäuden zusätzlich Dächer aus geometrischen Formen zugeordnet und die Fassaden können texturiert werden. Dieser mittlere LoD findet in den Maßstäben 1:10 000 bis 1:500 Anwendung.
- LoD 3: In diesem Modell werden die Gebäude realitätsnah erzeugt. Zusätzlich werden die Straßenmöblierung und die erweiterte Vegetation dargestellt. Diese Objekte werden mit einer Phototextur versehen. Aufgrund der großen Datenmengen ist hier nur eine projektbezogene Realisierung zu verwirklichen. Das Modell kann für Maßstäbe von 1:1000 bis 1:50 benutzt werden.
- LoD 4 ist ein begehbares Innenraummodell.

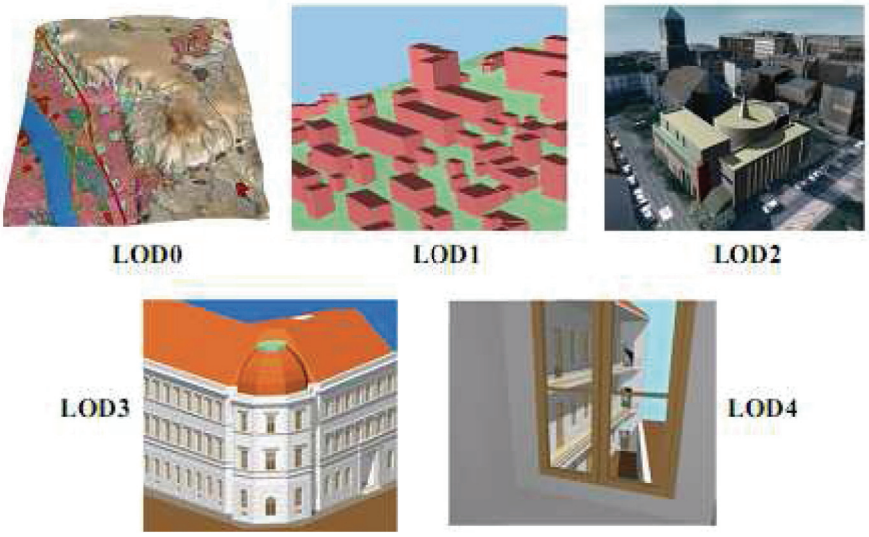


Abbildung 1. Detaillierungsgrade (Quelle: OGC, OpenGIS City Geography Markup Language (CityGML) Encoding Standard)

3.2. Anwendungen von 3D Stadtmodelle

Es gibt heutzutage bereits viele Anwendungsgebiete für 3D Stadtmodelle und je genauer die Modelle werden, desto höher steigt die Zahl möglicher neuer Anwendungen, die sich daraus ergeben. In dieser Arbeit werden die wichtigsten Anwendungen beschrieben.

Die erste Anwendung von 3D Stadtmodellen findet sich im Bereich des Katastrophenmanagements. Eine interessante Simulation ist ein Hochwasserereignis. Dabei werden wasserbautechnische Simulationsalgorithmen angewandt, um den genauen Verlauf des Hochwassers zu berechnen. Daraus lassen sich wirksame Schutzmassnahmen herleiten, um Schäden in Grenzen zu halten. Auch Evakuierungsmaßnahmen lassen sich durch solche Simulationen so planen, dass man für alle Fälle vorbereitet ist. Evakuierungspläne für Feuerkatastrophen und Erdbeben können ebenfalls aus Simulationen mit 3D Stadtmodellen entworfen werden.

Eine nächste Anwendung von 3D Stadtmodellen ist der Lärmschutz. Seit einiger Zeit verpflichtet die EU Gemeinden dazu, Bereiche zu lokalisieren, die besonders vom Lärm betroffen sind und Maßnahmen zur Lärmminderung zu ergreifen. Diese Bereiche werden durch Lärmausbreitungsberechnungen festgestellt. Unterstützend dazu werden Messungen vor Ort durchgeführt, um die Berechnungen zu prüfen und zu verfeinern. Zukünftig wäre eine vollautomatische Berechnung anhand genauer 3D Stadtmodelle möglich und würde aufwendige Messungen vor Ort unnötig machen.

Die dritte Anwendung von 3D Stadtmodellen ist im Bereich der Telekommunikation zu finden. Telekommunikationsunternehmen haben große Interessen an 3D Stadtmodellen. Diese nutzen sie, um die Funkausbreitung zu berechnen. Aus diesen Daten können nun effektiv Standorte neuer Mobilfunkantennen hergeleitet werden. Gebäude reflektieren Funkwellen, sind jedoch auch Hindernisse. Um eine möglichst flächendeckende Ausbreitung zu gewährleisten, sind solche Berechnungen unverzichtbar. Messungen vor Ort in jeder Stadt wären viel zu kostenintensiv.

Es gibt noch weitere Anwendungengebiete von 3D Stadtmodellen wie beispielsweise Navigationssysteme, die heute in fast jedem Auto genutzt werden, oder Tourismus. Auf diese wird jedoch nicht weiter eingegangen. Folgend werden stichpunktartig andere Anwendungsbereiche dargestellt:

- Immobilenmarkt,
- Stadt- und Regionalplanung,
- Architektur,
- Verkehr,
- Bauwesen,
- Planung von Solaranlagen,
- Training-Simulatoren (z.B. für die Ausbildung von Polizisten, Bundeswehr).
- uzw.

4. Stand der Technik

Geographische Informationssysteme spielen eine wichtige Rolle in vielen Bereichen unseres Lebens. Der hohe Prozentsatz von Informationen mit Raumbezug verlangt nach effizientem Datenmanagement und hat GIS, als Werkzeug hierfür, in vielen Bereichen zum Standard werden lassen. Am Beginn dieser Entwicklung standen zunächst Abbildungen der Realität in Datenform und Methoden zu deren Verwaltung. Die großen Anforderungen an räumliche Daten wurden durch den Wunsch nach geeigneten Präsentationsmöglichkeiten verursacht. Die Funktionalitäten EVAP - Eingabe, Verwaltung, Analyse, Präsentation sind von Anfang an Bestandteil der Definitionen von GIS [27]. Für die Verbindung der Entwicklung von GIS mit der Rechentechnik waren Datenverwaltung und Präsentationsmöglichkeiten lange Zeit limitiert. Je nach Stand der Technik, wurde nach geeigneten Möglichkeiten der Datenausgabe gesucht.

Es gibt sehr viele Einsatzgebiete und somit auch viele unterschiedliche Anforderungen an ein 3D Stadtmodell. Beispielsweise können sie zur Visualisierung verschiedener wichtiger Informationen, wie die Simulation der Schallausbreitung, benutzt werden. Also muss die Verwaltung der Daten sehr flexibel sein. Die Sammlung der Daten ist oft sehr teuer und personalaufwendig. Die Benutzung bereits vorhandener Daten ist dabei sehr hilfreich, insbesondere wenn neue Informationen durch Kombination mehrerer Datenquellen automatisch erzeugt werden können. Leider sind die Qualität und die Art der Quellen von Stadt zu Stadt unterschiedlich und manchmal kaum für eine automatische Verarbeitung zu gebrauchen [28]. Die Vielzahl der möglichen Anwendungen macht ein Modell sehr attraktiv für Städte, Firmen und andere Benutzer.

Die Erfassung und Fortführung von 3D Stadtmodellen sollte kostengünstig, flächendeckend sowie wiederholbar und vergleichbar durchzuführen sein. Die für die Stadtplanung erforderlichen Maßstäbe können durch Stereoauswertung von Luftbildern und terrestrischem oder Airborne Laser Scanning erreicht werden. Die hohen Kosten einer 3D Erfassung und der dabei zu verarbeitenden enormen Datenmenge einerseits und der Forderung nach Aktualität der Raumdaten und ihrer kontinuierlichen Fortführung für Monitoringmaßnahmen andererseits, macht eine Automatisierung der Auswertungsverfahren von Luftbilddaten und Laserdaten erforderlich.

In den nachfolgenden Abschnitten dieses Kapitels werden automatische und semiautomatische Verfahren sowie Systeme zur Herstellung von 3D Stadtmodellen, die heute

stetig weiterentwickelt werden und hauptsächlich auf der Stereoauswertung von Luftbildern und Laser Scanning basieren, dargestellt.

Das erste vorgestellte Verfahren wurde von Heiko Hirschmüller entwickelt. Der Algorithmus heißt Semi-Global Matching (SGM). Er basiert auf der Idee der pixelweisen Transinformation (Mutual Information) und approximiert die Minimierung einer globalen Kostenfunktion [35]. Die Transinformation bzw. Mutual Information ist eine Größe aus der Informationstheorie, welche die Stärke des statistischen Zusammenhangs zweier Zufallsgrößen angibt.[38] Die Transinformation wird auch als Entropie bezeichnet [36].Die Methode besteht aus Stufen matching cost calculation, cost aggregation, disparity computation / optimization und disparity refinement [36]. Matchingkosten werden pixelweise horizontal, vertikal und diagonal akkumuliert und mit einem kantenerhaltendem Glattheitskriterium verknüpft. Das Glattheitskriterium wird eingesetzt, da die Vergleichskosten oft nicht eindeutig sind und es dabei zu falschen Korrespondenzen bei niedrigere Vergleichskosten kommen kann [37].Durch das Glattheitskriterium werden alle Pixel des Bildes miteinander verknüpft. Die Minimierung erfolgt sehr effizient mit der gleichen Komplexität die typisch ist für schnelle korrelationsbasierte Ansätze (linear zur Anzahl der Pixel und Disparitätsstufen). Zur Berechnung der pixelweisen Matchingkosten wird die Transinformation eingesetzt, wodurch SGM sehr robust gegenüber großer radiomentrischer Differenzen und auch kleinen Störungen wie z.B. Spiegelungen ist [37]. Die Transinformation ergibt sich aus der Summe der Entropien abzüglich der Größen ihrer Vereinigungsentropie. Das Glattheitskriterium wird zunächst entlang der Richtung unabhängiger eindimensionaler Pfade, durch Definition der Pfadkosten, rekursiv formuliert. Um eine quasi-globale Abhängigkeit über das gesamte Bild zu erreichen, werden die Pfadkosten aus mehreren Richtungen zu jedem Pixel berechnet. Anschließend werden die Summenkosten berechnet. Anzahl und Lage der Pfade beeinflussen die Qualität der Disparitätskarte und müssen je nach Anwendung angepasst werden.

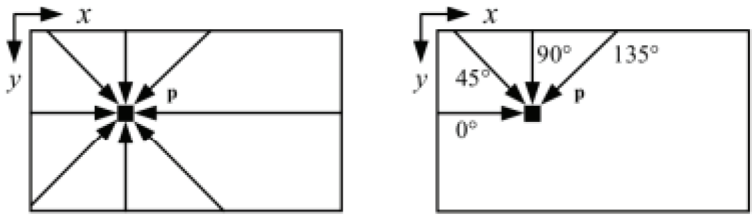


Abbildung 2. Mögliche Lage von Pfadrichtungen für acht und vier Pfade.

Mit diesem Verfahren wurden automatische Rekonstruktionen der Luftbilder von Berlin aus dem Jahre 1850 gemacht. Jedes Bild hatte 86 MPixel bei einer Auflösung von 7cm/Pixel. Folgende Abbildungen zeigen die Ergebnisse der automatischen Rekonstrukion.

Abbildung 3. Rekonstuierte 3D Berlin (Quelle: http://www.robotic.dlr.de/Heiko.Hirschmueller)

Das zweite vorgestellte System wurde an der ETH Zürich entwickelt. Das System heißt AMOBE (Automation of Digital Terrain Model Generation and Man-Made Object Extraction from Aerial Images) und basiert auf einem verallgemeinerten Polyedermodell sowie einer äußerst leistungsfähigen Bildauswertung [26]. In den zugrundeliegenden Luftbildern werden photometrische und chromatische Attribute der Projektionen von Gebäudekanten in der Merkmalsextraktion durch die Analyse von lokalen Nachbarschaftsbereichen (Flanking Regions) abgeleitet. Aus den extrahierten Bildlinien werden räumliche Kantenbeschreibungen durch eine Stereoanalyse generiert, in deren Korrespondenzanalyse neben der epipolaren Geometrie auch die photometrischen und chromatischen Attribute der Bildlinien eingesetzt werden. Die Gruppierung zu polygonalen Beschreibungen von Dachflächensegmenten erfolgt sowohl in den Luftbildvorlagen als auch im rekonstruierten Modellraum, indem solche Linienzyklen generiert werden, deren Bildlinien neben geometrischer Nähe und Verträglichkeit eine hinreichende Homogenität in ihren photometrischen und chromatischen Attributen aufweisen und deren korrespondierende räumliche Kantenrekonstruktionen eine koplanare Kantengruppe bilden.

Alle derart einer objektzentrierten Ebene zugeordneten Linienzyklen werden nach Kriterien der Einfachheit und Kompaktheit der Flächenform sowie der 3D Unterstützung durch die generierten Kanten bewertet. Die bestbewerteten, in die objektzentrierten Ebenen projizierten Linienzyklen definieren die polygonalen Dachflächensegmente, aus denen zusammenhängende Dachhypothesen generiert werden.

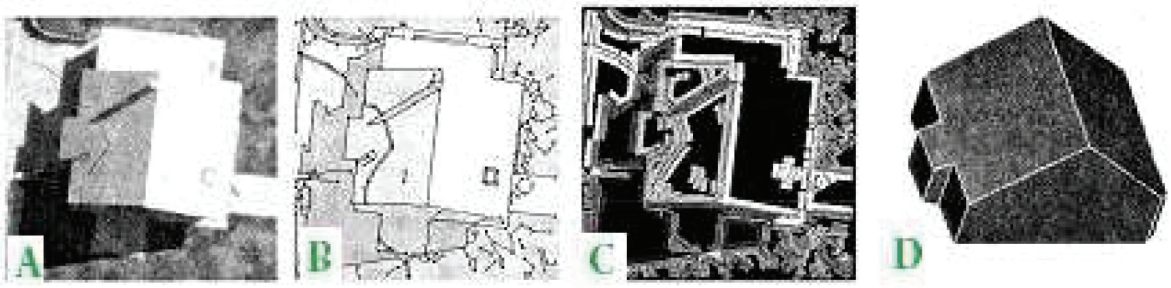


Abbildung 4. A) Luftbild, B) extrahierte Kanten, C) die lokalen Nachbarschaftsbereiche, D) das rekonstruierte 3D -Gebäude. (Quelle: [34])

Das nächste semiautomatische System wurde an der Katholische Universität Leuven fortgeführt. Das System basiert in der perzeptiven Gruppierung auf einer Triangulation der Bildvorlagen, dass jede extrahierte Bildlinie einer Kante der Triangulation entspricht. Innerhalb dieser Triangulation wird ein Prozess des Regionenwachstums eingesetzt, um benachbarte Zellen mit ähnlichen photometrischen und chromatischen Attributen zusammenzufassen [29]. Die abgeleiteten homogenen Bildregionen werden als betrachterzentrierte Dachflächenhypothesen interpretiert und bilden die Grundlage der Rekonstruktion von objektzentrierten Kantenbeschreibungen. Die Gruppierung der rekonstruierten Kanten zu objektzentrierten Beschreibungen von polygonalen Dachflächensegmenten basiert einerseits auf ihrer Zugehörigkeit zu den aus der Bildtriangulation abgeleiteten betrachterzentrierten Dachflächenhypothesen und andererseits auf der geometrischen Zugehörigkeit zu einer koplanaren Ebene [29]. Die Triangulation kann interaktiv gesteuert werden, so dass selbst in einer dichten Bebauung mit hohem Verdeckungs- und Verschattungsgrad sowie großem Detailreichtum zuverlässige Erfassungen ermöglicht werden. Die folgende Abbildung zeigt die Zwischenschritte und Ergebnisse des Systems.

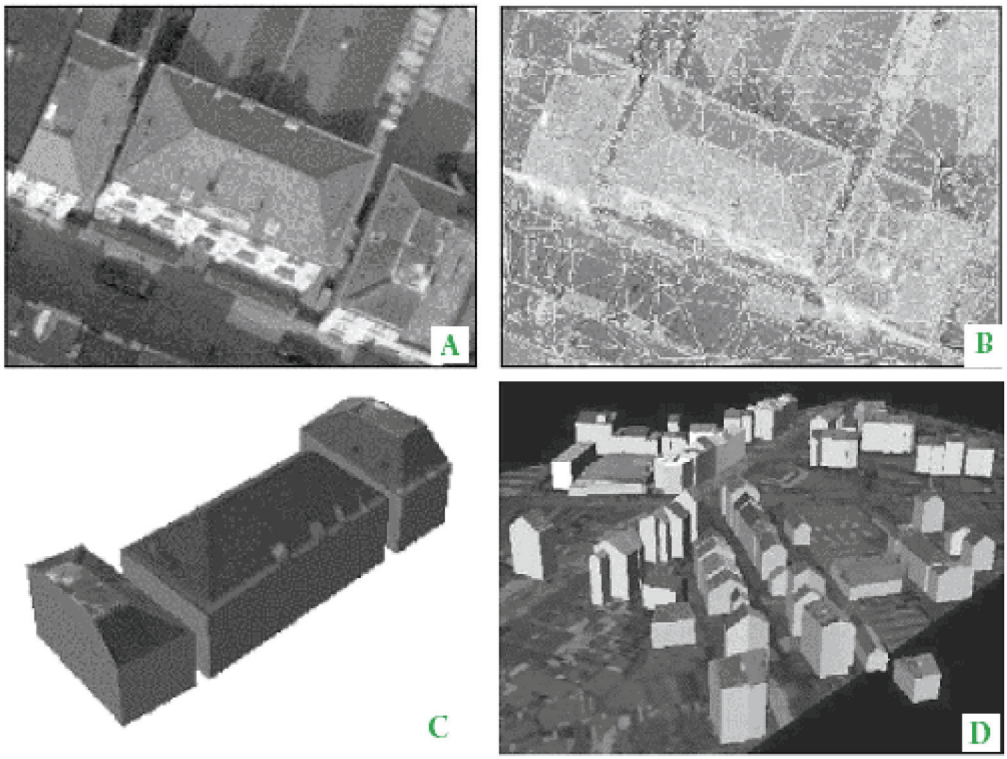


Abbildung 5. Das semiautomatische System der KU Leuven. A) Bildausschnitt, B) Triangulation, C) Rekonstruktion, D) Gesamtrekonstruktion.

Die GTA Geoinformatik GmbH mit Hauptsitz in Neubrandenburg (M-V) wird seit ihrer Gründung im Jahre 1991 von Peter Lieckfeldt geführt. Das Geoinformatik-Unternehmen gehört zu den Technologieführern in digitaler Photogrammetrie, Luftbildauswertung und der Erstellung von 3D Stadtmodellen [30]. Die GTA Geoinformatik GmbH bietet eine Software an, die es ermöglicht, in einem automatisierten Prozess 3D Stadtmodelle herzustellen. Diese Software heißt „tridicon 3D“ und liefert qualitativ hochwertige Modelle aus Stereoluftbildern, aber auch anderen geeigneten Quellendaten wie LiDAR Daten (Airborne Laserscanning) sowie Stereo-Satellitenbilder. Außerdem werden ein Digitales Geländemodell und Gebäudegrundrisse verwendet. „Mit der Software tridicon 3D wurden von GTA 2010 bereits mehr als 180 europäische 3D Stadtmodelle im Level of Detail 2 (LOD 2) mit einer Gesamtfläche von mehr als 5.000 km² erstellt“ [31]. Tridicon 3D ermöglicht die vollautomatische Texturierung aus Luftbildern sowie die Solarpotenzialanalyse. Die Software hat ein Zusatzmodul zur Verfügung, mit dem vielseitige Visualisierungen und Analysen der 3D Stadtmodelle durchgeführt werden können. Das dynamische Nachladen von Geländen und Gebäuden erlaubt die Darstellung beliebig großer Modelle in Echtzeit [31]. Die erzeugten 3D Stadtmodelle können in alle gängigen Datenformate (u.a. CityGML, 3D Shape, KML, OBJ, 3DS, VRML) gespeichert werden.

Abbildung 6. 3D Stadtmodell München LOD 2, photorealistisch texturiert (Quellen: GTA)

Die nächste Methode kommt von der Firma C3 Technologies AB aus Schweden. Alle C3-Produkte basieren auf Bildern aus kalibrierten Kameras. Die Kameraparameter (die Positionen und Winkel) werden für jedes Bild mit unter Nutzung eines Navigationssystems berechnet [32]. Die virtuellen 3D Stadtmodelle werden direkt aus Luftbildern berechnet und sind nicht abhängig von LIDAR-Daten. Die Straßenbilder werden unter Nutzung eines fortschrittlichen mehr-Kamera-Systems mit überlappenden Blickwinkeln für eine Stereolösung gemacht. Die Vorteile dieser Software sind die vollständige automatische Prozessierung von Bildern, die schnelle Herstellung von 3D Stadtmodellen für große Städte, der gleiche Detailierungsgrad für jeden Stadtteil und die Konzentration auf realistische Details, Farbe und geometrische Qualität.

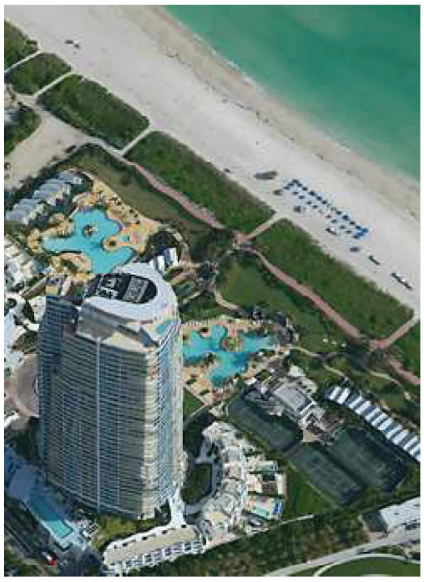


Abbildung 7. Miami Starand in 3D von C3 (Quelle: http://www.c3technologies.com)

Das nächste Verfahren ist Laser Scanning mit einer Software für die weitere Bearbeitung einer Punktwolke. Sehr vielversprechend ist ein Laser Scanner von Faro Focus 3D. Der 3D Laser Scanner ist sehr effizient durch die große Reichweite von 120 m. Die Punkte werden in ein paar millimetergenaues virtuelles Abbild der Realität mit einer Geschwindigkeit von knapp 1 000 000 Messpunkten pro Sekunde geschafft [33].

Abbildung 8. 3D Laser Scanner - Faro Focus 3D (Quelle: http://www.faro.com)

Die Punkte können weiter beispielsweise mit einer Software Geomagic Studio bearbeitet werden. Diese Software kann automatisch geometrische Grundkörper wie Kugel, Ebene, Kegel, Zylinder, Langloch, Bohrung und Punkt erkennen. Die Erkennung dieser geometrischen Grundkörpes befähigt die Software auch zur Erkennung der Grundbausteine von 3D Stadtmodellen. Geomagic Studio arbeitet mit fast allen bekannten Dateiformaten. Dieses Verfahren ist sehr effizient und sehr genau, hat aber leider einen Nachteil – den Preis. Ein Gesamtpaket mit allen Kosten beträgt ca. 75000 Euro.

5. Software für die semiautomatische Erstellung eines 3D Stadtmodells

Das folgende Kapitel befasst sich mit den theoretischen Grundlagen der Software, welche für die beschriebenen Verfahren in der Arbeit benutzt werden.

5.1. Bundler

Bundler ist ein Struktur-from-Motion-System (Sfm) für eine Reihe ungeordneter Bilder, die man z.B. im Intenet finden kann. Bilder, Bildeigenschaften und Kameraparameter werden als Eingangsdaten des Programms benutzt. Bundler erzeugt eine 3D Rekonstruktion und gibt diese als Geometrie aus. Derzeit wurde Bundler hauptsächlich unter Linux zusammengestellt und getestet. Es gibt aber auch eine Windows-Version für Cygwin.
In der binären Distribution von Bundler sind ein Skript für einfache Benutzung, eine Version der ANN- (aproximate nearest neighbor) Bibliothek [4], und ein Hilfsprogramm (Bundle2PMVS) zum Konvertieren der Bundlerdatei (.out) in die Eingangsdaten für PMVS Multi-View-Stereo-System von Yasutaka Furukawa's enthalten. Diese Distribution enthält auch ein Programm (RadialUndistort) zur Erzeugung nicht verzerrter Bilder.

Abbildung 9. Bundler [Quelle:http://phototour.cs.washington.edu/bundler/images/Colosseum.jpg]

Die Rekonstruktion einer 3D Struktur von gesammelten Bildern hat in der Regel drei grundlegende Schritte. Der Prozess beginnt mit einem Satz von Eingangs-Bildern ohne Kenntnisse über 3D Form der Szene oder des Kamerastandorts [17]. In Abbildung 3 werden drei Bilder eines Würfels dargestellt. Als Erstes werden die markanten 2D-Zeichen in den Bildern erkannt, die als schwarze Punkte auf den folgenden Abbildungen gezeigt werden. Dann werden die zueinander passenden Merkmale zwischen den Bildern, in der Abbildung durch Farbcodierung dargestellt, gesucht. Schließlich werden die 3D Geometrie von Kameras und die Punkte in einer Prozedur Structure for motion geschätzt. Das Sfm-System optimiert

die gewonnene 3D Struktur für eine selbständige Konsistenz. Für diese Berechnungen werden die Programme benötigt, die in den folgenden Unterkapiteln von 4.1.1. bis 4.1.3. beschrieben werden.

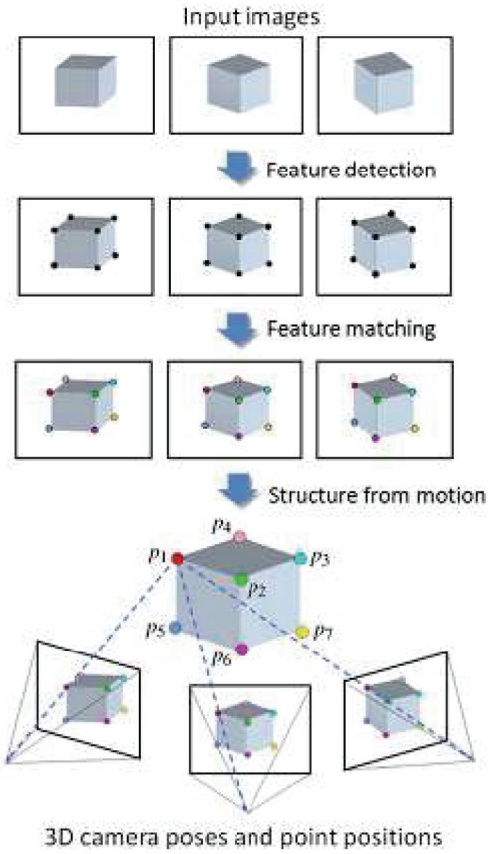


Abbildung 10. Szenenrekonstruktionschema (Quelle: Scene Reconstruction and Visualization From Community Photo Collections)

5.1.1. ImageMagick

ImageMagick ist ein freies Softwarepaket zur Erstellung und Bearbeitung von Rastergrafiken. Es kann mehr als 100 der üblichen Bildformate lesen, konvertieren und schreiben. Außerdem lassen sich Bilder dynamisch generieren, weshalb es auch im Bereich der Webanwendungen verwendet wird [6].

Die Funktionalität von ImageMagick wird in der Regel über die Kommandozeile oder über Funktionen der nachfolgend aufgezählten Programmierungssprachen verwendet. In ImageMagick können diese Schnittstellen benutzt werden [8]:

- G2F (Ada),
- MagickCore (C),
- MagickWand (C),
- ChMagick (Ch),

- ImageMagickObject (COM +),
- Magick + + (C + +),
- JMagick (Java),
- L-Magick (Lisp),
- NMagick (Neko / haXe),
- MagickNet (. NET),
- PascalMagick (Pascal),
- PerlMagick (Perl),
- MagickWand für PHP (PHP),
- imagick (PHP),
- PythonMagick (Python),
- RMagick (Ruby),
- oder tclmagick (Tcl / TK).

Abbildung 11. Logo von Imagemagick (Quelle: http://www.imagemagick.org/image/logo.jpg)

5.1.2. SIFT

SIFT (Scale-invariant feature transform, „maßstabsunabhängige Merkmalstransformation“) „ist ein Algorithmus zur Extraktion lokaler Bildmerkmale aus Abbildungen. Er kann vor allem bei der Bilderkennung verwendet werden. Er wurde von David G. Lowe an der University of British Columbia im Jahre 1999 veröffentlicht.“ [5]

Image Matching ist ein „sehr heißes” Thema und ein grundlegender Aspekt von vielen Problemen im Bereich Computer Vision, einschließlich Objekt- oder Szenenerkennung, Lösung für die 3D Struktur aus mehreren Bildern und Stereo-Korrespondenz sowie Bewegungskontrolle. Dieses Kapitel der vorliegenden Masterarbeit beschreibt Bildmerkmale, worunter eine Vielzahl von Eigenschaften zu verstehen ist, die für alle unterschiedlichen Bilder eines Objektes oder einer Szene zutreffen. Die Funktionen Skalierung, Rotation und teilweise Beleuchtungsvariation, Bildrauschen und geringere geometrische Deformation

höherer Ordnung werden automatisch angepasst. Eine große Anzahl dieser Funktionen können aus Bildern mit effizienten Algorithmen extrahiert werden.
Im Folgenden sind die wichtigsten Schritte der verwendeten Berechnung aufgezeigt, um die Bildmerkmale zu erzeugen [1]:

1. Scale-space extrema detection:

Die erste Stufe der Berechnung durchsucht alle Skalen und Kamerastandorte. Dies wird effizient mithilfe der Differenz zweier Gauß-Funktionen umgesetzt, um potenzielle Punkte, die unabhängig von Maßstab und Orientierung sind, zu identifizieren.

2. Keypoint localization (Schlüsselpunkt Lokalisation):

An jeder möglichen Position ist ein detailliertes Modell bezüglich Lage und Ausmaß zu bestimmen. Schlüsselpunkte werden auf der Grundlage ihrer Stabilität ausgewählt. Abbildung 5 beschreibt die Phasen der Auswahl der KeyPoints. Bild (a) zeigt das Originalbild. Auf (b) wurden 832 Schlüsselpunkte erkannt und die Maxima und Minima der Differenz-of-Gauß-Funktion dargestellt. (c) zeigt 729 KeyPoints nach der Benutzung des minimalen Kontrasts. Bild (d) zeigt die letzten 536 Schlüsselpunkte, die nach der Einführung einer zusätzlichen Schwelle bei der Bestimmung des Verhältnisses der Hauptkrümmungen noch vorhanden sind.

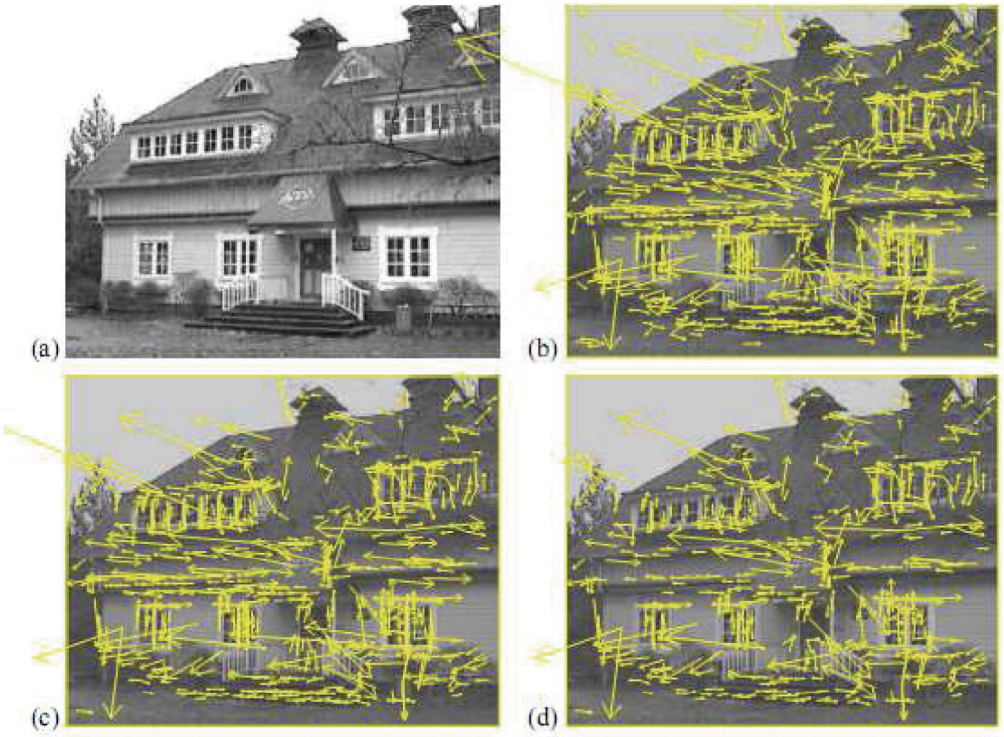


Abbildung 12. Die Phasen der Auswahl der KeyPoints (Quelle: Distinctive Image Features from Scale-Invariant Keypoints)

3. Orientation assignment (Orientierung Zuordnung):

Eine oder mehrere Orientierungen werden jedem Schlüsselpunkt basierend auf den Richtungen der lokalen Bildgradienten zugewiesen. Alle weiteren Operationen werden mit Bilddaten durchgeführt, welche relativ zur zugeordneten Orientierung, Skalierung und Position für jedes Objekt transformiert wurden.

4. Keypoint descriptor (Schlüsselpunkt Schlagwort):

Die lokalen Bildgradienten werden im gewählten Maßstab im Bereich um jeden Schlüsselpunkt gemessen. Diese sind in eine Darstellung umgewandelt, die signifikante Level für lokale Verzerrungen und Veränderungen der Beleuchtung erlaubt.

Ein „Keypoint descriptor" berechnet zunächst die Gradienten und die Orientierung an jeder Bildentnahmestelle in einem Bereich um die Schlüsselpunkte (Abbildung 6. auf der linken Seite). Diese werden durch ein Gauß-Fenster gewichtet, was durch den überlagerten Kreis repräsentiert wird. Diese Proben werden dann in orientierten Histogrammen, welche die Inhalte von 4x4 Teilbereichen zusammenfassen, dargestellt (Abbildung 6. auf der rechten Seite). Die Länge der Pfeile entspricht der Summe der Steigungen (Gradient) in die jeweilige Richtung der Einzelpfeile eines Bereichs.

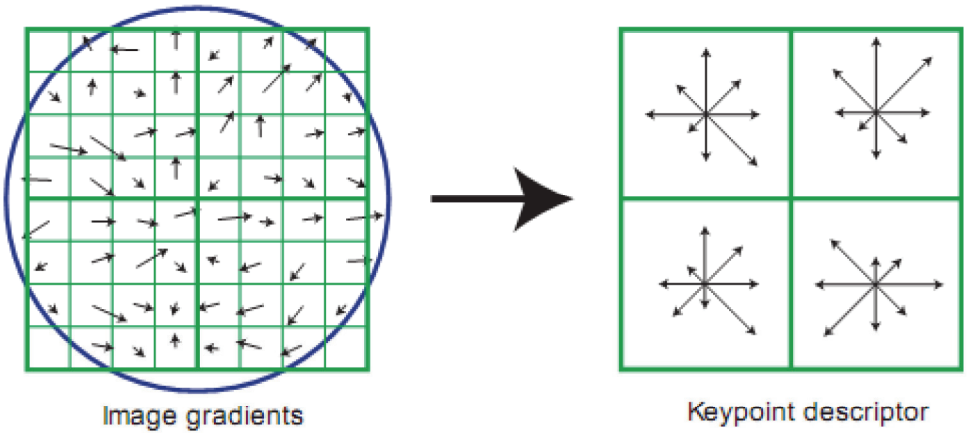


Abbildung 13. Diese Abbildung zeigt ein Berechne 2x2-Matrix-Deskriptor von einem 8x8 Matrix (Quelle: David G. Lowe, Distinctive Image Features from Scale-Invariant Keypoints)

Die SIFT-Kennzeichen werden in einer Gruppe von Referenz-Bildern (bereits bearbeitet) für Imageprozessing extrahiert und in einer Datenbank gespeichert. Jedes weitere Bild wird durch individuelles Vergleichen jedes Merkmals zu dieser Datenbank und durch das Finden passender Kennzeichen auf euklidische Distanz ihrer Merkmalsvektoren abgestimmt.

SIFT keypoint detektor ist ein Programm, das unter Linux oder Windows läuft. Die Demo-Software verwendet das PGM-Format als Eingangsdaten der Bilder. Die Keypoints

werden als Ausgangdaten in einem einfachen ASCII-Format gespeichert. Ein Matlab-Programm oder ein einfacher C-Code können die Schlüsselpunkte lesen und legen die Bilder über die Schlüsselpunkte übereinander.

```
33132 128
451.37 1311.17 138.51 1.384
 138 52 1 0 0 1 1 3 160 144 6 1 0 0 0 3 25 23 37 16
 0 0 0 1 0 0 2 1 0 0 0 0 142 43 1 0 0 9 2 10
 160 98 3 0 0 4 0 18 65 22 21 12 2 0 0 1 0 1 5 1
 0 0 0 0 156 80 2 0 0 4 3 18 160 69 14 1 0 0 0 53
 42 27 85 33 1 0 0 1 0 0 27 7 0 0 0 0 72 58 0 0
 6 17 2 2 141 128 8 0 0 0 0 3 7 20 53 8 0 0 0 0
 0 0 15 4 0 0 0 0
426.88 678.03 140.10 1.626
 75 6 0 0 0 0 0 0 20 158 3 0 0 0 0 0 76 125 1 0 0
 0 0 0 97 26 2 0 0 0 0 0 14 116 6 0 0 0 0 0 32
 158 15 0 0 0 0 0 0 36 158 18 18 1 0 0 0 70 14 9 15 0
 0 0 0 3 107 10 0 0 1 5 0 32 158 13 0 0 2 5 1 44
 151 31 29 3 0 0 0 10 0 3 13 1 0 0 0 0 114 23 0 0
 0 3 1 28 158 11 3 0 0 1 0 128 54 10 22 9 1 0 0 18
 0 2 6 0 0 0 0 0
726.05 1041.23 100.87 1.506
 81 6 0 0 0 0 0 0 20 114 18 1 10 28 8 1 4 163 60 0 1
 2 0 0 4 51 14 16 12 0 0 0 10 101 15 0 0 0 0 0 19
 121 12 0 4 53 32 5 15 163 83 1 1 10 5 0 8 64 27 17 8
 1 1 0 2 104 67 1 0 0 3 0 7 102 33 16 17 36 22 4 34
 163 9 1 3 9 5 1 104 66 5 3 8 9 3 0 14 73 24 0 0
 7 25 2 7 113 37 7 4 0 0 6 71 163 30 8 1 0 0 0 88
 16 16 38 15 1 0 0 1
724.22 383.57 89.66 1.617
 44 10 0 0 0 0 0 9 42 0 0 0 10 11 3 23 110 0 0 0
 2 2 3 60 35 0 0 0 0 0 4 36 132 6 0 0 0 0 0 32
 131 10 1 21 26 11 3 28 179 12 0 3 4 2 11 85 93 0 0 0
 0 0 25 93 119 3 0 0 0 0 0 28 138 10 2 26 35 0 0 6
 179 9 0 2 2 0 1 26 106 3 4 1 0 0 5 54 75 2 0 0
 1 2 1 30 106 3 0 8 38 19 3 11 179 22 0 0 2 0 0 5
 71 20 17 8 0 0 0 4
748.25 760.65 89.81 1.640
 75 5 0 0 0 0 0 0 21 93 6 2 35 46 3 0 11 174 13 0 10
 15 4 2 23 79 2 0 0 0 16 13 35 65 8 1 0 0 1 3 47
 99 7 4 33 57 26 5 19 174 9 0 5 11 3 0 30 94 10 8 10
 0 0 2 29 86 12 2 1 1 2 2 40 108 5 3 36 91 42 7 24
 174 12 0 4 13 4 0 15 70 19 18 11 1 0 0 4 83 40 6 3
 2 2 1 20 69 13 16 33 62 28 5 49 174 0 0 1 6 3 0 121
 30 1 0 2 10 5 1 14
```

Abbildung 14. Beispiele Ausgangdaten des SIFT Keypoint Detektor

5.1.3. ANN: A Library for Approximate Nearest Neighbor Searching

ANN ist eine in C++ geschriebene Bibliothek, die exakt und näherungsweise die Nachbarn in den verschiedenen Dimensionen sucht. Sie wurde von David Berg von der University of Maryland und Sunil Arya von der Hong Kong University of Science and Technology entwickelt. ANN bedeutet angenäherter nächster Nachbar (engl. Approximate Nearest Neighbor). ANN ist gleichzeitig ein Testprogramm zur Erzeugung von Datensätzen, zur Erfassung und statistischen Analyse der Leistung von nächste-Nachbarn Algorithmen, Datenstrukturen und zur Visualisierung der geometrischen Struktur dieser Daten [12].
Die Punkte werden in eine bestimmte Datenstruktur gebracht, so dass bei jeder Abfrage des Punktes q der nächste oder allgemein die k nächsten Punkte von P auf q effizient ermittelt werden können. Der Abstand zwischen zwei Punkten kann auf vielen verschiedenen Wegen

definiert werden. ANN setzt voraus, dass die gemessenen Abstände in einer Klasse der Entfernungen des Minkowski-Raums sind [2]. Dazu gehören die bekannten Algorithmen Euklidischer Abstand, Manhattan-Metrik und der maximale Abstand. David M. Mount schreibt, dass die Punktmengen effizient in einer Größe zwischen Tausend bis zu mehreren Hunderttausenden in mehr als 20 Dimensionen berechnet werden können.
Die Bibliothek implementiert verschiedene Datenstrukturen basierend auf k-d-Bäumen, box-decomposition Bäumen und ein paar verschiedenen Suchverfahren.

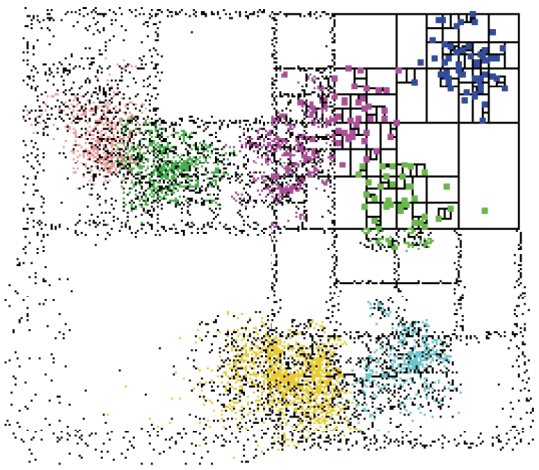


Abbildung 15. Beispiele ANN in k-d-Bäume und box-decomposition Bäume (Quellen: http://www.cs.umd.edu/~mount/ANN/)

5.2. PMVS (Patch-Based Multi-View Stereo)

PMVS nimmt eine Reihe von Bildern und Kamera-Parametern von Bundler, um so die 3D Struktur eines Objekts oder einer Szene von Bildern zu rekonstruieren. Es werden nur starre Strukturen rekonstruiert, d.h. nicht starre Objekte (wie z.B. Fußgänger vor einem Gebäude) werden durch die Software automatisch ignoriert und aus den Punkten eine Punktwolke erzeugt.

Der PMVS-Algorithmus versucht mindestens einen Patch [Abb. 10] in jeder Bildzelle $C_i(x, y)$ zu rekonstruieren. Er besteht aus drei Schritten: Feature Matching, Patch Expansion und Patch-Filterung [19]. Der Zweck der ursprünglich passenden Funktion (Feature Matching) ist es, eine „spärliche“ Datendichte für den Patch zu generieren. Die Erweiterung und die Filterung werden n-Mal durchlaufen, weil die Patches dichter und die fehlerhaften Übereinstimmungen entfernt werden.

Als erstes werden Blob und die Ecken der Objekte in jedem Bild mit Hilfe der Gaussian Differenz (DoG) und dem Harris-Operator erkannt. Man kann ein Bild (Ii) mit der

optischen Mitte O(Ii) der entsprechenden Kamera betrachten. Yasutaka Furukawa schreibt, dass für jede erkannte Funktion f in einem Bild (Ii), mit der Menge F von Funktionen f' des gleichen Typs (Harris oder DoG), aus den entsprechenden Epipolarlinien und Triangulation der 3D Punkte, die Punktpaare (f, f') bestimmt werden [19]. Weiterhin werden diese Punkte in der Reihenfolge ihrer zunehmenden Entfernung von O (Ii) als potenzielle Patch-Zentren zugeordnet und ein Patch versucht die Punkte nacheinander zu erzeugen. Dieses Verfahren wird viermal wiederholt.

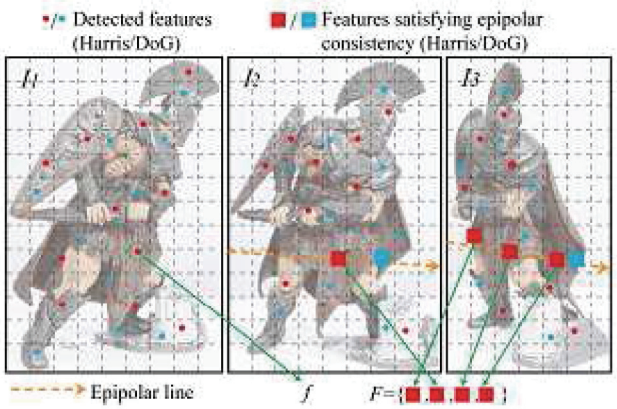


Abbildung 16. Beispiel zeigt die Anpassung der Funktion f'∈ F (erfüllt die Epipolarsbedingung in den Bildern I_2 und I_3) zu Funktion f in Bild I_1

Das Ziel des nächsten Schritts ist es, mindestens einen Patch für jede Bildzelle Ci (x,y) zu rekonstruieren. In dieser Phase werden iterativ die neuen Nachbarn zu bestehenden Patches hinzugefügt, bis die sichtbaren Oberflächen in der Szene abgedeckt sind. Man kann sagen, dass zwei Patches p und p' als Nachbarn gelten, wenn sie in benachbarten Zellen C (i, j) und C(i', j') im gleichen Bild (im potenziell sichtbaren Bild S(p)) liegen und ihr Tangentialraum dicht zusammen liegt [20]. Die neuen Nachbarn werden als Versuch nur bei Bedarf erstellt und der Abstand in der Tiefe vom Mittelpunkt c(p) und c(p') automatisch bestimmt. Dies entspricht einer Verschiebung um β1-Pixel im Referenzbild R(p). Dann wird der Patch p' durch die Zuordnung der entsprechenden Werte zum Referenzbild R(p') im (real) sichtbaren Bild T(p') wird der Normalenvektor n(p') initialisiert. Weiterhin werden (p') und n(p') optimiert.

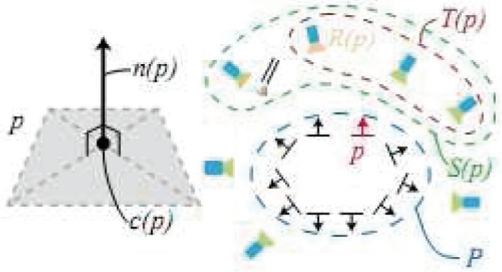


Abbildung 17. Definition des Patchs und seinen verbundenen Bildern (Quelle: Yasutaka Furukawa and Jean Ponce, Fellow, Juli 2007, Accurate, Dense, and Robust Multi-View Stereopsis)

In diesem Verfahren werden drei Filter verwendet, um fehlerhafte Patches zu entfernen. Der erste Filter beruht auf Konsistenz der Sichtbarkeit, das heißt, p und p' sind keine Nachbarn, aber liegen in der gleichen Bildzelle, wobei p sichtbar ist. Dann wird p als abweichender Wert gefiltert.

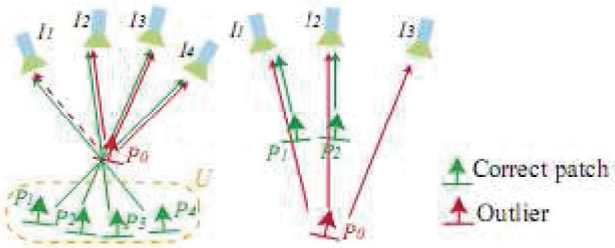


Abbildung 18. Idee des ersten Filters, bei dem die abweichenden Werte außerhalb (links) oder innerhalb (rechts) auf der richtigen Oberfläche liegen (Quelle: Yasutaka Furukawa and Jean Ponce, Fellow, Juli 2007, Accurate, Dense, and Robust Multi-View Stereopsis)

Der zweite Filter erzwingt auch die Konsistenz der Sichtbarkeit, aber auf strengere Art und Weise. Der dritte Filter basiert auf einer schwachen Form der Regularisierung. Für jeden Patch werden die Patches, die in allen Bildern in eigenen und benachbarten Zellen liegen, gesammelt. Wenn der Anteil der Patches, welche die Nachbarn von p in dieser Reihe sind, kleiner als 0.25 wird, wird p entfernt.

5.3. Poisson Surface Reconstruction (Poisson Oberflächenrekonstruktion)

Das folgende Kapitel befasst sich mit der Poisson Oberflächenrekonstruktion aus einer Punktwolke. Die Oberflächenrekonstruktion bezeichnet den Vorgang oder das Verfahren, aus einer beliebigen Punktmenge eine Oberfläche zu generieren, die diese Punktmenge approximiert [7]. Der Name Poisson-Rekonstruktion geht auf den französischen Mathematiker Siméon Denis Poisson zurück. Er definiert die Poisson-Gleichung

$(\nabla \bullet \vec{V} = \nabla \bullet \nabla \tilde{X})$, eine partielle Differentialgleichung zweiter Ordnung, die als Grundelement in dieses Verfahren eingeht.

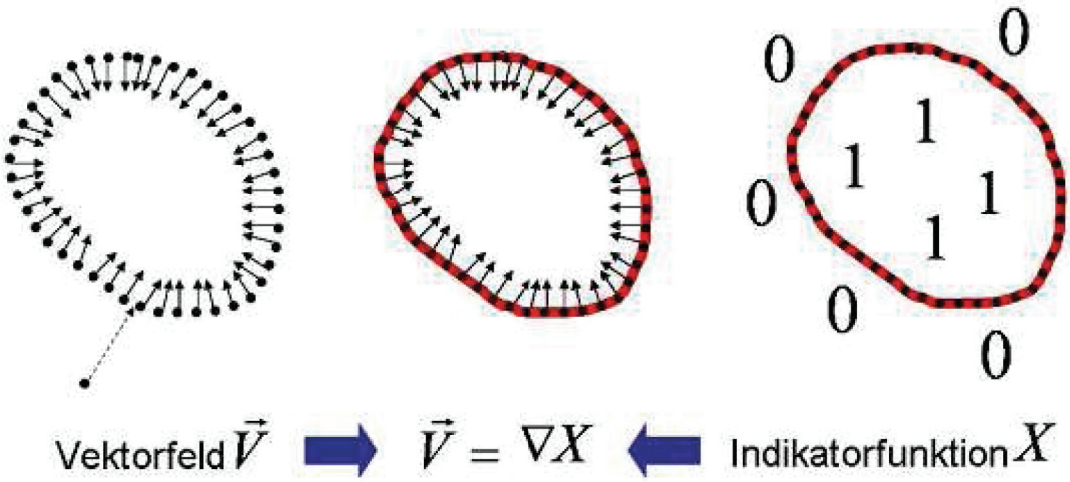


Abbildung 19. Idee der Poisson Oberflächenrekonstruktion

Die Idee bei der Poisson-Rekonstruktion ist in der Abbildung **[12]** dargestellt. Auf der linken Seite ist ein Vektorfeld $\vec{V}$ abgebildet. Dieses Vektorfeld ist so aufgebaut, dass alle Vektoren auf den Rand der gesuchten Oberfläche zeigen. Ein beliebiger Punkt $q \in R^3$ kann somit über $\vec{V}$ auf die gesuchte Oberfläche transformiert werden. Da für die Lösung aber eine implizite Funktion benötigt wird, reicht das Vektorfeld $\vec{V}$ allein nicht aus. Es bedarf eines weiteren Ansatzes, welcher auf der rechten Seite zu sehen ist. Man definiert eine Abstandsfunktion $f(q) = \chi(q)$, die für den Punkt $q \in R^3$ innerhalb der gesuchten Oberfläche den Wert 1 und außerhalb der gesuchten Oberfläche den Wert 0 zurück gibt. Solch eine Funktion wird Indikatorfunktion genannt. χ kann für alle Punkte q als Skalarfeld angesehen werden. Wendet man nun auf dieses Skalarfeld den Gradient-Operator an, so erhält man ein Vektorfeld, das in Richtung des stärksten Anstieges von χ zeigt. Das heißt, dass entstandene Vektorfeld zeigt dann auf die Oberflächenkante und entspricht genau dem Vektorfeld $\vec{V}$. Gelingt es, dieses Vektorfeld $\vec{V}$ nur auf Grund der gegebenen Punkte und Normalen aufzustellen, so kann über die eben beschriebene Beziehung eine Distanzfunktion definiert werden [14].

5.4. Textur

Das folgende Kapitel beschreibt das Programm Texture Stitcher, welches eine rekonstruierte Oberfläche texturiert. Diese Distribution implementiert einen Algorithmus (basierend auf dem Laplace-Beltrami Operator [15]) für die Verbindung der verschiedenen Scans als Textur auf dem Poissonmodell. Der Farbgradient wird über das Basisnetz definiert und dessen Divergenz berechnet. Ein Poissonmodell wird über das Farbfeld, welches am besten zum Farbgradienten passt, gelöst. Dieses Verfahren basiert auf der Lösung der Funktion f, welche minimiert werden soll [11]:

$$a\| f - g\|^2 + \|\nabla f - V\|^2$$

g ein Farbfeld extrahiert aus Scans

V der Vektor des aus den Scans erhalten Gradienten

a die Gewichtung

Diese Implementierung unterstützt die Spezifikation der Scans als Rang-Grid [9], was die Suche des Gradienten und der Netze mit Farbknoten erleichtert.

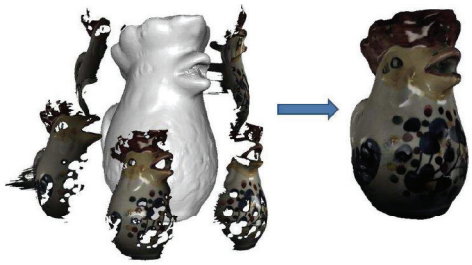


Abbildung 20. Idee des texturierten Objekts

6. Herstellung eines 3D Stadtmodells

Der folgende Abschnitt befasst sich mit der Herstellung eines 3D Stadtmodells und einem praktischen Beispiel, das zeigt, wie Urlaubsfotos zu einer Punktwolke extrahiert werden können und wie aus dieser Punktwolke das Modell hergestellt wird.

6.1. Bildkalibrierung

Die Daten für diese Masterarbeit kommen aus einer Bildsession. Die Bilder wurden mit einer NIKON COOLPIX L22 gemacht und im JPG- Format mit einer Bildgröße von 3.2 Megapixeln gespeichert.

Vor Verwendung von Bundler muss:

- ImageMagick installiert werden,
- die Konsole Cygwin unter Windows installiert sein,
- SIFT Detector zum Ordner `BASE_PATH/bin/` kopiert werden,
- die ANN- Bibliothek "`libANN_char.so`" zum `LD_LIBRARY_PATH` unter Linux oder zum Ordner `BASE_PATH/lib` unter Windows kopiert und weiterhin müssen folgende drei Skripte editiert werden:

Das erste Skript, welches das Hauptskript darstellt, ist "RunBundler.sh":

1. Die Variable "BASE_PATH" , dient als Zugangweg

```
15  # Set this variable to your base install path (e.g., /home/foo/bundler
16  BASE_PATH="/media/win_e/Master/bundler-v0.4-source"
17
```

2. Die Variable "IMAGE_DIR" , dient als Zugangweg für Bilder

```
38  TO_SIFT=$BASE_PATH/bin/ToSift.sh
39
40  IMAGE_DIR="/media/win_e/Master/bundler-v0.4-source/examples/ul"
41
```

Das zweite Skript, welches ein Skript für SIFT- Detektor ist, heißt "ToSift.sh". In diesem Skript soll die Variable "BIN_PATH", die als Zugangweg dient, geändert werden:

```
6  # Set this variable to your base install path (e.g., /home/foo/bundler
7  BIN_PATH="/media/win_e/Master/bundler-v0.4-source/bin";
8
```

Das dritte Skript "extract_focal.pl" wurde in Perl geschrieben und enthält die Kamerakonstante. In diesem Skript soll die Variable " %ccd_widths" editiert werden, das heißt, der Name der Kamera und die Kamerakonstante sollen im folgenden Format hinzufügt werden:

```
172      "NIKON COOLPIX S7c"                => 5.76,   # 1/2.5"
173      "NIKON COOLPIX L22"                => 6.16,
174      "NIKON CORPORATION NIKON D100"     => 23.7,
```

Nach diesen Schritten kann Bundler mit dem Befehl "`./RunBundler.sh`" oder "`sh RunBundler.sh`" gestartet werden. Diese vier Schritte werden zum Erstellen einer Rekonstruktion durch Bundler durchgeführt:

1. Eine Liste der Bilder (`"list.txt"`) wird mit dem Skript `"extract_focal.pl"` erstellt,
2. SIFT- Zeichen werden für jedes Bild generiert,
3. Die Zeichen werden zwischen einzelnen Paaren von Bildern eingepasst, wobei dieser Schritt einige Zeit in Anspruch nehmen kann. Die berechneten Funktionen werden in der Datei `"matches.init.txt"` gespeichert,
4. Bundler wird mit einer geeigneten Optionsdatei ausgeführt und die visualisierten Daten werden anschließend im Format `".ply"` gespeichert.

Abbildung 21. Ein Bienenstock als ein Ergebnis von Bundler

6.2. Punktwolke erstellen

Ein weiterer Schritt bei der Herstellung eines 3D Stadtmodells ist die Extraktion einer Punktwolke aus Bundlerdaten. Die Daten müssen anfangs mit dem Programm `Bundle2PMVS` konvertiert werden. Dieses Programm wird mit dem Befehl "`./Bundle2PMVS <list.txt> <bundle.out> <...>`" gestartet, wofür die Dateien `<list.txt>` und `<bundle.out>` benötigt werden. `<...>` ist der Zugangsweg für konvertierte Daten, wie beispielsweise

"`./Bundle2PMVS ../list.txt ../bundle/bundle.out /pmvs`".

Während dieses Prozesses werden die benutzten Bilder mit ihren SIFT Zeichen und das Skript `"prep_pmvs"` erstellt. Dieses Skript erzeugt drei neue Ordner: `"models"`, `"txt"`,

"visualize" und kopiert die SIFT-Zeichen in das Format %08d.txt (zB. 0000000.txt) in den Ordner "txt". Gleichzeitig wird der Name des Bildes in %08d.jpg (zB. 00000000.jpg) geändert und in den Ordner "visualize" kopiert. Das nächste Verfahren ist CMVS (Clustering Views for Multi-View Stereo), welches vier Typen von Dateien generiert:

1. "Ske.dat" - Diese Datei enthält die Informationen über das Clustering. Die Datei beginnt mit dem Header "SKE". In der zweiten Zeile wird die Anzahl der benutzten Bilder und Cluster geschrieben. Die nächste Zeile enthält die Anzahl der benutzten Bilder des ersten Clusters. Anschließend werden die Indexe der Bilder genannt.

```
1  SKE
2  219 3
3  85 0
4  2 3 4 5 8 9 10 11 12 13 15 16 17 18 19 20 22 24 25 26 27 28 29 31 3
   44 46 47 48 49 50 51 52 55 56 57 58 59 60 61 63 64 67 68 69 71 72 7
   184 185 186 187 188 189 191 192 194 195 196 197 198 199 202 203 204
```

2. "Vis.dat" - Diese Datei beinhaltet optionale Eingabedaten für PMVS2 und enthält Informationen über die Nachbarschaft der Bilder. Die Datei beginnt mit dem Header "VISDATA". Die zweite Zeile gibt die Anzahl der benutzten Bilder wieder. Die erste Spalte der darauffolgenden Zeile enthält den Index des Bildes. In der zweiten Spalte wird die Anzahl der benachbarten Bilder sowie die Indexe dieser Bilder genannt.

```
1  VISDATA
2  219
3  0 0
4  1 0
5  2 30  5 10 4 11 186 185 187 12 188 9 184 3 13 19 8 18 17 16 189 2
6  3 30  10 5 11 187 188 185 186 12 184 189 13 4 2 19 8 18 16 17 9 1
7  4 30  11 2 12 189 8 188 13 184 187 185 10 186 19 3 18 5 9 15 20 3
```

3. "*centers-%04d.ply* (*centers-0000.ply*, *centers-0001.ply*, ...)" - Diese Datei enthält eine Reihe von 3D Punkten, wobei jeder Punkt die Kamera-Mitte eines Bildes aus dem entsprechenden Cluster darstellt. Dabei ist die Anzahl der Teile abhängig von der Leistung des Rechners.
4. Die 4. Datei "*centers-all.ply*" enthält die Zentren aller Kameras als 3D Punkte.

Dieses Programm wird mit dem Befehl "./CMVS ../bundle/bundle.rd.out" ausgeführt. Als nächstes wird das Programm genOption gestartet, das die Datei "ske.dat" liest und zwei Typen dieser erstellt:

1. "option-%04d" – in der alle Optionen für PMVS2 stehen sowie
2. "pmvs.sh"- beinhaltet eine Liste der Befehle zu PMVS2. Dieses Skript kann einfach ausgeführt werden, oder jeden Befehl unabhängig auf anderen Rechnern (oder parallel) ausführen, um die Rechenzeit zu minimieren.

`genOption` wird mit dem Befehl `"./genOption ../bundle/ske.dat"` ausgeführt.

Das Skript `"pmvs.sh"` startet das Programm PMVS2. Die Ergebnisse der Rekonstruktion werden in drei verschiedenen Dateien gespeichert:

1. **.ply* enthält farbige 3D Punkte zur Visualisierung.

```
ply
format ascii 1.0
element vertex 345189
property float x
property float y
property float z
property float nx
property float ny
property float nz
property uchar diffuse_red
property uchar diffuse_green
property uchar diffuse_blue
end_header
-0.356487 0.395937 -3.00367 -0.119983 -0.0423547 -0.991872 49 47 56
-0.374649 0.392585 -3.02147 -0.0707218 0.00681445 -0.997473 66 67 76
-0.405778 0.38927 -3.04188 0.393774 -0.0506677 -0.91781 116 123 140
-0.405653 0.389914 -3.04197 0.350418 0.00630754 -0.936572 117 123 141
```

2. **.patch* enthält die vollständigen Informationen über die Rekonstruktion. Die Datei beginnt mit dem Header "Patches" und in die zweite Zeile wird die Anzahl der rekonstruierten 3D Punkte geschrieben. Jeder einzelne Punkt soll von einer Zeile "PATCHS" angeführt werden. Die zwei nächsten Zeilen enthalten die 3D Lage und den geschätzten Normalenvektor. Die folgende Zeile enthält drei Zahlen. Die erste ist die photometrische Konsistenz des Punkts, die als eine Korrelation von -1.0 (schlecht) bis hin zu 1.0 (gut) normalisiert wird. Die anderen beiden Zahlen werden für das Debugging benutzt. Der nächste Block besteht aus zwei Zeilen. In der ersten Zeile wird die Anzahl der Bilder, in welcher der Punkt und die Textur sichtbar sind, geschrieben. Anschließend werden die Indexe dieser Bilder gespeichert. Die nächste Zeile enthält die Anzahl der Bilder, in denen der Punkt nicht sichtbar ist. Die sich anschließende Zeile enthält die Indexe der Bilder.

```
PATCHES
345189
PATCHS
-0.356487 0.395937 -3.00367 1
-0.119983 -0.0423547 -0.991872 0
0.74805 0.00244734 0.554281
3
0 41 18
2
25 49
```

3. **.pset* ist eine Liste der 3D Standorte und der geschätzten Normalenvektoren für alle rekonstruierten Punkte, die als Eingangsdaten für die PoissonRecon Software benutzt werden.

```
-0.356487 0.395937 -3.00367 -0.119983 -0.0423547 -0.991872
-0.374649 0.392585 -3.02147 -0.0707218 0.00681445 -0.997473
-0.405778 0.38927 -3.04188 0.393774 -0.0506677 -0.91781
-0.405653 0.389914 -3.04197 0.350418 0.00630754 -0.936572
```

PMVS2 ist der letzte Schritt bei der Erzeugung einer Punktwolke. Danach kann die Oberfläche aus der Punktwolke erstellt werden.

Abbildung 22. Erstellte Punktwolke in PMVS2

6.3. Oberflächenrekonstruktion

Der folgende Abschnitt befasst sich mit der automatischen Oberflächenrekonstruktion aus der Punktwolke mit dem Programm PoissonRecon. PoissonRecon wird mit dem Befehl `"PoissonRecon --in <...> --out <...> --depth 10"` gestartet, wobei `" --in"` eine *.pset Datei und `" --out"` der Name der ply-Datei ist , um das graue 3D Modell mit der rekonstruktierten Oberfläche zu abzuleiten.

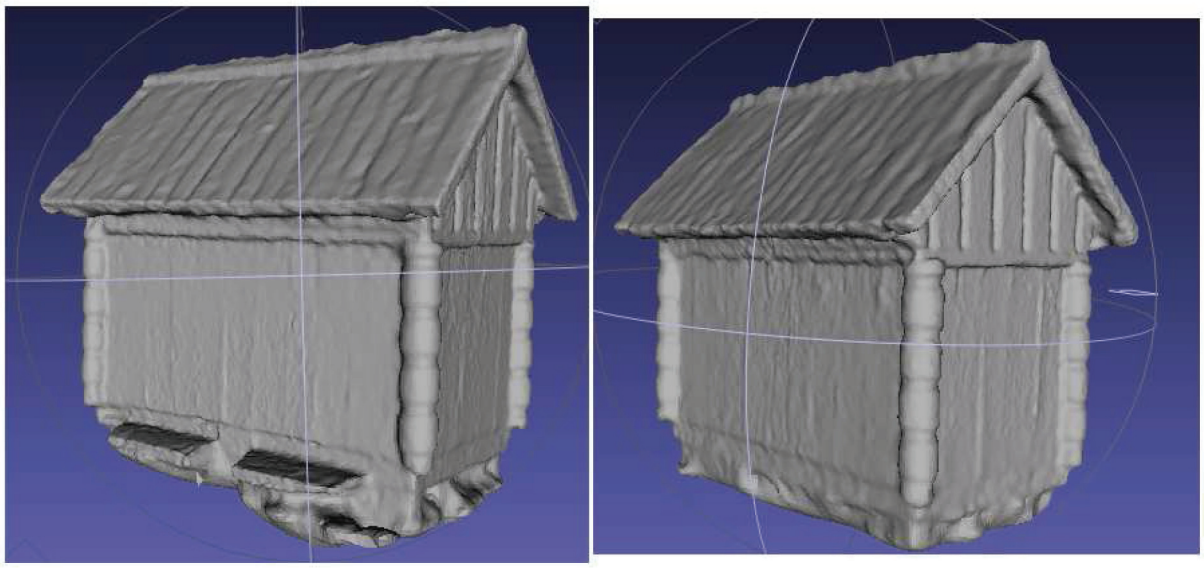


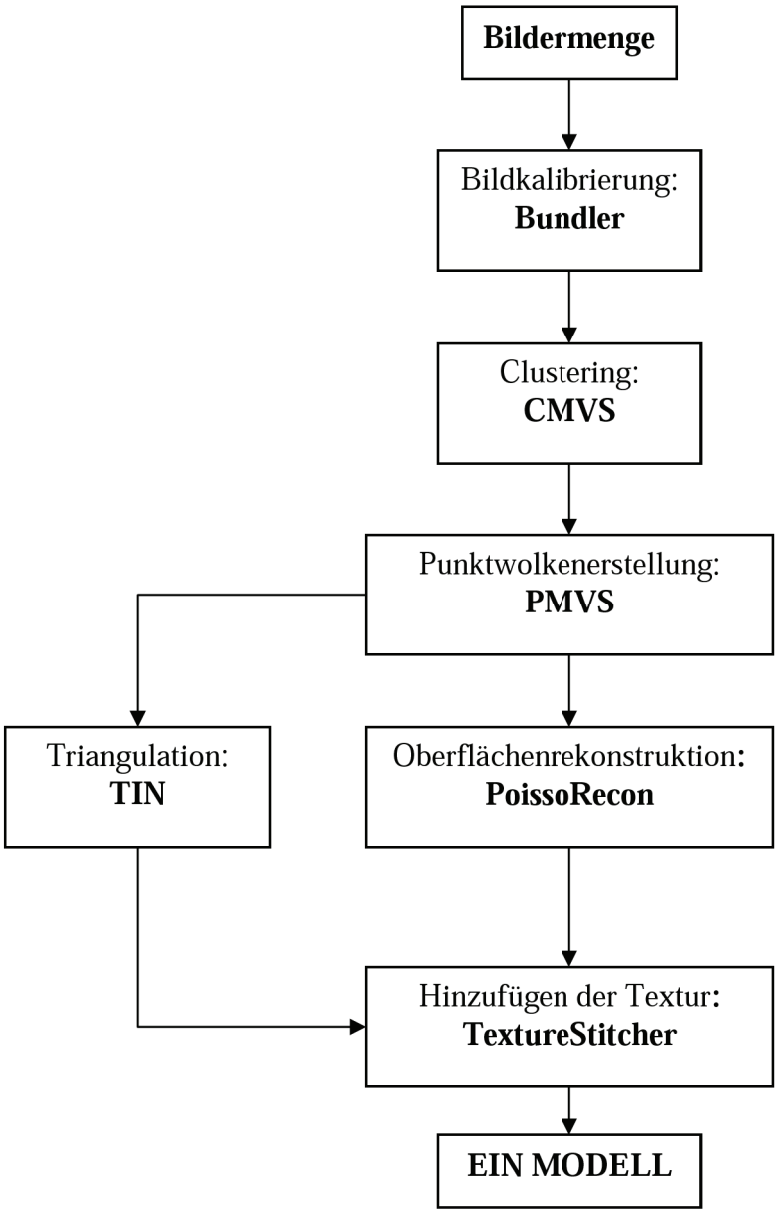
Abbildung 23. Rekonstruktierte Oberfläche in PoissonRecon

Nun wird die Färbung des 3D Modells vorgenommen und dabei Farbenlöcher entfernt. Hierzu wird zunächst ein PMVS- Modell mit dem Programm Meshlab erzeugt. Diese Punktwolke muss in ein TIN konvertiert und gespeichert werden. Das Netz wird leider ohne Farbe gespeichert, also muss es eingefärbt werden. Für diese Tätigkeit wird das TIN geöffnet und die Farben aus dem zugehörigen PMVS-Modell in Meshlab übernommen. Anschließend werden die bunten Netze in einer txt-Datei gespeichert. Das Programm (TextureStitcher) für die Färbung und die Farbenlochentfernung wird mit dem Befehl `TextureStitcher --in <.ply> --scans <.txt> --out <.ply> --useKD` ausgeführt. Hierbei ist `"--in"` das graue Modell nach PoissonRecon, `"--scans"` der Speicherort der TIN und `"--out"` die ply-Datei, in der das bunte 3D Modell gespeichert wird.

Abbildung 24. Texturierte Oberfläche

6.4. Blockschema

Im folgenden Kapitel wird ein Blockschema für den Ablauf des Algorithmus zur Erstellung eines automatischen 3D Modells dargestellt.



6.5. *Ergebnisse von einem 3D Stadtmodell*

Ein vorgestelltes Beispiel wurde aus 216 Bildern erstellt. Mit folgenden Bildern wurde ein 3D Stadtmodell dargestellt:

Abbildung 25. Die rekonstruierte Gebäuden

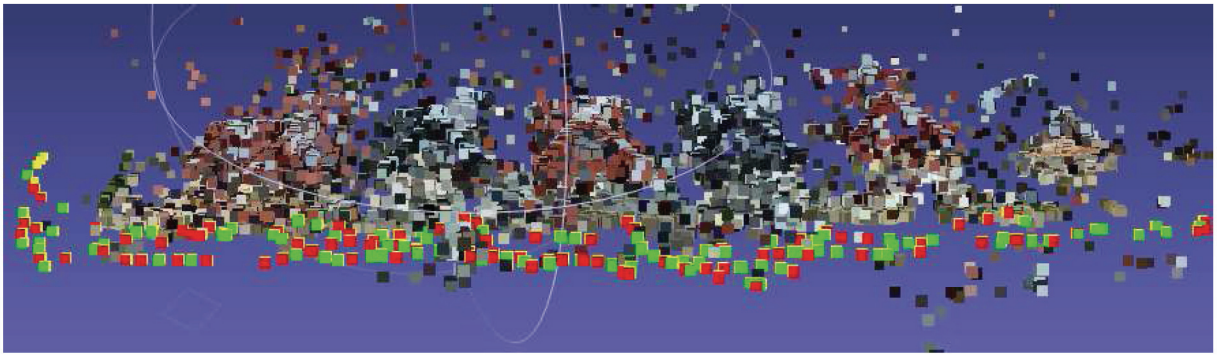


Abbildung 26. Ein 3D Stadtmodell nach Bundler

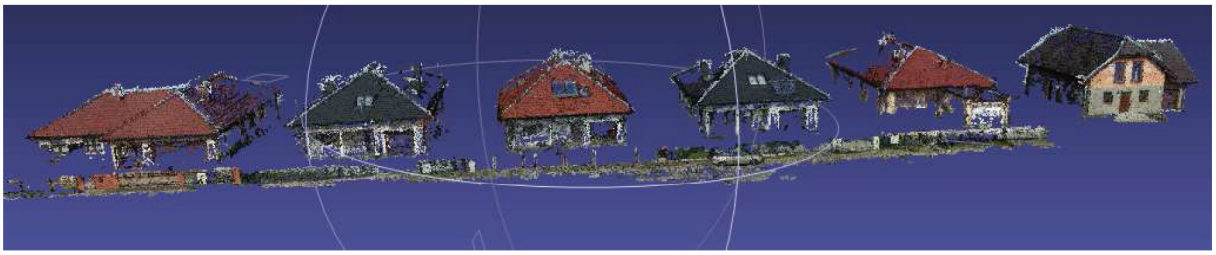


Abbildung 27. Ein 3D Stadtmodell nach PMVS

Abbildung 28. Ein 3D Stadtmodell ohne Umzäunung nach PMVS

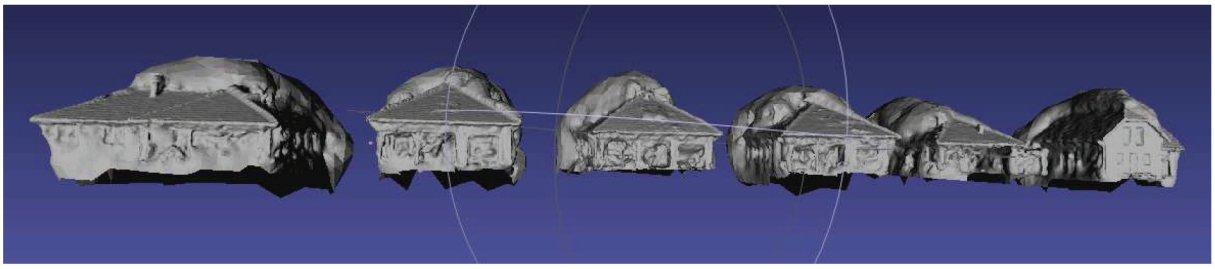


Abbildung 29. Ein 3D Stadtmodell mit rekonstruierten Oberflächen (PoissonRecon)

Abbildung 30. Ein texturiertes 3D Stadtmodell

7. Diskussion

Die Arbeit befasst sich mit den bestehenden und neuen Techniken zur Erstellung dreidimensionaler Stadtmodelle, sowie den gängigen Datengrundlagen. Neben diesem Themenfeld sollen mögliche Softwarelösungen untersucht werden. Das vorgestellte Verfahren basiert weitestgehend auf freien Softwareprogrammen von der University of Washington und setzt sich aus vier wesentlichen Komponenten zusammen. Eine gute Frage, die es in diesem Zusammenhang zu beantworten gilt, ist, wie viel Zeit man für die Erstellung eines (semi-) automatischen 3D Stadtmodells benötigt. Die Antwort auf diese Frage ist nicht so einfach, weil sie abhängig von der Anzahl der Bilder, Bildgröße und PC-Leistung ist. Die Untersuchungen für diese Masterarbeit wurden auf zwei Rechnern durchgeführt. Der erste Rechner hat die Parameter: AMD Athlon 2800+ (512KB Cache, 1.80 GHz), 2,5 GB RAM und der zweite Computer (ein Laptop) hat die Paramater: Intel Core 2 Duo T8300 (3MB Cache, 2.40 GHz,), 3.0 GB RAM. Die längste Laufzeit hat die erste Stufe Bundler, das heißt, SIFT-Zeichensuche und seine Anpassung. Ein Beispiel mit 795 Bildern dauert auf ersten Rechner 5 Wochen und auch auf dem zweiten Computer dauert dieser Vorgang fast eine Woche. Das Beispiel auf dem ersten PC wurde zweimal durch ein Gewitter unterbrochen, das heißt, dass dieser Rechner keine Stromsicherung (UPS) hat und die Daten folglich immer wieder gelöscht waren. Das Beispiel, das in dieser Masterarbeit beschrieben wurde, wurde nur auf dem Laptop erstellt und die erste Stufe (Bundler) nimmt eine Zeit von ca. 8 Stunden in Anspruch. Die nächste Stufe PMVS dauert ~ 1.5 Stunden, und die beiden letzten Schritte dauern insgesamt 0.5 Stunden. Die Erstellung des 3D Stadtmodells dauert folglich ungefähr 10 Stunden. In die folgende Tabelle wurden die Daten zusammenfasst.

795 Bilder	AMD Athlon 2800+	Intel Core 2 Duo T8300
Bundler	~5 Woche	~8 Stunden
PMVS	~1.5 Stunden	~1.5 Stunden
PoissoRecon +TextureStitcher	~0.5 Stunden	~0.5 Stunden
Σ	~5 Woche	~10 Stunden

Eine großflächige Erfassung von 3D Stadtmodellen ist fast ausschließlich durch die Auswertung von Luftbildern und Laserdaten möglich. Aufgrund der luftbasierten Datenerfassung sind gering- bis mitteldetaillierte Modelle mit flachen Fassaden und einfachen Dachstrukturen möglich (maximal LoD2). Die Rekonstruktion detaillierter

Fassadenstrukturen kann nur aus terrestrischen Bild- und Geometriedaten oder von Lasserdaten erfolgen (bis LoD4). Die vorgestellten Verfahren in dieser Arbeit können nicht ohne Weiteres an heutige Normen von OGC (CityGML) angepasst werden. Auf den ersten Blick kann das Modell bis zu einem Level of Details von 4 dargestellt werden, weil aus den Daten der Bilder zwar eine Punktwolke erstellt werden kann, es aber keine Verfahren gibt, welche daraus automatisch Kanten und Knoten erstellen. Durch automatische Erstellung wird das Modell heute nur bis zu einer Darstellung mit LoD2 visualisiert. Das vorgestellte Modell kann nur umständlich auf andere Konzepte, wie CityGML angepasst werden, da CityGML auf XML basiert. Die Daten werden strukturiert und alle geometrischen und semantischen Elemente mit einem spezifischen OGC-Code versehen. Ein Problem ist die Darstellung der Geometrie in CityGML. Dieses Problem kann mit der Hilfe Google Sketchup und zwei Plugins gelöst werden. Das erste verwendete Plugin ist ply-Importer von Jim's [24] und das zweite Plugin ist CityGML- Plugins von Fachhochschule Gelsenkirchen [25]. Die Daten müssen zunächst in Google Sketchup importiert werden, um sie anschließend automatisch in City-GML zu exportiern.

Die (semi-) automatische Erstellung des Modells führte während der Untersuchung in dieser Arbeit auf ein großes Problem. Dieses Problem heißt Dachformen und ist sehr gut aus dem Bereich der 3D Gebäude bekannt. Es existiert aber nicht für Gebäude mit nur einem Stock, weil eine Kamera mit ca. 10 m Abstand vom Objekt fast das ganze Dach erfassen kann. Dieses Problem kann gelöst werden, wenn der terrestrischen Bilderreihe einige Luftbilder hinzugefügt werden. Es dürfen hierbei keine senkrechten Luftbildaufnahmen verwendet werden, da der SIFT-Detektor dann keine Schlüsselpunkte finden kann. Am besten können Schrägluftbilder verarbeitet werden, weil auf jedem Bild die Wände mit charakteristischen Zeichen (z.B. Ecke oder Fenster) sichtbar sind und der SIFT-Detektor die gemeinsamen Punkte auf Luftbildern und Urlaubsfotos finden kann und so die Bilder zusammen kalibriert werden können.

Als großen Vorteile des vorgestellten Verfahrens sind die sehr günstigen Kosten und der einfache Zugang zu Daten für die Erstellung eines 3D Stadtmodells zu nennen. Die Daten können außer bei Verwendung eigener Bilder im Internet gefunden werden. Es gibt einige Portale, wie z.B. Panoramio von Google (www.panoramio.com) oder Flickr von Yahoo (www.flickr.com), in denen Leute ihre Urlaubsfotos hochladen. In Flickr erhält man beispielweise 8792 Bilder für das Stichwort „Rathaus München". Diese können heruntergeladen und zum 3D Gebäude weiterentwickelt werden. Eine gute und innovative Idee ist eine Nutzung von Google Street View Daten. Die Bilder können mit Hilfe der Taste

„PrtSc" (PrintScreen) erstellt werden und die Kamerastandorte können über die Variation der Ansicht geändert werden. Diese Quelle birgt jedoch zwei Schwierigkeiten. Zum einen gibt es keine Bilder der Dächer, was durch die Luftbilder gelöst werden kann. Zum anderen ist die Kamerakonstante der gemachten Bilder für die Bildkalibrierung in Bundler notwendig.

Eine der wichtigsten Fragen im Rahmen der Erstellung eines 3D Stadtmodells mit diesem Verfahren lautet: Wo kann man dieses Verfahren bei der Erzeugung eines 3D Stadtmodells nutzen? Die Antwort auf die Frage ist ganz einfach. Die Methode kann für alle heutigen Anwendungen benutzt werden, aber auch neue Anwendungen finden. Ein gutes Beispiel sind Kriegsschäden, wo nicht existente Denkmäler oder Stadtteile aus alten Bildern oder Postkarten rekonstruiert werden können.

8. Zusammenfassung

In den letzten Jahren wurden die 3D Stadtmodelle stark entwickelt und sind heute sehr populär. Es werden auch neue Methoden für virtuelle Städte im dreidimensionalen Raum erwünscht. In dieser Masterarbeit wurde gezeigt, wie ein 3D Stadtmodell (semi-) automatisches aus alternativen Quellen erzeugt werden kann. Der Ablauf wurde Schritt für Schritt dargestellt. Das vorgestellte Verfahren zeigt, wie die Kosten minimiert werden können, aber auch wie und wo die Daten zur Herstellung von 3D Objekten zu suchen sind. Es ist wünschenswert, das Verfahren weiter zu entwickeln und so zu optimieren, dass ein höherer Detaillierungsgrad in CityGML möglich ist.

9. Literaturverzeichnis:

[1] Lowe David G.: Distinctive Image Features from Scale-Invariant Keypoints, 2004.

[2] Mount David M.: ANN Programming Manual, Department of Computer Science and Institute for Advanced Computer Studies University of Maryland, College Park, Maryland, 2006.

[3] Vogiatzis George, Carlos Hernandez Esteban, Philip H. S. Torr, Roberto Cipolla: Multi-view Stereo via Volumetric Graph-cuts and Occlusion Robust Photo-Consistency, 2007.

[4] http://phototour.cs.washington.edu/bundler
Stand:10.02.2011

[5] http://de.wikipedia.org/wiki/Scale-invariant_feature_transform
Stand:10.02.2011

[6] http://de.wikipedia.org/wiki/ImageMagick
Stand:10.02.2011

[7] http://www.cs.jhu.edu/~misha/Code/PoissonRecon/
Stand:10.02.2011

[8] http://www.imagemagick.org/script/index.php
Stand:10.02.2011

[9] http://www.cs.jhu.edu/~misha/Code/TextureStitcher/range-grid.ply.html
Stand:10.02.2011

[10] http://grail.cs.washington.edu/software/pmvs/
Stand:10.02.2011

[11] http://www.cs.jhu.edu/~misha/Code/TextureStitcher/
Stand:10.02.2011

[12] http://www.cs.umd.edu/~mount/ANN/
Stand:10.02.2011

[13] OGC, OpenGIS® City Geography Markup Language (CityGML) Encoding Standard.

[14] Kazhdan Michael, Matthew Bolitho, Hugues Hoppe: Poisson Surface Reconstruction, 2006, Eurographics Symposium on Geometry Processing.

[15] Chuang Ming, Linjie Luo, Benedict J. Brown, Szymon Rusinkiewicz, Michael Kazhdan: Estimating the Laplace-Beltrami Operator by Restricting 3D Functions, 2009, Eurographics Symposium on Geometry Processing.

[16] Snavely Noah, Steven M. Seitz, Richard Szeliski: Photo Tourism: Exploring Photo Collections in 3D, 2009.

[17] Snavely Noah, Ian Simon, Michael Goesele, Richard Szeliski, Steven M. Seitz: Scene Reconstruction and Visualization From Community Photo Collections, 2010, IEEE.

[18] Agarwal Sameer, Yasutaka Furukawa, Noah Snavely, Brian Curless, Steven M. Seitz, Richard Szeliski: Reconstructing Rome, 2010, IEEE.

[19] Furukawa Yasutaka, Jean Ponce: Accurate, Dense, and Robust Multi-View Stereopsis, 2008, IEEE Transactions on Pattern Analysis and Machine Intelligence.

[20] Furukawa Yasutaka, Jean Ponce: Accurate, Dense, and Robust Multi-View Stereopsis, 2007, IEEE Computer Society Conference on Computer Vision and Pattern Recognition.

[21] Zeile Peter: Erstellung und Visualisierung von virtuallen 3B-Stadtmodellen aus kommunalen Geodaten am Beispiel des UNESCO Welterbes Bamberg.

[22] Brenner Claus, Norbert Haala: Erfassung von 3D Stadtmodellen, PFG – Photogrammetrie, Fernerkundung Heft 2/2000, 2000.

[23] Kolbe, Thomas H.: CityGML – Ein Standard für virtuelle 3D Stadtmodelle, 2008.

[24] http://sketchuptips.blogspot.com/2010/03/plugin-ply-file-importer-for-sketchup.html
Stand:10.02.2011

[25] http://www.citygml.de/index.php/sketchup-citygml-plugin.html
Stand:10.02.2011

[26] Henricsson Olof: The Role of Color Attributes and Similarity Grouping in 3D Building Reconstruction, Computer Vision and Image Understanding, 1998.

[27] Schiebold Michael: Konzeption eines virtuellen Stadtmodells, 2007.

[28] Lancelle Marcel: Automatische Generierung und Visualisierung von 3D Stadtmodellen, 2004.

[29] Steinhage Volker: CAD-basierte Modellierungstechniken zur Generierung von 3D Stadtmodellen aus Luftbildern, 2000.

[30] http://www.gta-geo.com/
Stand:10.02.2011

[31] GTA, Praxistest bestanden und nun neu auf dem Markt: Software für die automatische Erzeugung von 3D Stadtmodellen, 2011.

[32] http://www.c3technologies.com/
Stand:10.02.2011

[33] http://www.faro.com
Stand:10.02.2011

[34] Henricsson Olof, Frank Bignone, Wolfram Willuhn, Frank Ade, Olaf Kiibler: Project AMOBE: Strategies, current status and future work, 1996.

[35] Hirschmüller Heiko: Stereo Processing by Semi-Global Matching and Mutual Information, IEEE Transactions on Pattern Analysis and Machine Intelligence, 2008.

[36] Hirschmüller Heiko, Werner Randelshofer, Christian Schulz: Stereo Processing by Semi-Global Matching and Mutual Information, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vortrag, 2008.

[37] Föll Gregory: Stereo Vision: Vergleich verschiedener Algorithmen zur Lösung des Korrespondenzproblems, Hochschule für Angewandte Wissenschaften Hamburg, 2010.

[38] http://de.wikipedia.org/wiki/Transinformation
Stand:10.02.2011

Abbildungsverzeichnis