Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Aleksej Dygoduk

Entwicklung einer Android App für 3D Rekonstruktion

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Aleksej Dygoduk

Entwicklung einer Android App für 3D Rekonstruktion

Bachelorarbeit eingereicht im Rahmen der Bachelorprufung

im Studiengang Bachelor of Science Angewandte Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Ing. Birgit Wendholt
Zweitgutachter: Prof. Dr. sc. pol. Wolfgang Gerken

Eingereicht am: 30. August 2012

Aleksej Dygoduk

Thema der Arbeit
Entwicklung einer Android App für 3D Rekonstruktion

Stichworte
3D-Rekonstruktion, Metrische Rekonstruktion, Kamerakalibrierung, Triangulation, Harris corner detection, ORB, Android, OpenCV, Smartphone

Kurzzusammenfassung
In dieser Arbeit wird eine Lösung entwickelt die ein alltägliches Gerät, wie Smartphone, in einen 3D-Scanner im Taschenformat verwandelt. Dabei werden zunächst mehrere gängige Verfahren untersucht und deren Eignung bzw. Vor- und Nachteile im Zusammenhang mit der Zielplattform und späterer Einsatzweise festgestellt. Für die Umsetzung wird ein Algorithmus für die metrische 3D-Rekonstruktion ausgewählt, der von Marc Pollefeys in seiner Arbeit 1999 vorgestellt wurde. Dieses Verfahren zeichnet sich dadurch aus, dass man einfach nur ein Paar Aufnahmen mit einem Android Smartphone machen muss, um ein 3D-Modell von einem Objekt erstellen zu können.
Aleksej Dygoduk

Title of the paper
Developing Android App for 3D reconstruction

Keywords
3D reconstruction, Metric reconstruction, camera calibration, triangulation, Harris corner detection, ORB, Android, OpenCV, Smartphone

Abstract
In this paper a solution is developed that makes a device like smartphone to a pocket 3D scanner. Initially, several common methods are examined and their suitability as well as advantages and disadvantages associated to the target platform and later use manner determined. For the implementation is a metric 3D reconstruction algorithm selected, which was presented by Marc Pollefeys in his work 1999th. This method is characterized by the fact that you just only have to take some photos of the object in order to create its 3D model.

Inhaltsverzeichnis

Danksagung

Zunächst möchte ich mich an dieser Stelle bei all denjenigen bedanken, die mich während der Anfertigung dieser Bachelorarbeit unterstützt und motiviert haben.

Ganz besonders gilt dieses Dank Prof. Dr. Ing. B. Wendholt, die meine Arbeit und somit auch mich betreut hat. Nicht nur, dass sie immer wieder durch kritisches Hinterfragen wertvolle Hinweise gab, auch ihre moralische Unterstützung und Motivation waren unschlagbar. Sie hat mich dazu gebracht, über meine Grenzen hinaus zu denken. Vielen Dank für die Geduld und Mühen.

Herrn Professor Dr. sc. pol. W. Gerken danke ich für seine Bereitschaft, das Zweitgutachten zu erstellen.

Daneben gilt mein Dank meiner Freunde, die in zahlreichen Stunden Korrektur gelesen haben. Sie wiesen auf Schwächen hin und konnten als Fachfremde immer wieder zeigen, wo noch Erklärungsbedarf bestand.

Nicht zuletzt gebührt meinen Eltern Dank, da Sie während des Studiums nicht nur finanziell, sondern vor allem auch emotional immer für mich da waren.

1 Einleitung

1.1 Motivation

Worte sind manchmal nicht ausreichend um etwas genauer zu beschreiben und Bilder schaffen es nicht immer den Sachverhalt zu veranschaulichen um das gewünschte Ergebnis zu erzielen. Eine mögliche Hilfestellung können hier die 3D-Modelle bieten. Durch eine automatisierte 3D-Rekonstruktion[1], die nur durch einen Knopfdruck die Erstellung eines 3D-Modells aus vorliegenden Bildern ermöglicht, wird der Modellierungsprozess für jeden Nutzer vereinfacht und zugänglich gemacht. Die automatisierte 3D-Rekonstruktion bringt den Vorteil mit sich, dass kein fachliches Wissen erforderlich ist und es findet eine Reduzierung der Ausführungszeit statt. Die 3D-Rekonstruktion ist im alltäglichen Leben weit verbreitet und nicht mehr weg zu denken. Sie wird zum Beispiel in der Medizin, der Kriminalistik oder der Geomodellierung eingesetzt.

Die Tomografie spielt eine wichtige Rolle in der Medizin. Das Schnittbildverfahren ermöglicht den Ärzten einen Einblick in den menschlichen Körper durch die entstehenden Schnittbilder (Tomogramme). Sie liefern aus einem realen Objekt, also aus der dreidimensionalen Wirklichkeit, zweidimensionale Ausschnitte in Form von Schnittbildern. Aus diesen Schnittbildern wird mit Hilfe der 3D-Rekonstruktion ein 3D-Modell vom gewünschten Objekt (zum Beispiel eines Organs des menschlichen Körpers) erstellt. Dies ist für viele medizinische Zwecke wünschenswert wie zum Beispiel bei der Diagnose einer Erkrankung und deren Behandlung. Die 3D-Rekonstruktion wird in der Rechtsmedizin und Kriminalistik eingesetzt um aufgrund von oberflächlichen und inneren Verletzungen des Opfers den Tatvorgang zu rekonstruieren und ein Simulationsmodell zu erstellen. Die Verwendung von 3D-Rekonstruktion in der Geomodellierung vereinfacht die Planung der Städte bzw. die Neuerschließung der Stadtteile und ermöglicht die Erstellung von dreidimensionalen Karten bzw. Stadtplänen, so wie die Bearbeitung der Entwürfe auf digitaler Basis.

Die meisten Lösungen für die 3D-Rekonstruktion werden heute vorwiegend nur für leistungsfähige Computer angeboten und erfordern oft noch den Einsatz von zusätzlichen und

[1] Unter 3D- Rekonstruktion versteht man eine möglichst realitätsnahe Nachbildung von Objekten durch Computermodelle.

kostspieligen Geräten. Dies nimmt dem Mensch die Chance mobil und flexibel zu sein. Um die Verwendung von 3D-Modellen bzw. der 3D-Rekonstruktion für jedermann im alltäglichen Leben zugänglich zu machen, ohne weitere finanzielle Belastung, ist eine Lösung notwendig, die jeder Zeit einsatzbereit ist.

Der heutige Lebensrhythmus erfordert von den Menschen kommunikabel und flexibel zu sein. Dies gilt vor allem für das Berufsleben. Dieser Anforderungen werden viele gerecht in dem sie mit den Erneuerungen auf dem technischen Markt Schritt halten. Deshalb bevorzugen immer mehr Menschen Smartphones (Abbildung 1.1), welche die Möglichkeit bieten verschiedene Aufgaben von unterwegs zu erledigen. Egal ob es um das Schreiben von Emails, das Verwalten eines Terminkalenders oder um die Informationbeschaffung aus dem Internet geht, das Smartphone macht dies möglich. Im privaten Alltag werden die Smartphone auch immer beliebter. Sie gewähren neben den gewöhnlichen Handyfunktionen auch ein kompaktes Mediencenter, durch welches das Mitnehmen von zusätzlichen Geräten, wie Mp3-player, Navigationsgeräte oder der Fotokamera nicht mehr nötig ist.

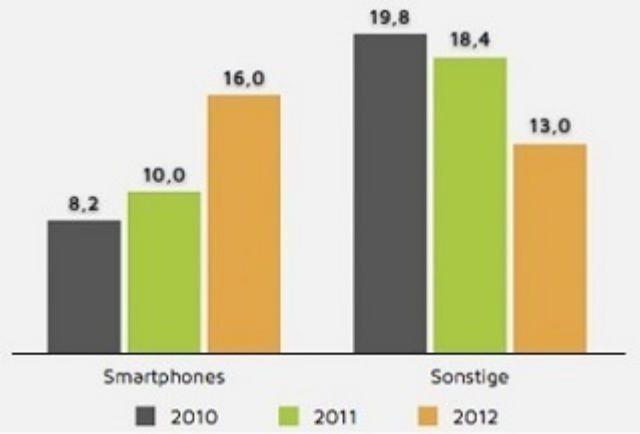

Abbildung 1.1: Der Popularitätszuwachs der Smartphones in Deutschland von 2010 bis 2012. Quelle: [http://unhive.com, 2010]

Heutzutage werden viele Smartphones mit dem Betriebssystem Android betrieben (Abbildung 1.2). Dieses System findet eine große Akzeptanz nicht nur bei den Softwareentwicklern sondern auch bei den Nutzern. Die Smartphones, die Android als Betriebssystem aufweisen, sind im Vergleich zu anderen Geräten mit alternativen Betriebssystemen, preiswerter. Außerdem haben die Nutzer eine breitgefächerte Palette an Apps, die zum größten Teil kostenlos oder preiswert sind ohne das die Qualität beeinträchtigt wird. Die Softwareentwickler besitzen hier die Möglichkeit einer bequemen und kostenlosen Nutzung von Entwicklungsumgebungen.

Abbildung 1.2: Die Statistik der unterschiedlichen Smartpone-Betriebssysteme in Deutschland. Quelle: [http://www.bitkom.org/, 2012]

Durch die oben beschriebenen Auswahlkriterien kommt man zu dem Schluss, die 3D-Rekonstruktion mobil und flexibel zu machen. Diese Idee ermöglicht dem Menschen dreidimensionale Modelle zu erstellen und zwar immer und überall ohne an bestimmte Orte, Geräte oder Zeiten gebunden zu seien. Um die 3D-Rekonstruktion durchzuführen, benötigt man Bilder aus unterschiedlichen Perspektiven des gewünschten Objekts, die man mittels des Smartphones vor Ort aufnehmen kann. Durch die vorliegenden Bilder kann das 3D-Modell direkt und ohne Zeitverzögerung rekonstruiert werden.

In den letzten 25 Jahren wurden unterschiedliche Methoden erarbeitet um den Rekonstruktionsprozess der 3D-Modelle zu ermöglichen und zu vereinfachen. Manche von diesen Methoden sind im Stadium der Theorie geblieben, andere wiederrum haben es in die praktische Umsetzung geschafft. Die Entwicklung steht aber nicht still. Die Methoden werden immer weiter verfeinert und dem Stand der heutigen Erkenntnisse, in Verbindung mit der Technik, angepasst.

Der berufliche Alltag verlangt von vielen Menschen den Einsatz von der 3D-Rekonstruktion an Ort und Stelle um einerseits die gestellten Aufgaben effizienter zu bewerkstelligen und andererseits sich die Arbeit zu erleichtern. Deswegen steigt die Nachfrage nach Entwicklungen, die dies ermöglichen. Das Bestreben wird immer stärker den Menschen die gewünschte Lösung zu bieten. Um dies zu ermöglichen benötigt man ein Gerät, welches untrennbar mit dem alltäglichen Leben verbunden und aus diesem nicht mehr wegzudenken ist. Bei diesem Gerät handelt es sich um das Smartphone.

Um die 3D-Rekonstruktion für eine möglichst große Anzahl von Menschen zugänglich zu machen ohne unnötigen Aufwand zu betreiben, ist es vom Nutzen ein weit verbreitetes Betriebssystem auszuwählen. Der Entwicklungsprozess der Anwendungen für das Betriebssystem soll nach Möglichkeit einfach und ohne zusätzlichen Aufwand für den Softwareentwickler sein. Durch diese Auswahlkriterien werden die Systeme selektiert und Android bietet die bestmöglichen Voraussetzungen um die gewünschte Lösung zu erzielen.

1.2 Zielsetzung

Die Zielsetzung für diese Arbeit ist die Erarbeitung einer Anwendung für das Android-Betriebssystem, mittels welcher eine 3D-Rekonstruktion von einem Objekt mit Hilfe von einer bestimmten mindest Anzahl von Bildern ermöglicht wird. Im Rahmen dieser Aufgabe werden die verschiedenen Methoden und Algorithmen erläutert, diskutiert und zusammengefasst. Im Verlauf der Bearbeitung wird aus den angebotenen Möglichkeiten ein Lösungsweg ausgewählt, welcher die besten Voraussetzungen bietet, um praktisch umgesetzt zu werden.

1.3 Gliederung der Arbeit

Die verschiedenen Kapitel dieser Arbeit haben unterschiedliche Schwerpunkte. Kapitel Vergleichbare Arbeiten beschäftigt sich mit dem aktuellen Stand der Entwicklungen im Gebiet 3D-Rekonstruktion. Es werden zwei Ansätze präsentiert, die wie in dieser Bachelorarbeit das Problem lösen, aber nicht für Smartphones geeignet sind.

Im Kapitel Grundlagen werden verschiedene Verfahren für 3D-Rekonstruktion vorgestellt und diskutiert. Es wird festgestellt, welches Verfahren in der Anwendung, die für Smartphones entwickelt wird, eingesetzt werden kann. Kapitel Analyse beschreibt einzelne Schritte von ausgewählten Rekonstruktionsverfahren detailliert.

Im Kapitel Laufzeitumgebung und Bibliotheken wird das Android-Betriebssystem und freie Bibliothek für Bildverarbeitung bekannt gegeben. Von der Theorie zur Praxis kommt man in Kapitel Realisierung, wo unterschiedliche Verfahrens, wie SURF, Harris corner detector und andere, getestet wurden.

2 Vergleichbare Arbeiten

In den letzten Jahren gibt es immer wieder Weiterentwicklungen und Erneuerungen auf dem Gebiet der 3D-Modellierung. Es wurde eine Vielzahl an Entwicklungsmöglichkeiten angeboten, aber nicht viele haben es über den Stand des theoretischen Modells geschafft. Dieses Kapitel beschäftigt sich mit dem aktuellen Stand der Entwicklungen in dem Bereich der 3D-Rekonstruktion mit dem Hinblick auf die verschiedenen schon vorhandenen Programme, so wie deren praktische Umsetzung und dem Ergebnis. Autodesk 123D Catch und Visual SFM sind einige der wenigen 3D-Rekonstruktionsprogramme, die gebührenfrei zur Verfügung stehen und ein akzeptables Ergebnis liefern. Die beiden Programme wurden Anhand der 3D-Rekonstruktion aus derselben Bildsequenz der Safttüte getestet.

2.1 Autodesk 123D Catch

Das weltgrößte Softwareunternehmen in dem Bereich des digitalen 2D- und 3D-Designs namens Autodesk entwickelte die erste Version der 3D-Rekonstruktionssoftware Autodesk 123D Catch im Jahre 2009. Im Frühling 2011 wurde eine Beta-Version veröffentlicht und seit März 2012 ist eine stabile Version verfügbar. Diese Software ist für das Windows-Betriebssystem entwickelt worden und stellt folgende Systemanforderungen:

- Microsoft Windows 7 (32-bit and 64-bit), Microsoft Windows XP Service Pack 3 or higher (32-bit and 64-bit).
- Intel® Core™2Duo
- 1 GB RAM
- 1 GB Speicherplatz
- OpenGL-unterschtützte Grafikkarte mit mindestens 250Mb Speicher
- Internetverbindung

Die Autodesk 123D Catch-Homepage[1] bietet die Möglichkeit die Software problemlos herunterzuladen. Die Installation ist in wenigen Minuten abgeschlossen und Autodesk 123D Catch einsatzbereit. Man hat auch die Möglichkeit die Software in Form der Web-Applikation auf der Homepage zu nutzen. Es besteht zudem noch die Möglichkeit mit Hilfe der vorhandenen Videoanweisung den Umgang mit der Software zu erlernen.

Um das gewünschte Objekt als 3D-Modell zu bekommen, benötigt man für die 3D-Rekonstruktion Bilder von dem Objekt. Um das bestmögliche Ergebnis zu erzielen, sollen folgende Kriterien beim Bilduafnahme beachtet bzw. erfüllt werden:

- Die Aufnahme soll in ca. 20-Grad-Schritten um das Objekt durchgeführt werden.
- Die benachbarten Aufnahmen sollen unbedingt große Überlappungen haben
- Während der Aufnahme bewegt sich nur die Kamera und auf keinen Fall das Objekt selbst.
- Das Objekt darf nicht transparent, gespiegelt oder glänzend sein.
- Die Beleuchtung soll gleichmäßig und konstant sein. Aufnahmen mit Blitz sind von Vorteil.
- Auf gleichmäßig gefärbten Oberflächen sollen zum Beispiel farbige Papierstücke geklebt werden.
- Es sollte keine Array von identischen Objekte sein
- Es macht keinen Sinn, Bilder in Auflösung größer als 3 Megapixel zu machen.
- Es können maximal 70 Bildaufnahmen verwendet werden.

Nach dem man die benötigten Aufnahmen gemacht hat, erfolgt das Hochladen der Bilder auf den Server, wo das 3D-Modell rekonstruiert wird. Das Ergebnis der Modellierung, also das fertige 3D-Modell, wird automatisch vom Server auf den Computer heruntergeladen. Der gesamte Vorgang (Abbildung 2.1) nimmt ca. drei bis zehn Minuten in Anspruch.

Als Ergebnis bekommt man ein texturiertes 3D-Modell und ein Schema von Kamerapositionen, von denen die Aufnahmen durchgeführt wurden (Abbildung 2.2). Unter dem Modell findet man alle aufgenommenen Bilder (Quellenbilder), welche auf den Server hochgeladen wurden.

[1] http://www.123dapp.com/

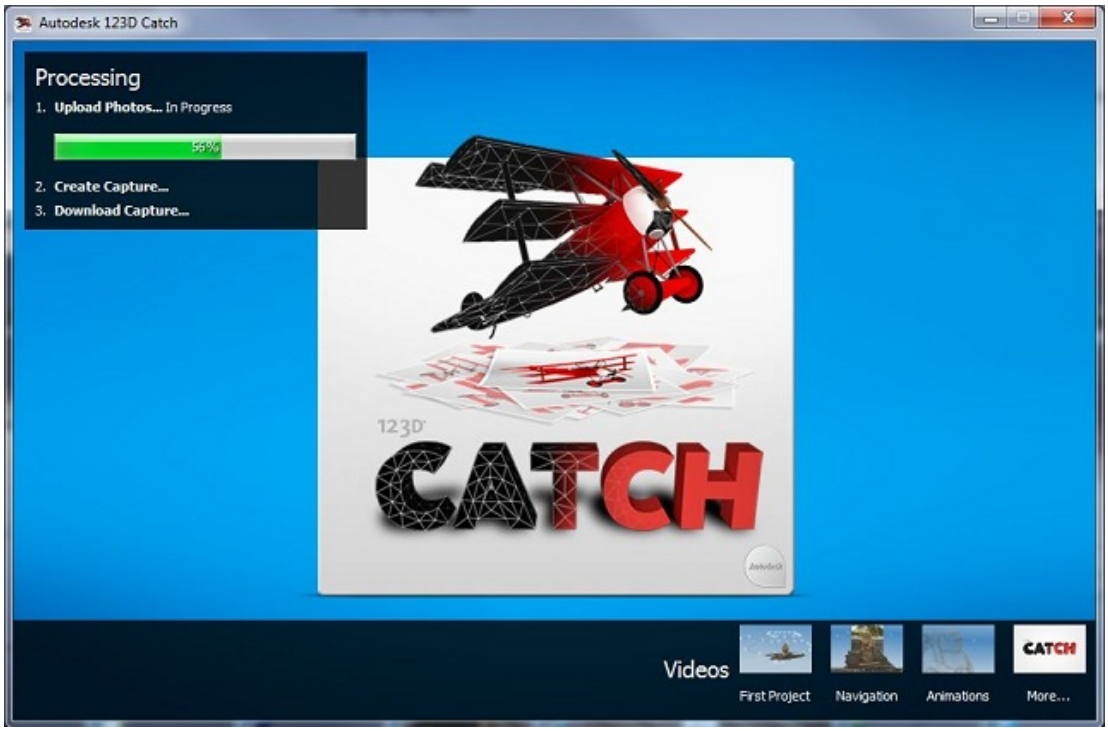

Abbildung 2.1: Erstellung eines 3D-Modells mit Autodesk 123D Catch.

Bilder, welche zu dem berechneten Modell nicht zugeordnet werden konnten, markiert das Programm mit einem gelben Rahmen und einem Dreick der gleichen Farbe. Auf diesen Bildern kann man die markanten Stellen manuell markieren und das Modell neu berechnen lassen. Am Ende hat man die Möglichkeit das 3D-Modell in unterschiedlichen Formaten wie .*dwg, .*fbx, .*OBJ zu speichern. Bei Bedarf kann man eine Videodatei des 3D-Modells erstellen. Diese Videodatei zeigt das Objekt aus verschiedenen Perspektiven, die zuvor aus den Schemata der verschiedenen Kamerapositionen gewählt wurden.

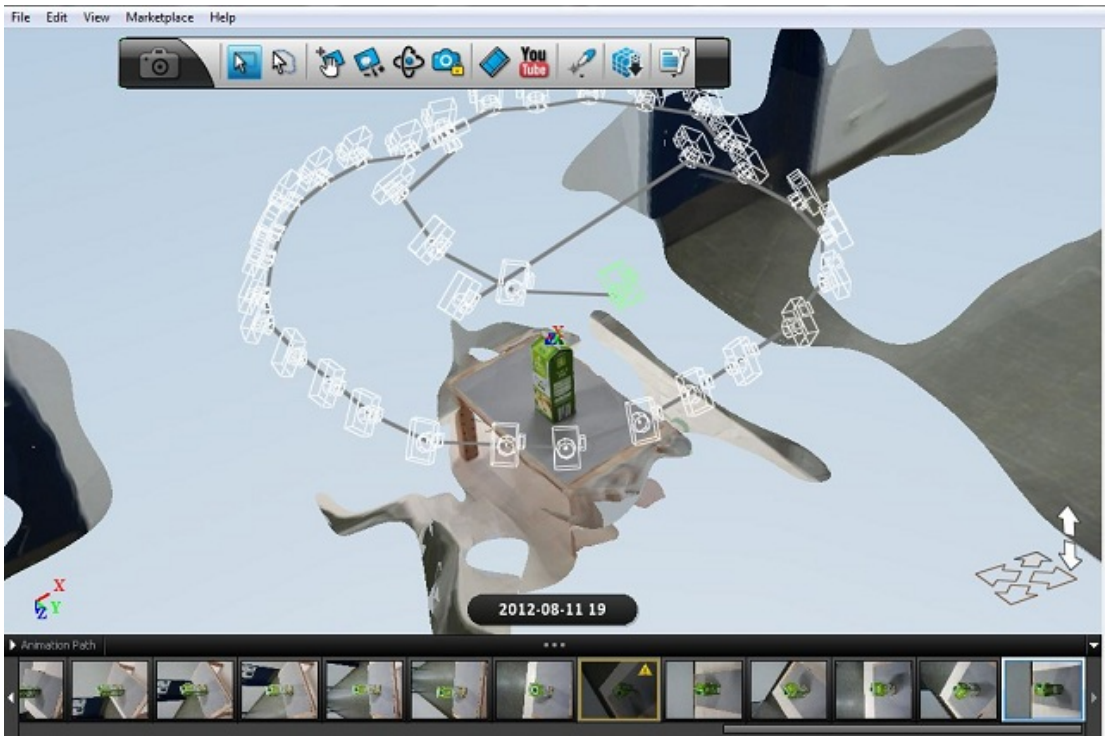

Abbildung 2.2: Von Autodesk 123D Catch generiertes 3D-Modell

2.2 Visual SFM

Bei dem VisualSFM handelt es sich auch um eine 3D-Rekonstruktionssoftware, welche von einem Student namens Changchang [Wu] entwickelt wurde. Im Jahre 2006 wurde diese Software als ein Grundkonzept für das „3D Urban Modeling“- Seminar an der University of North Carolina in Chapel Hill erarbeitet. VisualSFM verfügt über eine grafische Benutzeroberfläche und obwohl es immer noch mit Features und Optionen verbessert wird, bietet es hervorragende Fähigkeiten zur schnellen Verarbeitung von Hunderten von Fotos durch die Nutzung der nVidia oder ATI Grafikkarten des Computers.

Um mit der VisualSFM-Software arbeiten zu können, lädt man als erstes diese Software von der Hompage [2] herunter. Die herunter geladene Datei ist komprimiert und trägt den Namen $VisualSFM\_windows\_64bit.zip$. Die Datei wird entpackt und es erscheint ein Ordner, welcher alle Hauptbestandteile des VisualSFM enthält. Für diese Software entfällt der Installationsprozess, da die VisualSFM-Datei selbstausführend ist. Um die Hauptbestandteile der Software zu erweitern und das 3D-Rekonstruktionsmodell anschaulicher zu machen,

[2] http://www.cs.washington.edu/homes/ccwu/vsfm/

kann man die dazugehörigen Optionen PMVS/CMVS-Kode[3] herunterladen und die sollen in demselben Ordner wie die VisulaSFM.exe- Datei platziert werden. Eine denkbare Option ist zum Beispiel die Erstellung von Punktewolken. Punktewolken sind die Punkte, die das Programm für die 3D-Rekonstruktion bestimmt bzw. errechnet hat. In dem Basispaket ist eine bestimmte Anzahl der Punktewolken erhalten. Die Erweiterung dieser Option ermöglicht eine dichte Punktewolke zu bekommen, also werden mehr Punkte des gewünschten Objekts in dem 3D-Modell angezeigt und berücksichtigt. In dem Fall, das man eine nVidia Grafikkarte besitzt, soll man als ersten Schritt das CUDA Toolkit direkt von der NVIDIA-Website herunterladen und installieren. Ohne diesen Vorgang ist es sonst nicht möglich die Software VisualSFM in Betreib zu nehmen.

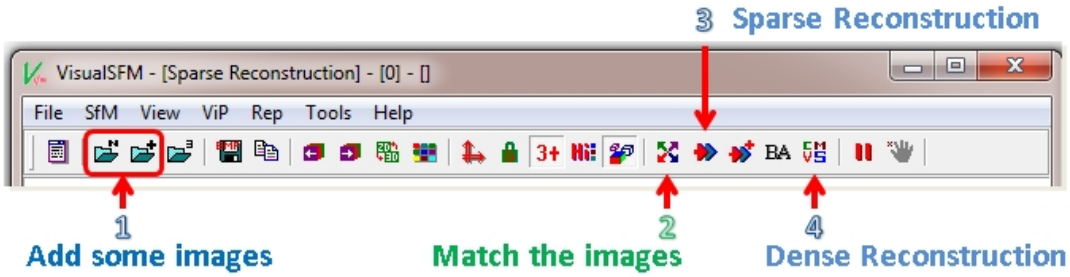

Abbildung 2.3: Die typischen 3D-Rekonstruktionschritte mit VisualSFM. Quelle: [Wu]

Nach der Ausführung des Programms sieht man ein Hauptfenster und rechts davon einen „Task Viewer“, dieser dient zur Beobachtung der einzelnen Schritte und des Gesamtprozesses. Die 3D-Rekonstruktion wird typischerweise in vier Schritten durchgeführt (sehe Abbildung 2.3). Die Abbildungen 2.4 und 2.5 unten zeigen das Ergebnis der 3D-Rekontruktion aus den 34 Bildern der zur Verfügung stehenden Bildsequenz.

[3] https://github.com/TheFrenchLeaf/CMVS-PMVS

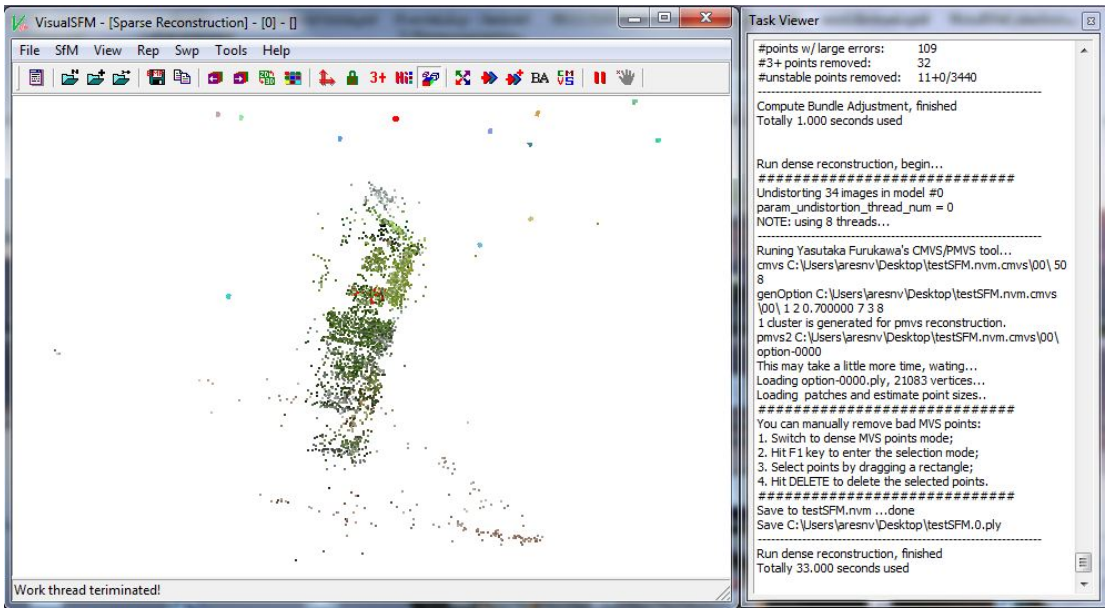

Abbildung 2.4: Das 3D-Modell als Punktewolke mit geringer Dichte

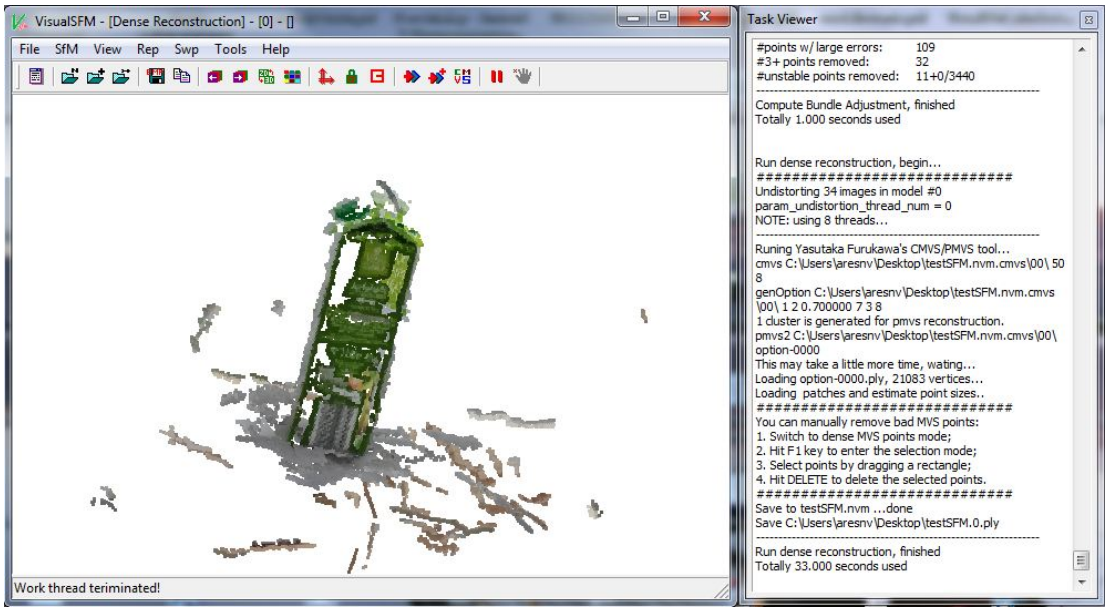

Abbildung 2.5: Das 3D-Modell als dichte Punktewolke

2.3 Zusammenfassung

Beide vorgestellte Lösungsvarianten für die 3D-Rekonstruktion orientieren sich für die Anwendung auf einem Computer. Bei der ersten Software, Autodesk 123D Catch, werden die Aufgaben der Bildverarbeitung und die Berechnung des 3D-Modells, welche viel Leistung von dem Computer abverlangen, auf einem Web-Server durchgeführt. Der Rechner des Nutzers wird nur durch die grafische Oberfläche der Darstellung sowie die Nachbearbeitung der 3D-Modelle beansprucht. Bei der zweite Software findet der ganze Modellierungsprozess auf dem Benutzerrechner statt und abverlangt diesem Leistungen, vor allem liegt der Schwerpunkt auf der Grafikkarte.

Auf dem Markt sind auch andere Programme vorhanden, die meisten davon sind aber kostenpflichtig. Diese Programme wurden auch für die Nutzung an einem Computer entwickelt. Ein Beispiel einer solchen Software ist der PhotoModeller[4].

Die IEEE[5] organisiert alle zwei Jahre eine „Internationale Conference on Computer Vision[6]“. Bei dieser Konferenz gibt es oft Vorträge zu Methoden und Algorithmen die bei der 3D-Rekonstrukrion verwendbar sind. Einige dieser Algorithmen, wie zum Beispiel [Ethan Rublee, 2011], [M. Calonder u. Fua., 2010] und [Rosten u. Drummond, 2006], sind für diese Arbeit verwendet worden.

[4] http://www.photomodeler.com/
[5] Institute of Electrical and Electronics Engineers. Hompage http://www.ieee.org/index.html
[6] http://www.icccv.org/

3 Grundlagen

In diesem Kapitel wird das Konzept der 3D Rekonstruktion erläutert. Es gibt unterschiedliche Techniken für Rekonstruktion der 3D Modelle. Natürlich sind nicht alle für Smartphones geeignet. Dafür gibt es mehrere Gründe: einige Verfahren brauchen bestimmte Vorkenntnisse über den modellierten Gegenstand, einige sind nur mit zusätzlicher Hardware realisierbar. Ziele dieses Kapitels sind Vor- und Nachteile der meist in der Praxis benutzten 3D-Rekonstruktionsverfahren zu entdecken und eine Wahl eines geeigneten Algorithmusses anhand der Anforderungen zu treffen.

3.1 Stereorekonstruktion

Stereo-Korrespondenz ist ein Verfahren bei dem aus mindestens zwei Bildern ein 3D-Modell geschätzt wird. Dabei wird versucht korrespondierende Punkte auf den Bildern zu finden. Sobald deren Lage bekannt ist, ist es möglich 3D-Tiefen (depth map) zu bestimmen. Das Grundprinzip dieses Verfahrens ist in der Abbildung 3.1 skizziert. Sei P = (X, Y, Z) ein Weltpunkt im dreidimensionalen Raum der von zwei Kameras beobachtet wird. Falls alle Kameraparameter (innere und äußere, sieh Kamerakalibrierung) bekannt sind, kann man dessen Lage im Koordinatensystem der Kamera für jedes Bild bestimmen. Mithilfe dieser Information ist es dann möglich die 3D-Koordinaten von P im Weltkoordinatensystem zu berechnen.

Die Suche nach korrespondierenden Punkten bringt ein Problem mit sich: wenn man für jeden Pixel aus Bild 1 nach einem ähnlichen Pixel aus Bild 2 sucht, findet man oft mehrere „passende“ Korrespondenzpunkte. Dies gilt insbesondere für großflächige einfarbige Bildbereiche. Dieses Problem ist aber dadurch lösbar, dass man nicht für alle Pixel nach deren Korrespondenzpunkten sucht, sondern nur für diejenigen, für die es am leichtesten ist den eindeutigen zu finden. Dieses Verfahren, erweitert auf mehrere Bilder, ist sehr wichtig für diese Arbeit.

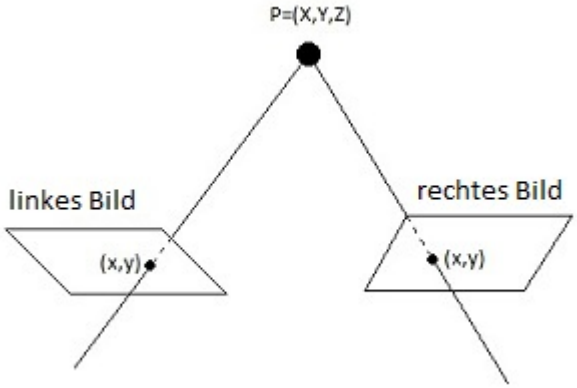

Abbildung 3.1: Grundprinzip der Stereorekonstruktion

3.2 Modellbasierte Rekonstruktion

Wenn man vorher etwas mehr über die zu modellierenden Objekte weiß, kann man mithilfe spezieller Techniken und Darstellungen detailliertere und zuverlässige 3D-Modelle erstellen. Man kann sich z. B. zunutze machen, dass die Architektur oft aus großen planaren Flächen und anderen parametrischen Formen, die in der Regel senkrecht zur Schwerkraft und zueinander sind, besteht. Das 3D-Modellierungssystem, das von Shum, Han und Szeliski 1998 entwickelt wurde, konstruiert zunächst kalibrierte Panoramen aus mehreren Bildern und lässt danach den Benutzer horizontale und vertikale Linien in das Bild zeichnen, um die Grenzen der planaren Regionen zu markieren. Die Linien werden zunächst zum Bestimmen der absoluten Rotation für jedes Panorama und dann zur Optimierung der 3D-Struktur verwendet, die aus einem bis mehreren Bildern wiederhergestellt werden kann (Abbildung 3.2).

360° High Dynamic Range Panoramen können auch für 3D-Modellierung im Freien verwendet werden, da sie sehr zuverlässige Schätzungen der relativen Kameraausrichtung sowie der Fluchtpunktrichtungen bieten. Während frühere bildbasierte Modellierungssysteme Benutzerinteraktionen erfordern, präsentieren Zisserman und Werner 2002 ein vollautomatisiertes linienbasiertes Rekonstruktionssystem. Es erkennt zunächst Linien und Fluchtpunkte und benutzt diese, um die Kamera zu kalibrieren, dann werden die Korrespondenzlinien mithilfe von Matching und trifokalen Tensoren bestimmt, dies erlaubt 3D-Liniensegmente, wie in Abbildung 3.3 gezeigt, zu rekonstruieren.

Ein weiterer Bereich, in dem speziell Form und Erscheinung der Modelle sehr hilfreich sein kann, ist die Modellierung der Köpfe und Gesichter. Auch wenn die Erscheinung eines Menschen auf den ersten Blick sehr variabel aussieht, kann die tatsächliche Form des Kopfes

Abbildung 3.2: Interaktive 3D-Modellierung von Panoramen. Quelle: [Shum, Han, and Szeliski, 1998]

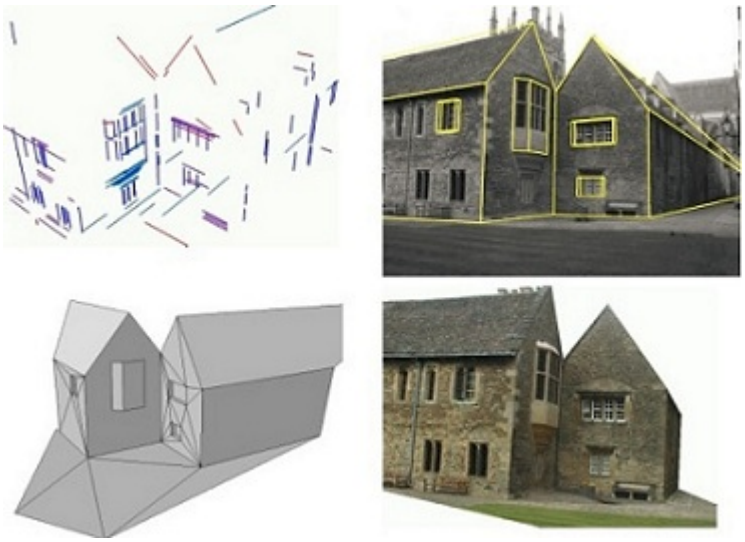

Abbildung 3.3: Automatische Architekturrekonstruktion mit Hilfe von 3D-Linien und Ebenen. Quelle: [Werner and Zisserman, 2002]

und des Gesichtes mit wenigen Parametern vernünftig beschrieben werden. Abbildung 3.4 zeigt ein Beispiel eines bildbasierten Modellierungssystems, in dem vom Benutzer angegebene

Schlüsselpunkte in mehreren Bildern dazu verwendet werden ein allgemeines Kopfmodell an das Gesicht einer Person anzupassen.

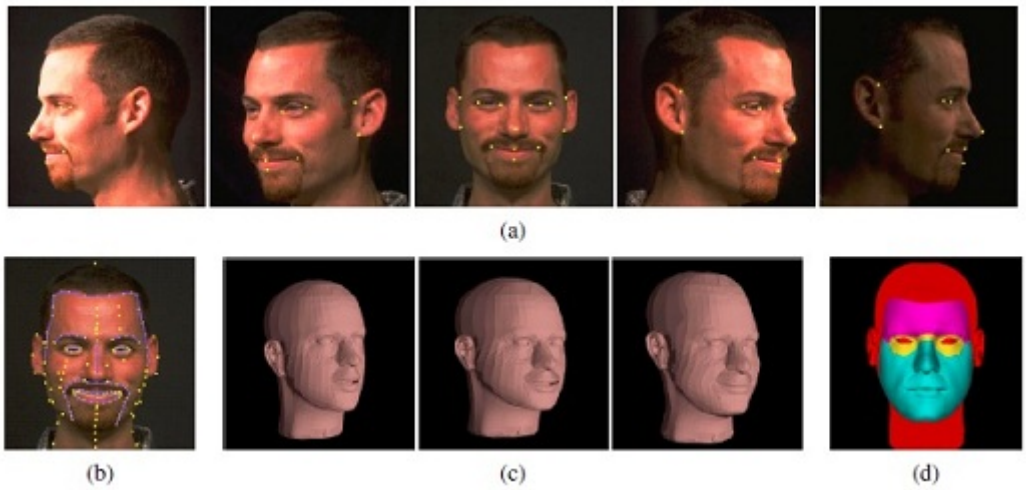

Abbildung 3.4: Übereinstimmung des 3D-Modells mit der Bildsequenz. Quelle: [Pighin, Hecker, Lischinski, 1998]

Wie man in Abbildung 3.4c sieht, ist das Gesicht, nach Angabe von knapp über 100 Schlüsselpunkten, gut angepasst und erkennbar. Das Extrahieren der Textur aus den Originalbildern und das Anwenden dieser auf das Kopfmodell führt zu einem animierten Modell mit erstaunlicher Originaltreue. Durch die Anwendung von Principal Component Analysis (PCA) auf eine Sammlung von 3D gescannten Gesichtern kann ein leistungsstärkeres System gebaut werden. Wie man in Abbildung 3.5 sehen kann, ist es dann möglich, morphable 3D Modelle an einzelne Bilder anzupassen und sie für eine Vielzahl von Animationen und visuellen Effekten zu verwenden (Blanz und Vetter, 1999). Es ist auch möglich stereo matching Algorithmen zu entwerfen die direkt für die Kopfmodellparameter optimiert sind (Shan, Liu und Zhang 2001; Kang und Jones 2002) oder Echtzeitstereoausgabe mit aktiver Beleuchtung zu verwenden (Zhang, Snavely, Curless, 2004) (Quelle: [Szeliski, 2011]).

Alle modellbasierte Verfahren setzten eine Benutzerinteraktion oder Vorkenntnisse über das zu modellierende Objekt voraus. Sie sind zwar für Architektur-, Gesichts- oder Körperrekonstruktion geeignet, genügen unseren Anforderungen aber nicht. Da es in unserem Fall um eine Anwendung für Smartphones geht, die ein Modell von einem beliebigen Objekt ohne Benutzerunterstützung erstellen können soll, können wir solche Verfahren nicht als Basis nehmen.

Abbildung 3.5: Beispiel für 3D-Rekonstruktion mit Pose- und Beleuchtungsänderung. Quelle: [Blanz and Vetter, 1999]

3.3 Active rangefinding

Diese Methoden basieren auf der Nutzung zusätzlicher Beleuchtungsquellen. Eine der populärsten Sensoren mit aktiver Beleuchtung ist ein Laser- bzw. Lichtstreifensensor der einen Lichtstreifen über das Objekt bewegt und dessen Verformung von einem anderen Sichtpunkt beobachtet (Abbildung 3.6). Wenn der Lichtstreifen über das Objekt fällt, deformiert er sich entsprechend an der von ihm beleuchteten Oberfläche. Danach ist es leicht mithilfe der optischen Triangulation die 3D-Koordinaten aller von ihm beleuchteten Punkte zu schätzen.

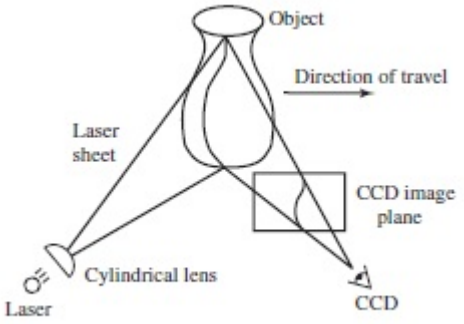

Abbildung 3.6: Ein Laserstreifen wird vom Sensor erfasst (die Verformung des Streifens codiert die Entfernung zum Objekt). Quelle: [Curless and Levoy, 1996]

Da alle solche Techniken zusätzliche Lichtquellen brauchen, können sie im Rahmen dieser Arbeit nicht berücksichtigt werden.

3.4 Shape from X

Es gibt verschiedene Verfahren die Oberflächeninformationen zu rekonstruieren. Eine Gruppe solcher Verfahren nennt man Shape from X. Dabei steht X für shading, focus, texture, motion usw.

3.4.1 Shape from Shading

Wenn man auf Bilder von glatten schattierten Objekten schaut, wie in der Abbildung 3.7 gezeigt, kann man allein durch die Variation der Schattierung die Form des Objektes klar erkennen.

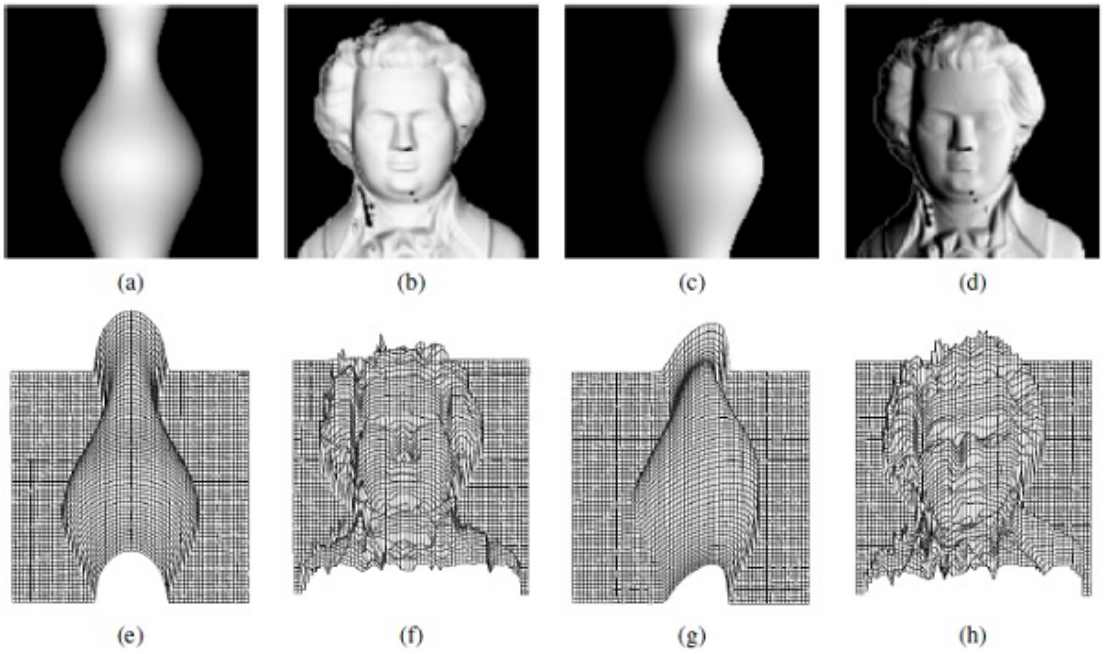

Abbildung 3.7: Rekonstruierte Form von Shading: (a-b) Licht vorne; (c-d) Licht vorn-rechts; (e-h) entsprechende Rekonstruktionen mit Verfahren von Tsai und Shah (1994). Quelle: [Zhang, Tsai, Cryer, 1999]

Die Normale zur Oberfläche des Objektes ändert sich fließend, die Helligkeitsveränderung der Reflexion ist eine Winkelfunktion zwischen lokaler Ausrichtung der Oberfläche und der Lichtquelle (Abbildung 3.8).

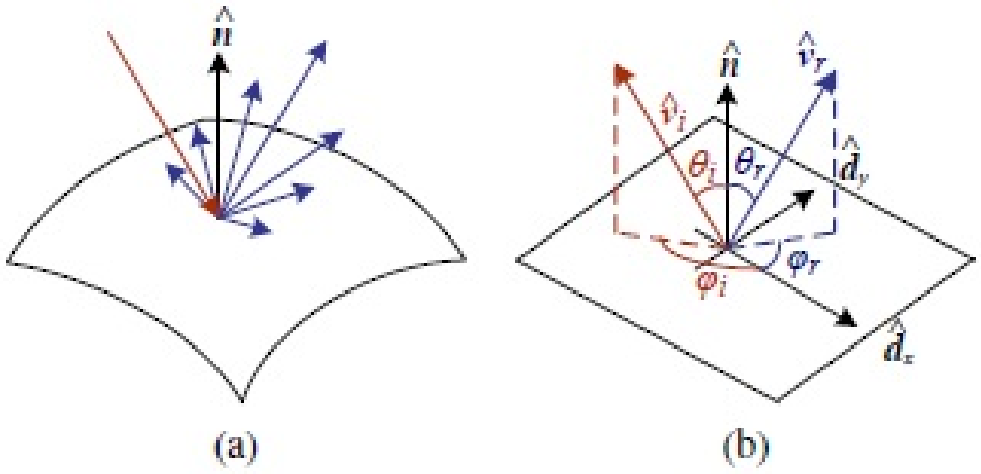

Abbildung 3.8: (a) Licht wird zerstreut, wenn es auf die Oberfläche trifft. (b) Bidirektionale Reflektanzverteilungsfunktion (BDRF) $f(\theta_i, \phi_i, \theta_r, \phi_r)$ ist von den Winkeln der einfallenden $\hat{\nu_i}$ und reflektierten $\hat{\nu_r}$ Strahlen mit dem lokalen Koordinatenabschnitt der Oberfläche $(\hat{d_x}, \hat{d_y}, \hat{n})$ parametrisiert. Quelle: [Szeliski, 2011]

Die meisten shape from shading Algorithmen gehen davon aus, dass das Rückstrahlvermögen sowie der Reflexionsgrad der betrachteten Oberfläche einheitlich sind und die Richtung der Beleuchtungsquelle entweder bekannt ist oder mit einem Referenzobjekt kalibriert werden kann. In der Praxis ist es schwierig shape from shading anzuwenden, weil verschiedene Oberflächen unterschiedliche Reflexionsvermögen haben, außerdem ist eine spezielle Lichtquelle notwendig, die ihre Strahlen parallel und mit konstanter Intensität ausstrahlt. Nützlich können shape from shading Algorithmen in Kombination mit anderen Methoden sein, z. B. stereo matching (Fua and Leclerc 1995) und known texture (White und Forsyth 2006) bieten Informationen in texturierten Regionen, während shape from shading in gleichmäßig gefärbten Regionen hilft und so Informationen über die Oberflächenform bietet.

3.4.2 Shape from Focus/Defocus

Shape from Focus oder Shape from Defocus ist das Problem der Schätzung von 3D-Objektoberflächen einer Szene, wenn zwei oder mehr Bildaufnahmen dieser Szene für geänderte Kameraparameter (wie fokale Länge oder Linsenöffnung) vorliegen. Aus der Gleichung

$$\frac{1}{u} = \frac{1}{f} - \frac{1}{v}$$

geht hervor, dass für eine gegebene Brennweite f und eine gegebene Kamerakonstante v nur solche Objektpunkte scharf abgebildet werden, die sich genau bei der Szenentiefe u

befinden. Bei einer unscharfen Abbildung eines Raumpunktes in die Bildebene ergibt sich ein Unschärfekreis mit dem Radius r (Abbildung 3.9) (Quelle: [Ebers, 2004]).

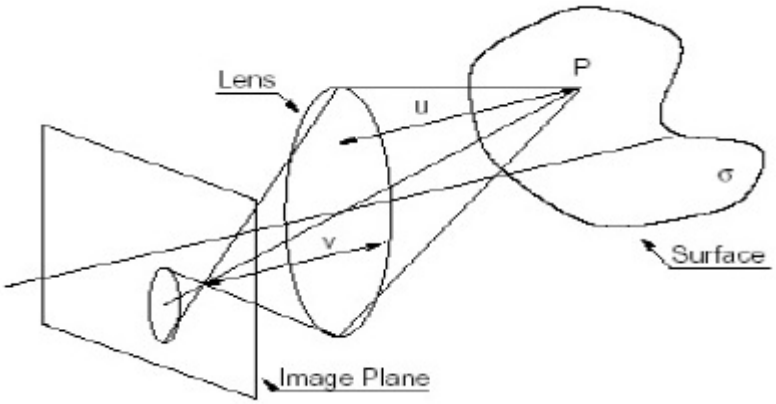

Abbildung 3.9: Linsenmodell. Quelle: [Favaro, P., Osher, S., Soatto, S. and Vese, L.: 3D Shape from Anisotropic Diffusion,2003, www.citeseer.com]

Der Vorteil der fokusbasierten Techniken liegt darin, dass man keine Korrespondenzen auf den Bildern suchen soll. Dabei ist aber zu beachten, dass die Größe des Objekts bei seiner Bewegung oder Veränderung des Brennpunkts variieren kann. Auch die Unschärfe vergrößert sich mit der Entfernung des Objekts vom Brennpunkt in beide Richtungen. Dies ist ein Grund dafür, zwei oder mehr Bilder mit verschiedener Brennweite zu verwenden, was die Berechnungszeit erhöht. Um die Unschärfe zuverlässig schätzen zu können, verwendet man am besten verschiedene rationale Filter. Die fokusbasierten Techniken sind also außerhalb eines Labors und ohne spezieller Hardware nur schwer anwendbar.

3.4.3 Shape from Textur

Wenn die genaue Form der elementaren Texturelemente (Texel) bekannt ist, kann man aus deren Deformierung und Größenveränderung nützliche Informationen über lokale Oberflächenorientierungen bekommen. Abbildung 3.10 zeigt ein Beispiel eines solchen Musters sowie geschätzte lokale Orientierung der Oberflächennormale.

Solche Algorithmen gehören zur strukturellen Texturanalyse und können nur auf deterministischen Texturen angewendet werden, die in der Natur fast nicht vorkommen. Die meisten natürlichen Texturen sind zwar statistisch regulär, können aber nicht so einfach in die Basismuster zerlegt werden, so dass nur eine statistische Texturanalyse für die Gewinnung der Tiefenformationen eingesetzt werden kann und zwar so, dass man Texturmerkmale wie

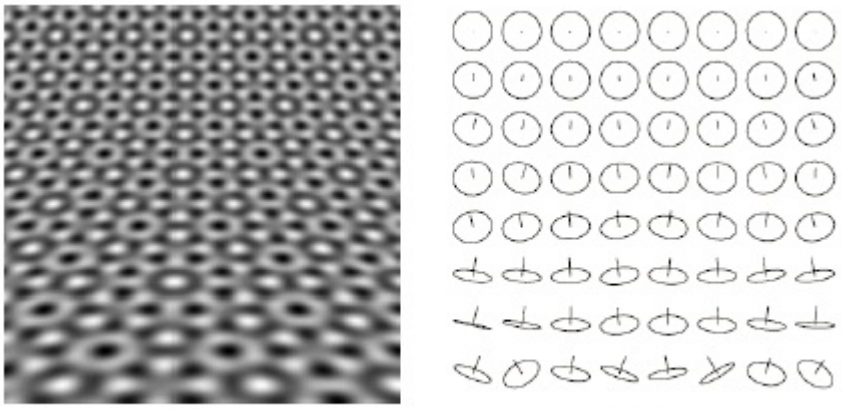

Abbildung 3.10: Rekonstruierte Form von Texture: a) Elementare Textur auf einer gekrümmten Oberfläche; b) Entsprechende Schätzungen der Oberflächennormale. Quelle: [Garding, 1992]

Gradient, Entropie, Korrelation, normales Histogramm oder das Richtungshistogramm usw. extrahiert und auswertet. Die existierenden Techniken in dem Bereich sind aber so komplex und gleichzeitig so unzuverlässig, dass die statistische Texturanalyse praktisch nur für die Mustererkennung oder Bildsegmentierung verwendet wird.

Eine weitere Möglichkeit ist die Analyse von eventuell vorhandenen geometrischen Beziehungen im Bild wie parallele Kanten oder symmetrische Körper usw. Die Texturanalyse erfordert einerseits keine Korrespondenzsuche und ist gegen die Beleuchtungsänderungen invariant, andererseits liefert sie im Allgemeinen (außer wenn die Texelgröße genau bekannt ist) nur Oberflächenorientierungen und keine absoluten Tiefeninformationen. Die Notwendigkeit von aufwendiger Bildsegmentierung und Texelsuche bzw. Merkmalextraktion schränkt ihre Einsatzmöglichkeiten erheblich ein, so dass zurzeit kaum ein schnelles oder genaues Verfahren existiert (Quelle: [Ebers, 2004]).

3.4.4 Shape from Motion

Es ist eine wichtige Aufgabe von Computer-Vision, Rekonstruktion von Geometrie der 3D-Szene und Kamerabewegung aus einer Reihe von Bildern einer statischen Szene zu ermöglichen. Die meisten Lösungen gelten unter besonderen Bedingungen. Allgemein nehmen vorhandene Techniken eine oder mehrere der folgenden Eigenschaften an:

- Bekannte Korrespondenz: Durch korrespondierende Punkte auf Bildsequenzen werden 3D-Positionen und Kamerabewegung ermittelt. Diese klassische Formulierung wurde von [Pollefeys, 1999](metrische Rekonstruktion), [Poelman u. Kanade, 1997] und anderen untersucht.

- Bekannte Kameras: Für 3D-Rekonstruktion sind kalibrierte Bilder von bekannten Kameraperspektiven gegeben. Für diese Kategorie sind Lösungen von [Faugeras, 1993] und [Kutulakos u. Seitz, 1999] passend.

- Bekannte Form: Es ist ein oder mehrere Bilder sowie ein 3D-Modell gegeben, es sind Kamerapositionen zu jedem Bild zu bestimmen. ([Lowe, May], [Huttenlocher u. Ullman., Nove])

Ein weiterer Unterschied der Lösungen ist die Art der Bewegung. Am schwersten ist der Fall, bei dem sich sowohl die Kamera als auch das Objekt bewegen. Im Rahmen dieser Arbeit wird auf die Fälle beschränkt, bei denen sich die Kamera bewegt und das Objekt statisch ist. Als Basis wird die Metrische Rekonstruktion genommen.

3.5 Metrische Rekonstruktion

Marc Pollefeys beschreibt in seiner Arbeit [Pollefeys, 1999] ein Verfahren für 3D-Rekonstruktion aus einer Sequenz von Bildern, die mit einer Handkamera aufgenommen wurden. Der Benutzer kann sich mit der Kamera frei rund um das Objekt bewegen. Weder die Kamerabewegung noch ihre Einstellungen müssen bekannt sein. Das erhaltene 3D-Modell ist eine skalierbare Version des ursprünglichen Objektes (d.h. Eine metrische Rekonstruktion), die Oberfächentextur wird ebenfalls aus der Bildsequenz bekannt. Das beschriebene System bekommt schrittweise mehr Informationen über die Szene und die Kameraeinstellungen.

Im ersten Schritt sind die Bilder in Zusammenhang zu bringen. Dies geschieht paarweise durch die Bestimmung von Epipolare Geometrie. Zuerst wird die Rekonstruktion für die ersten zwei Bilder der Sequenz durchgeführt. Für die nachfolgenden Bilder wird die Kameraposition im Projektionsframe der ersten beiden Kameras geschätzt. Für jedes weitere Bild werden in dieser Phase die Punkte, die von Interesse sind und mit den entsprechenden Punkten aus dem vorherigen Bild korrespondieren, rekonstruiert, verfeinert oder korrigiert. Das Ergebnis dieses Schrittes ist eine Rekonstruktion, typischerweise, einiger hundert bis einiger tausend Punkten sowie die (projektive) Position der Kamera.

Der nächste Schritt ist die Beschränkung der Mehrdeutigkeit der Rekonstruktion zu einer metrischen. An diesem Punkt verfügt das System über eine kalibrierte Bildsequenz. Die relative

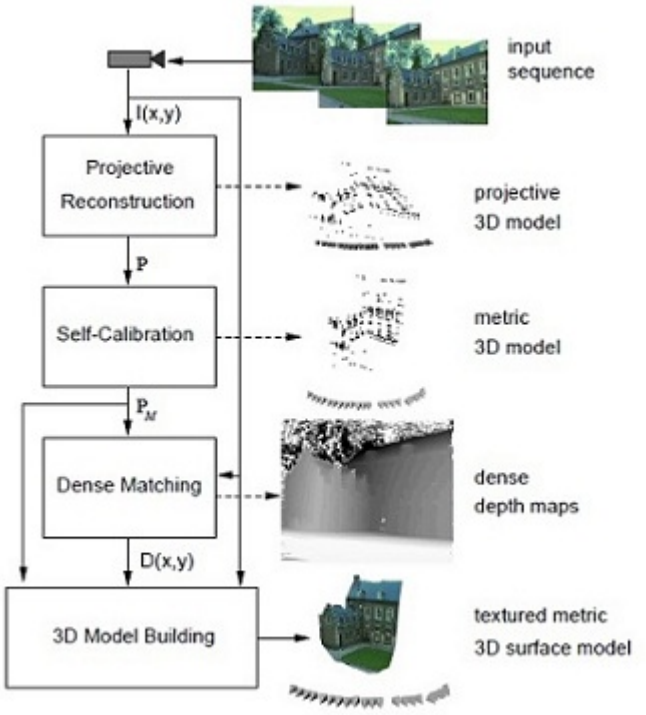

Abbildung 3.11: 3D-Rekonstruktion nach Pollefeys [Pollefeys, 1999]

Position und Ausrichtung der Kamera ist für alle Sichtpunkte bekannt. Dies erleichtert die Suche nach korrespondierenden Punkten und ermöglicht den Einsatz von stereo Algorithmen die für kalibrierte Systeme entworfen wurden und korrespondierende Punkte für die meisten Pixel im Bild finden. Durch die Triangulation kann aus diesen Korrespondenzen der Abstand der Punkte von dem Zentrum der Kamera berechnet werden. Die Ergebnisse werden durch Kombination der Korrespondenzen aus mehreren Bildern verfeinert und ergänzt. Feine metrische Oberfläche des 3D-Modells bekommt man durch die Approximation[1] der Tiefenmap mit dreieckigem Drahtgittermodell. Die Textur wird von den Bildern extrahiert und auf die Oberfläche abgebildet.

Detaillierte Beschreibung aller Schritte wird im nächsten Kapitel vorgestellt.

[1] Approximation (Synonym für Näherung) ist eine wissenschaftliche Methode, die darin besteht, den Ersatz einiger Objekte durch ähnliche aber einfachste Objekte

3.6 Zusammenfassung

In diesem Kapitel wurden die meisten für 3D-Rekonstruktion verwendeten Methoden untersucht. Einige, wie zum Beispiel „active rangefinding“, brauchen zusätzliche Hardware, andere, wie modellbasierte Verfahren, können nur mit einigen Vorkenntnissen über das zu rekonstruierende Objekt gute Ergebnisse liefern. Da es in dieser Bachelorarbeit um eine Anwendung für Smartphones geht, was Mobilität und Flexibilität bedeutet, ist wohl die metrische Rekonstruktion für diese Arbeit als die am besten passende anzusehen.

4 Analyse

Das Ziel dieses Kapitels ist es den Lösungsweg zu bestimmen. Hier werden das grundlegende Kameramodell sowie die epipolare Geometrie und die Triangulation erläutert. Es wird auch erklärt was Kamerakalibrierung ist und wofür sie dient. Außerdem werden verschiedene Verfahren wie zum Beispiel der SIFT-Algorithmus vorgestellt, die notwendig sind, um korrespondierende Punkte auf Bildern zu bestimmen.

4.1 Kameramodell

Alle Smartphone-Kameras basieren auf dem in Abbildung 4.1 vorgestellten Kameramodell. Aus diesem Modell lassen sich die Punktkoordinaten auf dem Bild wie folgt berechnen:

$$x = f\frac{X}{Z} \tag{4.1}$$

$$y = f\frac{Y}{Z} \tag{4.2}$$

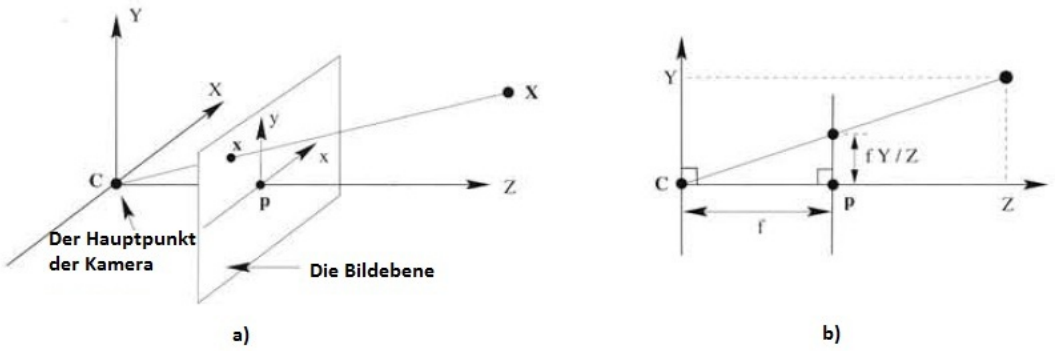

Abbildung 4.1: Kameramodell: a) C ist der Hauptpunkt der Kamera und P ist der Hauptpunkt der Projektion. X ist ein Punkt im Weltraum und x seine Abbildung auf der Bildebene. b) f ist die Brennweite. Quelle: [Hartley u. Zisserman, 1997] s. 154

Um weitere Berechnungen vereinfachen zu können ist es notwendig nicht-lineare Abbildungen für x und y in lineare Form zu bringen. Um dies zu ermöglichen werden die Punktkoordinaten auf dem Bild mithilfe von Homogenen Koordinaten[1] dargestellt. Die Transformation von kartesischen Koordinaten in homogene und zurück ist in Abbildungen 4.2 und 4.3 veranschaulicht.

$$(x, y) \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \qquad (x, y, z) \Rightarrow \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Abbildung 4.2: Transformation vom kartesischen in das homogene Koordinatensystem. Quelle: [Konushin, 2011]

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} \Rightarrow (x/w, y/w) \qquad \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \Rightarrow (x/w, y/w, z/w)$$

Abbildung 4.3: Transformation von homogene Koordinaten zu kartesisches Koordinatensystem. Quelle: [Konushin, 2011]

Eine Abbildung kann man mathematisch mithilfe einer Projektionsmatrix im homogenen Koordinatensystem beschreiben. Diese spezielle Matrix beschreibt eine perspektivische Abbildung eines dreidimensionalen Objektpunktes auf dem zweidimensionalem Bild (Abbildung 4.4).

Matrix K ist eine Kalibrierungsmatrix, diese sieht im Koordinatensystem der Kamera wie folgt aus:

[1] In der Geometrie erlauben homogene Koordinaten die Darstellung von Transformationen durch eine Multiplikation der Koordinaten mit Matrizen in einfacher Weise.

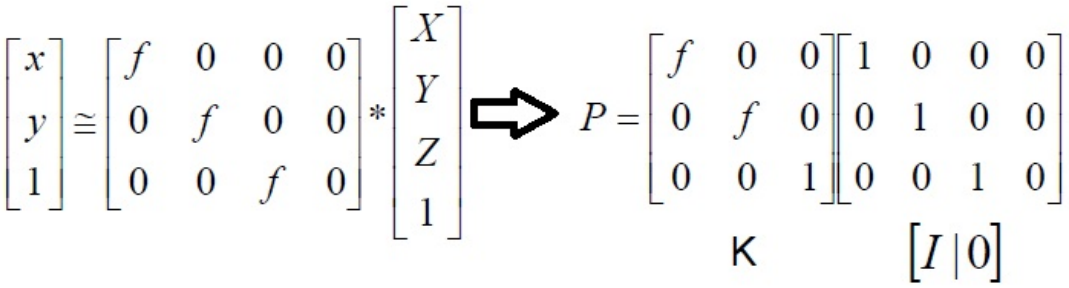

Abbildung 4.4: Projektionsmatrix P. Quelle: [Konushin, 2011]

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix},$$

dabei sind c_x und c_y Koordinaten des Kamerahauptpunktes.

Transformation vom Welt- in das Kamerakoordinatensystem wird mithilfe der Matrix

$$C = \begin{bmatrix} R & T \\ [0,0,0] & 1 \end{bmatrix}$$

beschrieben (Abbildung 4.5). Die Matrix C^{-1} wird äußere Kalibrierung genannt, R ist dabei eine Rotationsmatrix und T der Translationsvektor.

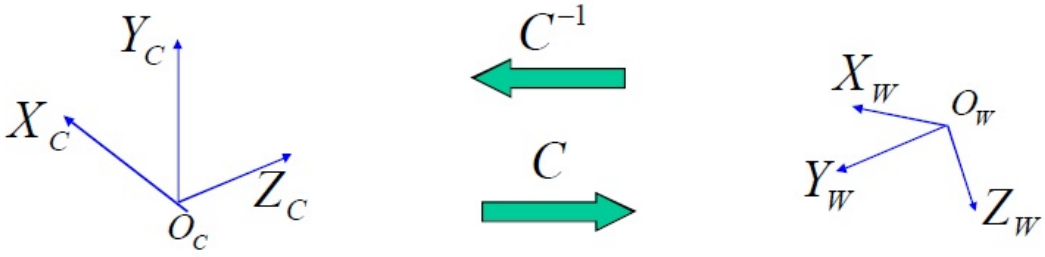

Abbildung 4.5: Transformation vom Welt- in das Kamerakoordinatensystem (C) und zurück (C^{-1}). Quelle: [Konushin, 2011]

Die Projektionsmatrix kann man also in folgender Form schreiben:

$$P = K[I|0]C^{-1} \tag{4.3}$$

Nächster Abschnitt befasst sich mit der Berechnung der Kameramatrix K und äußerer Parametern R und T.

4.2 Kamerakalibrierung

Unter einer vollständigen Kamerakalibrierung versteht man die Bestimmung der Abbildungseigenschaften einer Kamera (innere Orientierung) sowie die Orientierung des Weltkoordinatensystems relativ zum Kamerastandpunkt (äußere Orientierung):

- Äußere Parameter: Beschreiben die Position und die Orientierung der Kamera im Raum: R ist eine Drehmatrix und T ist ein Translationsvektor

$$C^{-1} = \begin{bmatrix} R^T & -R^T T \\ [0,0,0] & 1 \end{bmatrix}$$

- Innere Parameter: Parameter der inneren Orientierung, z.B. Brennweite (f_x, f_y) der Kamera und die Position des Bildhauptpunktes (c_x, c_y).

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

Es gibt zwei Möglichkeiten für die Kamerakalibrierung: Mit einem Kalibrierungsobjekt und ohne (automatische Kalibrierung).

4.2.1 Automatische Kalibrierung

Automatische (oder Selbst-)Kalibrierung ist die Berechnung der metrischen Eigenschaften einer Kamera aus einer Menge von nicht kalibrierten Bildern. Sobald dies geschehen ist, ist es möglich mittels metrischer Rekonstruktion 3D-Modelle aus den Bildern zu berechnen. Automatische Kalibrierung vermeidet die lästige Aufgabe die Kameras mit einem Kalibrierungsobjekt kalibrieren zu müssen. Dies sorgt für hohe Flexibilität, da zum Beispiel eine Kamera direkt aus der Bildsequenz trotz der unbekannten Bewegung und Änderungen einiger innerer Parameter kalibriert werden kann. Der Kern dieser Methode liegt in einer bestimmten Bewegungsbahn der Kamera die einen Kegelschnitt beschreibt. In [Hartley u. Zisserman, 1997] sind verschiedene Techniken für Selbstkalibrierung und Problematik von solchen Methoden beschrieben.

Automatische Kalibrierung kann unter den richtigen Umständen gut funktionieren, wenn man sie aber rücksichtslos verwendet, wird sie scheitern. Es gibt mehrere Empfehlungen dazu:

- Man sollte mehrdeutige Bewegungsabläufe vermeiden. Die Bewegung sollte nicht zu klein sein oder ein zu kleines Sichtfeld abdecken, weil die automatische Kalibrierung oft zur Schätzung einer unendlichen Homographie kommt, deren Auswirkungen auf kleineren Sichtfeldern nicht offensichtlich sind.

- Man soll möglichst viele Informationen verwenden. Zum Beispiel bekanntes Seitenverhältnis sowie der Bildmittelpunkt. Auch wenn die Werte ungenau sind können sie in die Berechnung mit aufgenommen werden, jedoch mit geringerem Gewicht.

- Methoden, die auf eingeschränkten Bewegungen basieren, sind in der Regel zuverlässiger als diejenigen, die allgemeine Bewegungen zulassen.

In dieser Arbeit geht es um eine Anwendung für Smartphones, dies bedeutet in der Regel chaotische Bildaufnahmen. Für die 3D-Rekonstruktion ist es sehr wichtig genaue Kameraparameter zu bekommen, deswegen können wir keine der automatischen Verfahren nutzen, da sie unsicher sind.

4.2.2 Kalibrierung mit Hilfe einen Kalibrierungsobjekt

Es gibt viele verschiedene Verfahren für Kamerakalibrierung mithilfe eines Kalibrierungsobjekts. Im Rahmen dieser Arbeit wird es mit dem Verfahren von Zhang befasst. [Zhang, 2000]. Bei der Entwicklung dieses Verfahrens wurde an die Flexibilität, Robustheit und niedrige Hardwarekosten gedacht. Die Hauptidee liegt darin, dass man mit der zu kalibrierenden Kamera mindestens zwei Aufnahmen einer planaren Struktur aus unterschiedlichen Sichtpunkten macht. Das Muster (für diese Arbeit siehe Abbildung 4.6) kann auf einem Laserdrucker ausgedruckt und an einer starren flachen Oberfläche befestigt werden. Entweder die Kamera oder das flache Muster kann zwischen den Aufnahmen beliebig bewegt werden.

Für die Kalibrierung sind also folgende Schritte erforderlich:

- Ausdruck des Musters.

- Aufnahme mehrerer Bilder aus verschiedenen Perspektiven.

- Ermittlung der Merkmalpunkte in den Bildern.

- Einschätzung von fünf intrinsischen und allen äußeren Parametern.

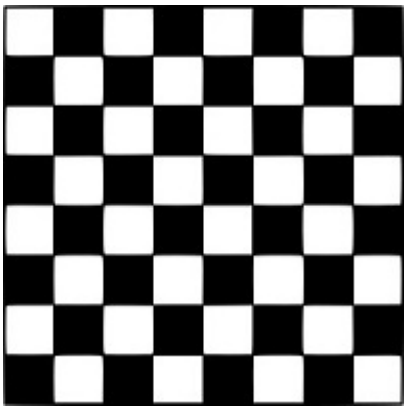

Abbildung 4.6: Kalibrierungsmuster

- Einschätzung der Koeffizienten der radialen Verzerrung.

Im Vergleich zur Selbstkalibrierung gewinnt diese Technik ein erhebliches Maß an Robustheit. Aus der Kamerakalibrierung werden in dieser Arbeit nur innere Parameter verwendet.

4.3 Epipolare Geometrie

Wenn man geometrische Beziehungen zwischen zwei oder mehreren Bildern, die das gleiche Objekt abbilden, beschreiben will, nutzt man die Epipolare Geometrie. Es ist möglich, ohne dass Kamerapositionen bekannt sind, Beziehungen zwischen korrespondierenden Punkten herzustellen.

Auf zwei Bildern x und x' ist ein Weltpunkt X abgebildet (Abbildung 4.7a). Die Projektionszentren C und C' liegen zusammen mit x, x' und X auf einer Ebene π. Die Linie CC', die Basislinie heißt, erzeugt mit den Bildebenen die Schnittpunkte e und e', die Epipole heißen. In der Abbildung 4.7b sind die Geraden l und l'- Epipolarlinien, die Schnittgeraden zwischen der Ebene π und den jeweiligen Bildebenen sind.

Die Fundamentalmatrix F enthält die gesamte Information über die Epipolargeometrie. Sie beschreibt sowohl die Beziehung zwischen dem Bildpunkt und Epipolarlinie als auch die Beziehungen zwischen den Bildpunkten.

$$l = F^T x \tag{4.4}$$

$$l' = F x' \tag{4.5}$$

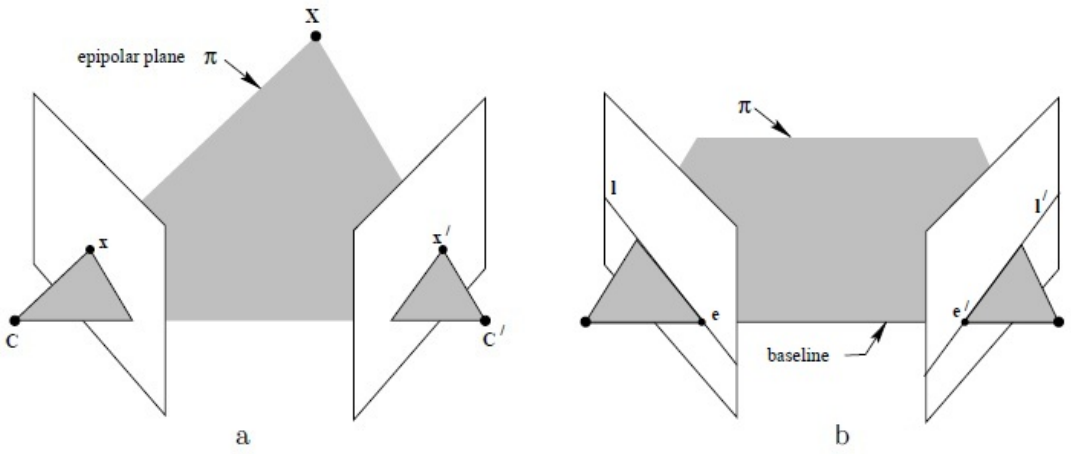

Abbildung 4.7: Geometrie von Korrespondierenden Punkten [Hartley u. Zisserman, 1997]

$$x'^T F x = 0 \tag{4.6}$$

Aus der Fundamentalmatrix kann man die essentielle Matrix E berechnen. Diese beschreibt die äußeren Parametern wie Rotation und Translation von beiden Kameras (K und K'):

$$E = [t]_x R \tag{4.7}$$

Der Zusammenhang zwischen der essentiellen und der Fundamentalmatrix ist über die inneren Parameter der beiden Kameras K und K' gegeben:

$$E = K^T F K' \tag{4.8}$$

Jetzt kann man die restlichen unbekannten äußeren Parameter aus der essentiellen Matrix berechnen. Im ersten Schritt soll die Singularwertzerlegung[2] von der essentiellen Matrix durchgeführt werden [Pollefeys, 1999].

$$E = UZV^T \tag{4.9}$$

[2]Die Singulärwertzerlegung (Abk.: SVD für Singular Value Decomposition) einer Matrix bezeichnet deren Darstellung als Produkt dreier spezieller Matrizen. Daraus kann man die Singulärwerte der Matrix ablesen.

Dann, kann man mithilfe der Matrizen

$$Z = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

und

$$W = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

den Translationsvektor sowie die Rotationsmatrix bestimmen:

$$R = UWV^T oder UW^TV^T \tag{4.10}$$

und

$$t = U(001)^T = u_3 \tag{4.11}$$

Man kann jetzt die Projektionsmatrizen (oder Kameramatrizen) für die beiden Kameras bestimmen. Die erste Kameramatrix hat die Form

$$P = [I|0] \tag{4.12}$$

und vier mögliche Lösungen für die Zweite Kameramatrix (Abbildung 4.8) sind

$$P' = [UWV^T| + u_3], [UWV^T| - u_3], [UW^TV^T| + u_3] oder [UW^TV^T| - u_3] \tag{4.13}$$

Die richtige Lösung für die zweite Kamera kann man mithilfe der Triangulation bestimmen: Bestimmte Punkte sollen sich vor der Bildebene befinden.

4.4 Triangulation

Nach dem die Projektionsmatrizen bestimmt wurden, kann man mithilfe der Triangulation die 3D-Koordinaten der Bildpunkte berechnen. In [Hartley u. Sturm, 1997] sind mehrere Verfahren für die Triangulation beschrieben und analysiert. Die einfachste und populärste Methode ist die lineare Triangulation. Sei X ein auf den Bildern x und x' abgebildeter Punkt. Dann gilt $x = PX$ und $x' = P'X$. Die ersten drei Elemente jeder Zeile von P werden zu einem Vektor

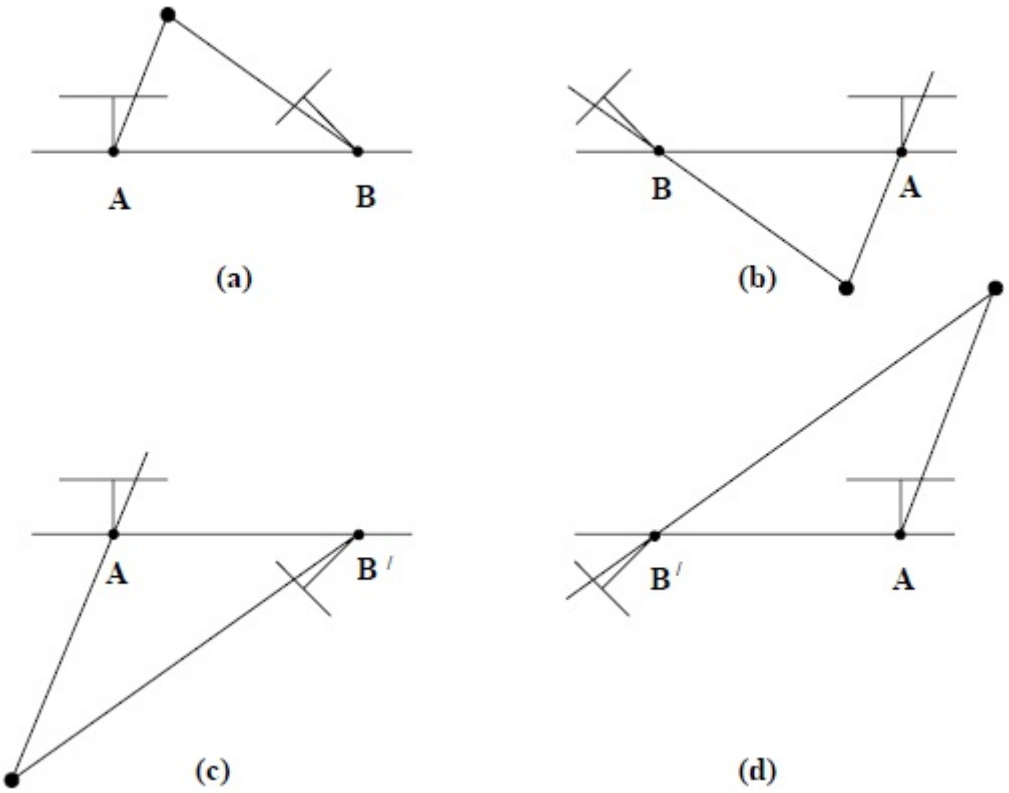

Abbildung 4.8: Vier mögliche Lösungen für die Projektionsmatrix der zweiten Kamera

$p_j = (p_{j1}, p_{j2}, p_{j3})$ zusammengefasst. Um die Koordinaten von X berechnen zu können, soll man die Gleichung der Form $AX = 0$ lösen, wo

$$A = \begin{bmatrix} x_x p_3 & -p_1 \\ x_y p_3 & -p_2 \\ x'_x p'_3 & -p'_1 \\ x'_y p'_3 & -p'_2 \end{bmatrix}$$

4.5 Features Detection

Es ist nicht möglich jedes Pixel eines Bildes mit jedem Pixel eines anderen zu vergleichen, deswegen ist es notwendig die kombinatorische Komplexität zu reduzieren. Außerdem sind nicht alle Punkte für automatische Zuordnung geeignet. Die Umgebungen einiger Punkte können möglicherweise starke Schwankungen der Intensität haben. Solche Punkte, die so genannten Interest-Operatoren, sind daher leicht von den anderen zu unterscheiden. Es gibt

viele verschiedene Verfahren für die Merkmalsuche die man verwenden kann. Diese Arbeit beschränkt sich auf drei davon.

4.5.1 Harris corner detector

Einer der mehreren Punktdetektoren, der bessere Lösungen liefert, ist Harris corner detection [Harris u. Stephens, 1988]. Die Hauptidee dieses Verfahrens ist die Suche nach Ecken (Abbildung 4.9). Wenn man ein Bild durch ein kleines „Fenster“ betrachtet, also bis auf einen kleinen Ausschnitt abdeckt, so stellt man fest, dass dieses Fenster oft nur ein Stück weit verschoben werden kann, ohne dass es zu nennenswerten Änderungen des Inhalts kommt. Dies funktioniert solange sich nicht eine Kante oder Ecke im Fenster befindet. Ist letzteres der Fall, führt eine kleine Bewegung des Fensters in jede beliebige Richtung sofort zu einer Änderung dessen Inhalts [Erschenburg, 2006]. Dieses Verfahren ist in OpenCV in der Funktion cornerHarris() realisiert und wird in dieser Arbeit untersucht.

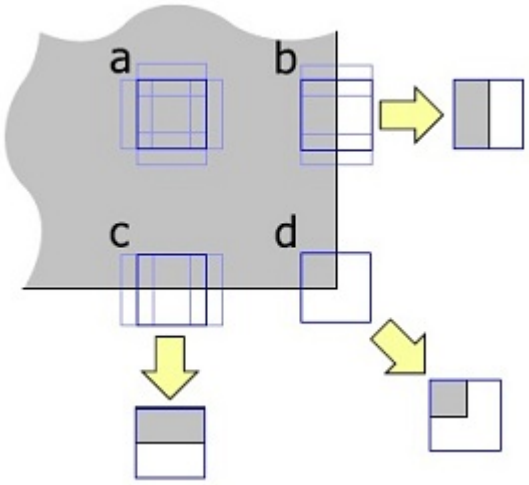

Abbildung 4.9: Das Prinzip der Harris corner detection. Das Fenster a liegt weder auf einer Kante, noch auf einer Ecke. Es kann in beide Richtungen verschoben werden, ohne dass sich der Inhalt ändert. Die Fenster b und c liegen auf einer Kante. Sie lassen sich nur noch in eine Richtung bewegen, ohne das sich ihr Inhalt ändert. Fenster d liegt auf einer Ecke und lässt sich nicht bewegen, ohne das sich dessen Inhalt ändert. [Erschenburg, 2006]

4.5.2 SIFT und SURF

Scale-invariant features transform (SIFT) ist ein weiterer Algorithmus zur Extraktion der Bildmerkmale. Der Algorithmus wurde von [Lowe, 2004] im Jare 1999 veröffentlicht. Der erste Schritt ist die Ermittlung potentieller Merkmale durch Berechnung der Extrema an den mit Gaus-Algorithmus geglätteten Bildern in verschiedenen Skalierungen. Als nächstes wird jedem gefundenen Punkt eine Skalierung und Position zugeordnet – Lokalisierung der Merkmalpunkte. Die Bestimmung der Orientierung ist der dritte Schritt des Algorithmus, dieser dient dazu die Merkmalpunkte gegenüber der Rotationen invariant zu machen. Deren endgültige Bestimmung wird im vierten Schritt durchgeführt. Die Umgebung jedes gefundenen Punktes wird untersucht um die Invarianz gegenüber den Helligkeitsveränderungen sowie den Verformungen zu erhöhen.

In dieser Arbeit wird der SURF (speeded up robust features)-Algorithmus untersucht. Der SURF Algorithmus ersetzt die im SIFT verwendeten Gaus-Filter durch Mittelwertfilter um die Berechnung zu beschleunigen.

4.6 RANSAC-Algorithmus

Oft ist es notwendig aus einer Datenmenge ein Modell zu bestimmen. RANSAC-Algorithmus (Random Sample Consensus, deutsch etwa „Übereinstimmung mit einer zufälligen Stichprobe“) hilft dabei die Werte, die nicht in die erwartete Messreihe passen, weg zu lassen. Der Algorithmus hat folgende Eingabedaten:

- Ausgangswerte.
- Funktion zum Berechnen der Modellparameter anhand der gegebenen Punkte
- Funktion zum Bewerten der Gültigkeit der Modellpunkte.
- Grenzwert für Bewertungsfunktion.
- Anzahl der Iterationen.

Der Algorithmus besteht aus einem Zyklus dessen Iterationen jeweils in zwei logischen Stufen unterteilt werden kann:

- Die erste Stufe ist die Auswahl der Punkte sowie die Berechnung des Modells. Aus einer Menge der Ausgangspunkte werden zufällig n verschiedene ausgewählt. Anhand dieser Punkte werden Modellparameter berechnet. Das resultierende Modell wird Hypothese genannt.

- Die zweite Stufe ist die Überprüfung der Hypothese. Jeder Punkt wird auf die Übereinstimmung mit der Hypothese bewertet und als brauchbar bzw. ausfallender markiert. Nach dem alle Punkte überprüft wurden, wird die Hypothese mit der zuletzt besten verglichen, die beste davon wird behalten.

4.7 Zusammenfassung

Um eine 3D-Rekonstruktion durchführen zu können, braucht man eine Kameramatrix die durch Kamerakalibrierung berechnet wird und die inneren Parameter der Kamera beschreibt. Danach werden korrespondierende Punkte auf zwei Bildern gesucht und die Fundamentalmatrix berechnet. Im nächsten Schritt werden die essentielle und anschließend die Projektionsmatrizen für die beiden Kameras bestimmt. Sobald alles über die Szenengeometrie bekannt ist, kann man mittels Triangulation die 3D-Koordinaten der Punkte im Weltkoordinatensystem berechnen. Die Ergebnisse für die ersten zwei (dann drei, vier usw.) Bilder fließen immer in die Berechnung für das nächste Bild ein, bis alle Bilder bearbeitet wurden.

5 Laufzeitumgebung und Bibliotheken

In Rahmen dieser Arbeit wird eine Applikation erarbeitet für Androidunterstützte mobile Geräte mit Hilfe von OpenCV Bibliothek. In diesem Kapitel wird Android als Betriebssystem erläutert, sowie die Android-Laufzeitumgebung und die wichtigsten Android-Komponenten ausführlich beschrieben. Zusätzlich werden Vor- und Nachteile von Android-Plattform diskutiert.

Alle Schritte der Bildverarbeitung werden mittels Bibliothek OpenCV realisiert. Abschnitt 5.2 beschäftigt sich mit der Beschreibung von OpenCV und seiner Anwendung bei Android.

5.1 Android

5.1.1 Überblick und Systemaufbau

Android ist ein Betriebssystem für mobile Geräte wie Mobiltelefone, Smartphone, Netbooks, Tablets, Smartwatch. Android wurde von der Open Handset Alliance (Hauptmitglied: Google Inc.) entwickelt. Es wird über einen Touchscreen bedient und definiert eine Reihe von Hardwaretasten. Die Architektur von Android baut auf dem Linux-Kernel 2.6 auf, mit Android 4.X ist auch ein Kernel der 3.X Serie möglich.

Zur Geschichte der Android-Betribssysteme. Im Herbst 2003 wurde von Andy Rubin das Unternehmen Android gegründet. Das Unternehmen hat eine Software für Mobiltelefone entwickelt. Im Sommer 2005 wurde Android von Google Inc. gekauft. Im November 2007 hat die Entwicklung von Mobiltelefon-Betriebssystem namens Android durch Google Inc. und 33 anderen Mitgliedern der Open Handset Alliance begonnen. Seit dem 21. Oktober 2008 ist Android offiziell verfügbar. Im Juni 2012 erreicht Google 1 Million Aktivierungen von Android-Geräte pro Tag (im Juni 2011 waren es ca. 500000 Aktivierungen pro Tag).

Den Kern der Laufzeitumgebung bildet die Dalvik Virtual Machine (DVM). Wird eine Android-Anwendung gestartet, so läuft sie in einem eigenen Betriebssystemprozess. Außerdem ist die DVM so klein und performant, dass Android jeder Anwendung darüber hinaus noch eine eigene DVM spendiert. Dies kostet zwar extra Ressourcen, gibt aber erheblichen Auftrieb in puncto Sicherheit und Verfügbarkeit, da sich die Anwendungen keinen gemeinsamen

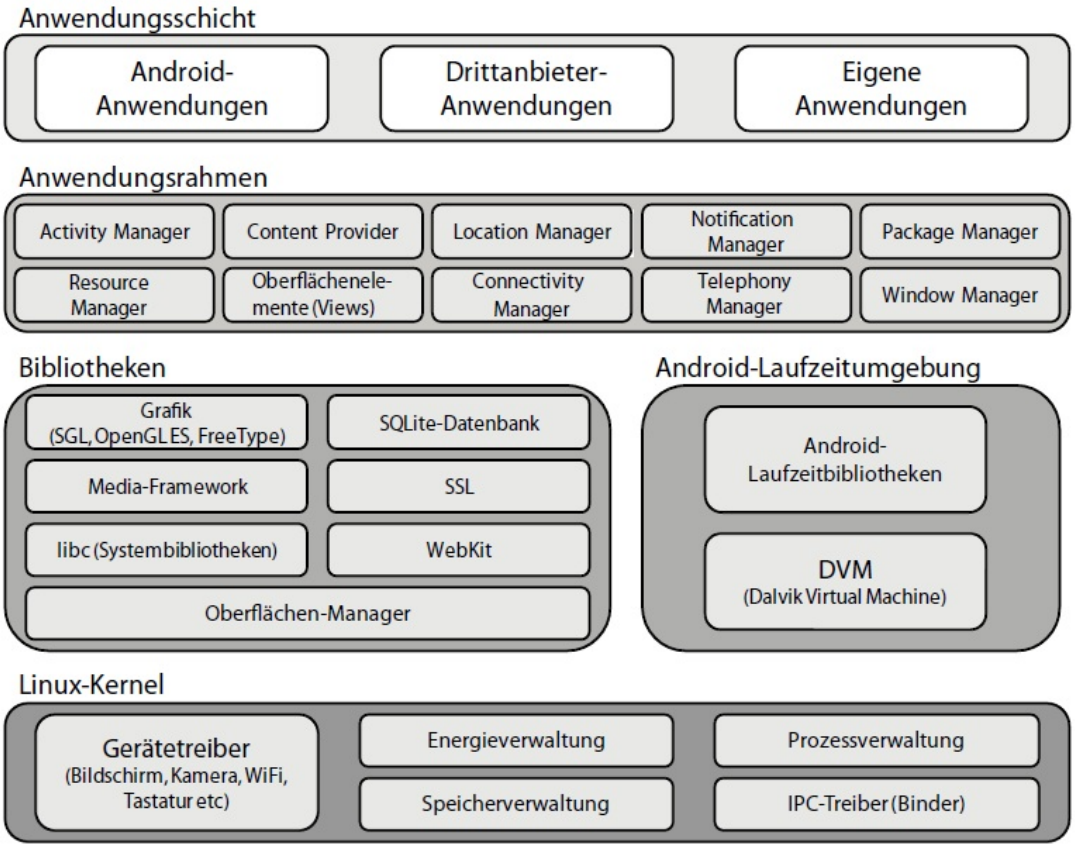

Abbildung 5.1: Android-Systemarchitektur. Quelle: [Arno Becker, 2009]

Speicher teilen und ein sterbender Prozess nur eine Anwendung mit in die Tiefe reißt. Die Anwendungen selbst werden in Java geschrieben und auch zur Entwicklungszeit von einem normalen Java-Compiler in Java-Bytecode übersetzt (Quelle: [Arno Becker, 2009] s. 15,16).

5.1.2 Dalvik virtual machine

Die Dalvik Virtual Machine wurde von einem Google-Mitarbeiter namens Dan Bornstein entwickelt. Benannt ist sie nach dem isländischen Ort Dalvík, in dem Verwandte von Bornstein lebten. Sie stellt das Herzstück der Android-Laufzeitumgebung dar und basiert auf der quelloffenen JVM Apache Harmony, wurde aber in Aufbau und Funktionsumfang an die Anforderungen mobiler Endgeräte angepasst.

Android lässt sich komplett in Java programmieren. Dies hat den großen Vorteil, dass vorhandenes Wissen genutzt und vorhandene Entwicklungsumgebungen weiterverwendet werden können. Es stellt sich nun die Frage, wie der Java-Code Schritt für Schritt in ablauffä-

higen Code transformiert wird und worin die Unterschiede zwischen DVM und JVM liegen. Das Schaubild in Abbildung 5.2 skizziert den Weg des Java-Codes von der Erstellung bis zur Ausführung im Android-Endgerät. Oberhalb der gestrichelten Linie findet die Entwicklung in der IDE auf dem PC statt. Der Pfeil nach unten deutet den Deployment-Prozess auf das Mobilgerät an.

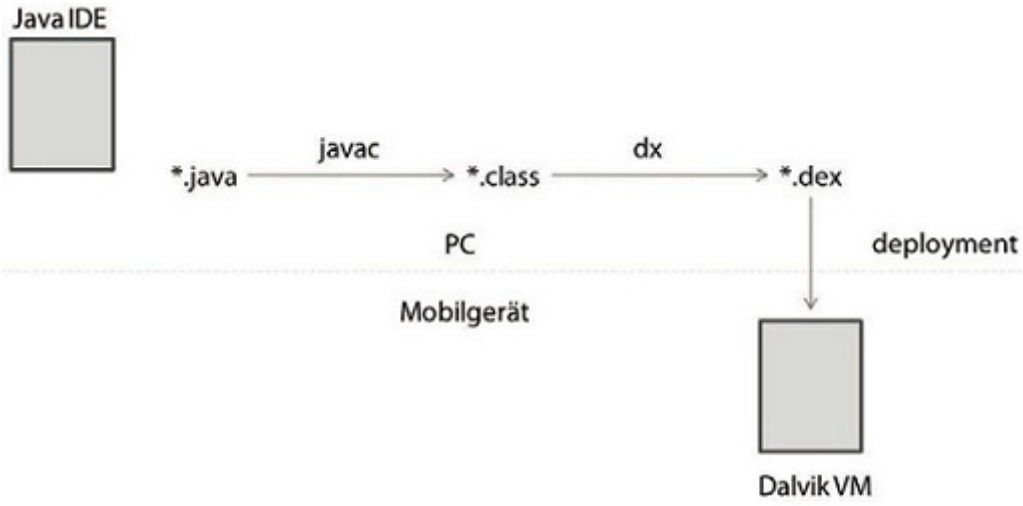

Abbildung 5.2: Weg von *.java zu *.dex. Quelle: [Arno Becker, 2009]

Klassische JVMs nutzen in ihrer virtuellen Prozessorarchitektur nicht aus, dass Mikroprozessoren Register haben. Register sind Zwischenspeicher direkt im Mikroprozessor, die Berechnungen über mehrere Zwischenergebnisse extrem schnell machen. Google hat mit der DVM eine Java-Laufzeitumgebung geschaffen, die Registermaschinencodes verarbeitet, also die Möglichkeiten moderner Prozessoren gut ausnutzt.

Da normalerweise pro Android-Anwendung ein eigener Betriebssystem-User verwendet wird (ein Prozess, ein User, eine DVM), sind gespeicherte Daten zum einen über die Berechtigungen des Betriebssystems geschützt und zum anderen über die Sandbox, in der die Anwendung innerhalb der VM ausgeführt wird. Es ist daher nicht möglich, unerlaubt aus einer Android-Anwendung heraus auf die gespeicherte Daten einer anderen Anwendung zuzugreifen (Quelle: [Arno Becker, 2009] s. 17,18).

5.1.3 Entwicklungsumgebung

Eine sehr bequeme Oberfläche fur Anwendungsentwicklung ist Eclipse. Um für Android entwickeln zu können, bietet Google für den Eclipse Plugin- ADT, der Android Virtual Device-

Manager, das Android-SDK und der Dalvik Debug Monitor Server(DDMS) enthält. Android Virtual Device(AVD) ist ein Emulator, mit dem fast jedes Android-Gerät simuliert werden kann. Auf dem Rechner soll Java(mindestens 1.5) vorhanden sein. Installationshinweise findet man unter dem Link [1].

5.1.4 Android-Komponenten

Anwendungskomponenten sind die wesentlichen Bausteine einer Android-Anwendung. Es gibt vier verschiedene Arten von Anwendungskomponenten. Jede Art dient einem besonderen Zweck und hat einen eindeutigen Lebenszyklus.

Activity

Anwendungen, die mit dem Anwender interagieren, brauchen mindestens eine Activity, um eine Oberfläche darzustellen. Activities sind sichtbar und können miteinander zu einer komplexeren Anwendung verknüpft werden. Sie kümmern sich um die Darstellung von Daten und nehmen Anwendereingaben entgegen.

Service

Nicht jeder Teil einer Anwendung braucht eine Oberfläche. Ein Service erledigt Hintergrundprozesse.

Content Provider

Nicht jede, aber viele Anwendungen bieten die Möglichkeit, Daten zu laden oder zu speichern. Ein Content Provider verwaltet Daten und abstrahiert die darunterliegende Persistenzschicht. Er kann über Berechtigungen seine Daten einer bestimmten Anwendung oder auch vielen Anwendungen zur Verfügung stellen. Er hat eine definierte Schnittstelle und wird darüber lose an Anwendungen gekoppelt.

Broadcast Receiver

Durch die komponentenbasierte Anwendungsentwicklung unter Android ist es notwendig, zwischen Betriebssystem und Anwendungen zu kommunizieren. Auch die Kommunikation zwischen Anwendungen ist möglich. Broadcast Receiver lauschen jedoch als Komponente auf Broadcast Intents, die auf Systemebene verschickt werden und z.B. über Störungen der Netzwerkverbindung informieren oder über einen schwachen Akku.

(Quelle: [Arno Becker, 2009] s. 20,21).

[1] $http://www.androidpit.de/de/android/wiki/view/Android\_Anf\%C3\%A4nger\_Workshop\#toc33$

5.1.5 Vorteile von Android und Kritik

Auf dem Markt gibt es mehrere Betriebssysteme für mobile Geräte: iOS, Android, Windows Mobile, Symbian. Android, wie alle anderen Betriebssystemen, hat seine Stärken und Schwächen.

Vorteile

- Interne Software ist auf das gleiche API geschrieben wie die Software von anderen Entwicklern und die Ausführungszeit ist gleich. Diese API gibt Zugriff an Sensorsteuerung, GPS, Datenbankensteuerung, 2D und 3D usw.
- Es ist möglich die Anwendungen ohne Internetverbindung, z.B. von Flash-Memory, problemlos zu installieren.
- Für Android kann Jedermann kostenfrei Anwendungen schreiben und diese auf eigenem Gerät austesten.
- Android ist für verschiedene Architekturen wie ARM, MIPS, x86 geeignet.
- Android kann Dateisysteme von einem Gerät exportieren, wie USB mass storage device, was einen direkten Austausch von Daten zwischen z.B. Smartphone und PC ermöglicht (im Gegensatz zu iOS, das die Daten nur synchronisieren kann).

Kritik

- Fernzugriff auf Geräte von Google: es ist möglich, Software zu löschen oder zu installieren, ohne der vorherigen Nachfrage beim Nutzer.
- Obwohl Android freie Software ist, befinden sich bei den meisten standardmäßig ausgelieferten Android-Geräten proprietäre Software von Google.
- Wegen hoher Verbreitung von Android und fehlender Kontrolle im Google Play, gibt es sehr viele schädliche Programme für Android.

Alle Kritikpunkte haben für diese Arbeit keinen Einfluss. Die Vorteile, wie kostenlose Entwicklungsumgebung, einfaches Testprozess und Popularität von Android-Plattform, sind von großer Bedeutung.

5.2 OpenCV

5.2.1 OpenCV Überblick

OpenCV[2] ist eine freie Programmbibliothek für die Bildverarbeitung und maschinelles Sehen. Sie wurde für die Programmiersprachen C und C++ geschrieben und läuft auf Windows, Linux, Android und Mac. OpenCV steht als freie Software unter den Bedingungen der BSD-Lizenz (d.h. für die akademische und kommerzielle Nutzung ist OpenCV frei verfügbar). OpenCV (Computer Vision) wurde zunächst von Intel unter dem Namen IPL (Image Processing Library) entwickelt und der Quellcode später als OpenCV freigegeben. Seither kümmert sich die Firma Willow Garage federführend um OpenCV, wobei viele Unternehmen, Hochschulen und freie Entwickler weltweit mit Verbesserungen, und neuen Funktionen und Schnittstellen zur Weiterentwicklung beitragen.

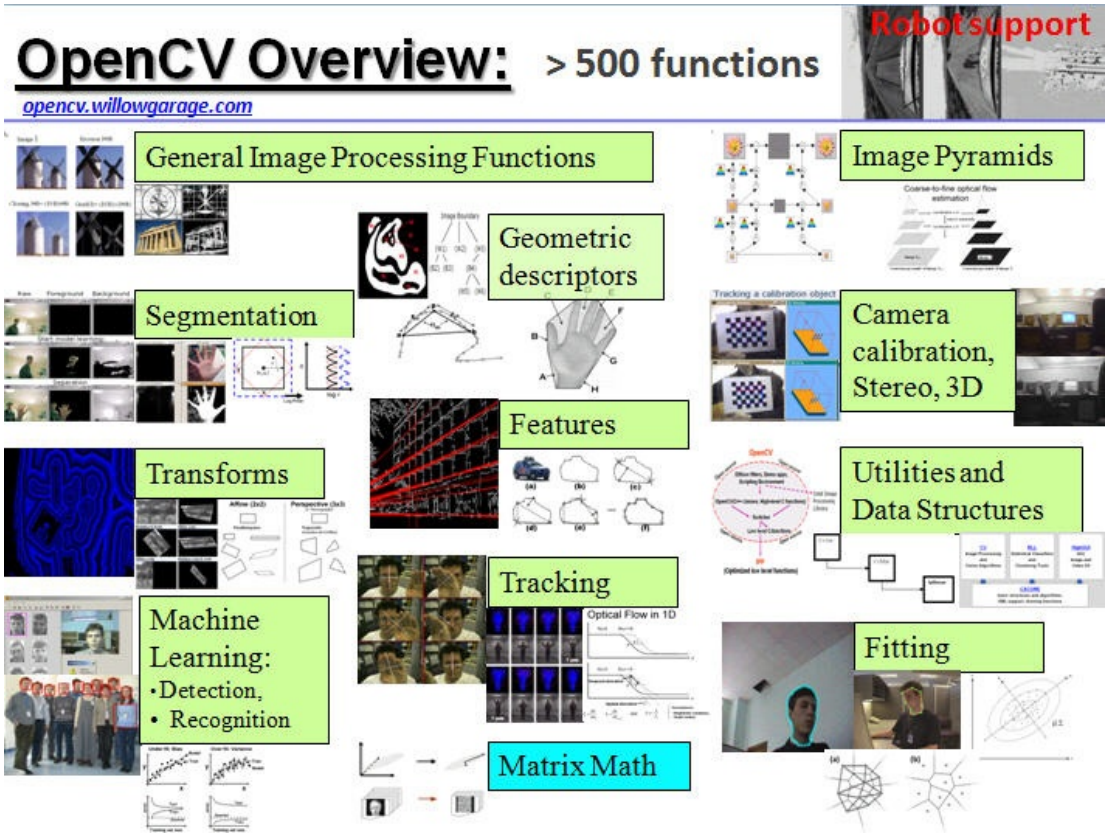

Abbildung 5.3: OpenCV Überblick

[2] http://opencv.willowgarage.com/wiki/

Die Stärke von OpenCV liegt in ihrer Geschwindigkeit und in der großen Menge der Algorithmen aus neuesten Forschungsergebnissen. In den über 2500 optimierten Algorithmen sind über 500 Funktionen für die allgemeine Bildverarbeitung aber auch für speziellere Aufgaben wie z.B. Kantenerkennung, Segmentierung, Transformationen, Geometrische Deskriptoren, Matrix-Kalkulation, Kamera-Kalibrierung, Stereo und 3d, Gestenerkennung, etc. enthalten. Außerdem enthält OpenCV rudimentäre GUI-Elemente, die besonders bei der Entwicklung und beim Testen der Anwendung benutzt werden (HighGui) und Schnittstellen zur Bildaufnahme von verschiedenen Kameras und Framegrabbern (Abbildung 5.3).

5.2.2 Opencv und Android

Letztes Jahr wurden 5 Versionen von OpenCV für Android herausgestellt. Aktuelle Version von OpenCV ist 2.4.2 (Stand 21.07.2012). Das ist ein gutes Zeichen für die Popularität von Android als Platform für die bildverarbeitungsbasierte Lösungen. In Rahmen dieser Arbeit werden folgende OpenCV Modulen benutzt:

Core
: Grundfunktionalität: einfache Operationen auf Arrays; Matrixoperationen, matematische Funktionen; DFT; XML; Zeichenfunktionen (2D-Graphik), komplexe Datenstrukturen: spärliche Matrizen, Wachsende Sequenzen, Graphen.

calib3d
: Kamerakalibrierung, Stereo-Korrespondenz.

features2d
: Feature Detection; Keypoint Suche; Deskriptor extraction and matching.

highgui
: Aufnahme und Wiedergabe von Bilder und Videos; Einfache GUI mit Nutzerinteraktionen.

imgproc
: Methoden zur Bildverarbeitung; Histogramme; Filter; Geometrische Bildtransformationen; Merkmalserkennung; Objekterkennung.

Bis zum heutigen Tage gibt es zwei Möglichkeiten OpenCV für Android zu verwenden: kompilierte Java-Bibliothek oder Native Code.

OpenCV als Java-Bibliothek

Auf der Internetseite http://sourceforge.net/projects/opencvlibrary/files/opencv-android/ steht OpenCV für Android als Java-Bibliothek zu verfügung. Um eine Anwendung Enwickeln zu können, benötigt man dazu noch SUN JDK 6, Android SDK, Eclipse IDE, ADT Plugin for Eclipse. Benötigte Software kann man einzeln herunterladen oder kann man Tegra Android Development Pack (TADP) von nVIDIA nutzen. Detalierte Beschreibung für die Instalationsschritte findet man in der Dokumentation[3] auf OpenCV-Homepage. Man muss die OpenCV nicht mehr selbst kompilieren, außer wenn eine spezielle Konfiguration notwendig ist. Dafür werden OpenCV Source Code, Cygwin (ein Übersetzer von Linux-Befehlen für Windows) und Android NDK (Android Native Development Kit) benötigt. Ein Tutorium [www.stanford.edu, 2012] zeigt Schritt für Schritt wie man selbst OpenCV kompilieren kann.

OpenCV als C/C++ Bibliothek und JNI

Entwicklung für Android bedeutet Entwicklung mittels Java. Entwicklung für Android bedeutet Entwicklung mittels Java. Aber manchmal werden einige Computer Vision Codes, die man in der Android-Anwendung verwenden möchte, mit C/C++ geschrieben. In diesem Fall, wenn man den bereits geschriebenen C/C++ Code in Java nicht neu schreiben möchte, wird ein JNI verwendet. JNI ist ein Java-Framework für die Interaktion mit nativem Code. Das bedeutet, dass in Java-Klasse native Methoden einsetzt werden, um der Java-Teil von Android-Anwendung die C++ Funktionalität hinzuzufügen. Die genauere Vorgehensweise, wie man es realisieren kann, findet man in der OpenCV-Dokumentation.

In Rahmen dieser Arbeit wird nur OpenCV als Java Bibliothek benutzt. D.h. für die Implementierung wurden ausschließlich Java und keine C/C++ Codes verwendet.

[3] *http : //docs.opencv.org*

5.2.3 Android OpenCV Manager

OpenCV-Manager[4] ist ein Android-Dienst, der OpenCV Bibliotheksdateien auf Endanwender-Geräten verwaltet. Es ermöglicht die Aufteilung der OpenCV dynamischen Bibliotheken von verschiedenen Versionen zwischen den Anwendungen auf dem selben Gerät.

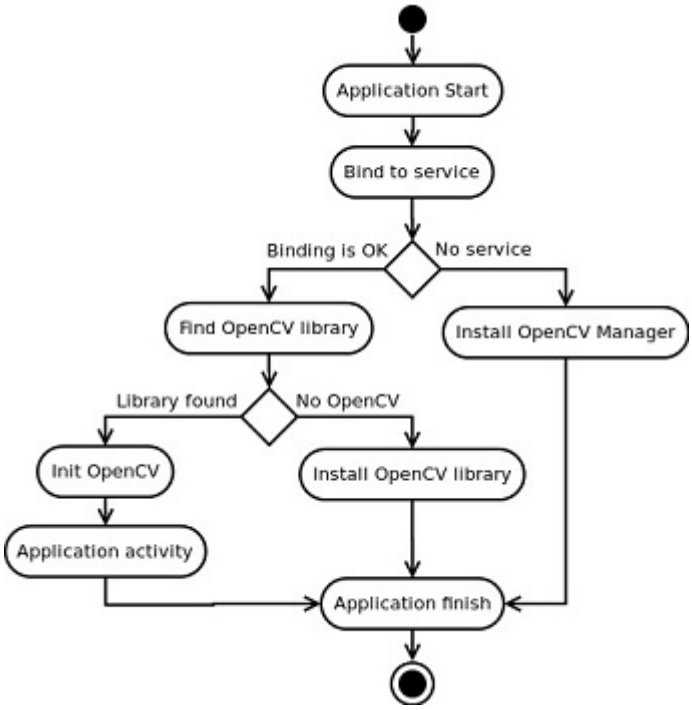

Abbildung 5.4: Nutzungsmodell von OpenCV Manager

Die Abbildung 5.4 zeigt, dass beim Start erster OpenCV Anwendung der OpenCV Manager von Google Play installiert wird (Internetverbindung wird notwendig). Nach der Installation soll der Benutzer manuell die gewünschte Anwendung neu starten. Wenn noch keine benötigte Bibliothek installiert wurde, wird diese von Google Play runtergeladen und installiert. Dann soll der Benutzer ein drittes Mal die Anwendung neu starten und diese wird funktionieren. Details zur Implementierung und Nutzung findet man auf der Android Manager Home Page [5].

[4] ist ab OpenCV-Version 2.4.2 (04.07.2012) vorhanden
[5] http://docs.opencv.org/android/service/doc/index.html

5.3 Zusammenfassung

Android sowie OpenCV sind Projekte, die sich ständig weiterentwickeln. Da die Android-Smartphones immer wieder mehr Leistung mit sich bringen und sich OpenCV in Richtung Android entwickelt, ist es möglich viele schwere und interessante Aufgaben mit ihrer Hilfe zu lösen. Aufgrund der Tatsache, dass beide Umgebungen kostenfrei zur Verfügung stehen und viele Möglichkeiten mit sich bringen, sind sie die best passenden Werkzeuge für diese Arbeit.

6 Realisierung

In diesem Kapitel wird der Algorithmus Metrische Rekonstruktion, der in Kapitel Grundlagen kurz vorgestellt wurde, praktisch umgesetzt.

Dieses Kapitel beschreibt Testergebnisse von einzelnen Schritten der Metrischen-Rekonstruktion, die im Kapitel Analyse detailliert beschrieben wurden. Alle Tests wurden mittels Samsung Galaxy S9001i (Single Core 1.4Ghz, 512Mb Arbeitsspeicher) durchgeführt.

6.1 Kamerakalibrierung mit OpenCV

Um Triangulieren zu können, werden innere Kameraparameter benötigt. Diese Parameter werden mittels Kamerakalibrierung berechnet. Die Kamerakalibrierung kann nur einmal auf einem Smartphone durchgeführt werden. Die berechneten Werte können im Programm gespeichert werden und bei jedem Rekonstruktionsprozess wieder benutzt werden.

Der Algorithmus für die Kamerakalibrierung wurde im Kapitel Grundlagen beschrieben. Um diesen Algorithmus praktisch umzusetzen, wurde OpenCV Funktion calibrateCamera (objectPoints, imagePoints, imageSize, cameraMatrix, distCoeUs, rvecs, tvecs) verwendet, wobei

- objectPoints: ein Vektor von Vektoren ist, welcher die Kalibrierungsmuster-Punkte in dem Kalibrierungsmuster-Koordinatenraum enthalten. Der äußere Vektor enthält so viele Elemente, wie die Anzahl der Muster ist. Wenn in jedem Bild dasselbe Muster abgebildet ist und das Bild vollständig sichtbar ist, werden alle inneren Vektoren gleich sein. Koordinatenberechnung: $X = Punktnummer/Musterbreite$, $Y = Punktnummer\%Musterbreite$ und Z-Koordinate ist gleich null.

- imagePoints: ein Vektor von Vektoren ist, welcher Projektionskoordinaten von Musterpunkten enthält. Der äußere Vektor soll genauso groß sein, wie der äußere Vektor bei objectPoints. Die Berechnung von Koordinaten wird im Folgenden erklärt.

- imageSize: Breite und Höhe von Quellbildern in Pixel ist.

- cameraMatrix: berechnete K-Matrix (siehe Kapitel Kamerakalibrierung). f_x, f_y, c_x und c_y vorinitialisiert werden soll.
- distCoeffs: die berechnete optische Verzerrung ist [1].
- rvecs und tvecs: Berechnete Drehmatrix und Translationvektor für äußere Kalibrierungsmatrix ist (siehe Kapitel Kamerakalibrierung).

ObjectPoints und imagePoints sind wichtigste Parametern für die Kalibrierung. Um Projektionskoordinaten von Musterpunkten zu berechnen, kann man OpenCV-Funktion findChessboardCorners (image, patternSize, corners) verwenden. Auf *image*, das die Größe *paternSize* hat, werden Musterpunkte erkannt und mit seinen X-und Y-Koordinaten in corners gespeichert. Mit Funktion cornerSubPix() werden Koordinaten von Musterpunkten verfeinert. Mit Hilfe von drawChessboardCorners() werden erkannte Punkte veranschaulicht (Abbildung 6.1).

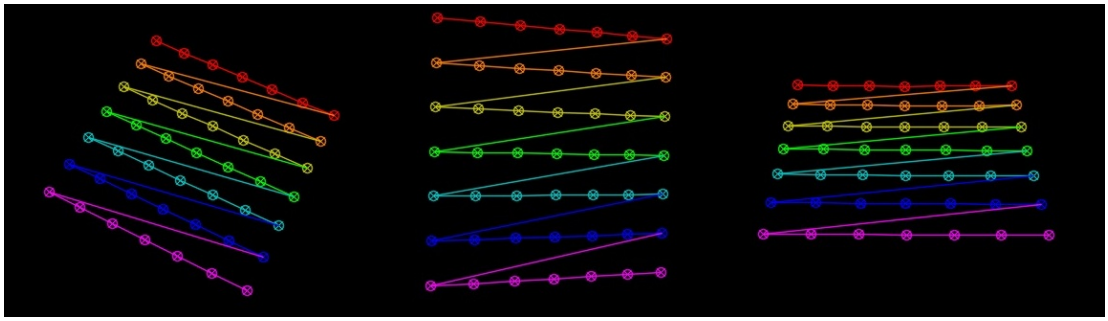

Abbildung 6.1: Erkannte Punkte auf Kalibrierungsmuster (siehe Abbildung 4.6)

Kalibrierungsprozesse können für einen Smartphone zu aufwendig sein und sehr lange dauern. Die benötigten Ressourcen (Zeit und Smartphoneleistung) hängen von der Auflösung der aufgenommenen Bilder ab. Je kleiner die Auflösung ist, desto weniger Ressourcen werden für den Kalibrierungsprozess benötigt. Bei Minimierung der Auflösung soll man aber die Ergebnisse der Kalibrierung beachten. Um festzustellen, ob die Kalibrierungsergebnisse von der Bilderauflösung unabhängig sind, werden 5 Tests mit den unterschiedlichen Auflösungen von Bildern durgeführt (Bildsequenz für Kalibrierung besteht aus 14 Bildern). In der Tabelle 6.1 sind die Ergenisse aller Tests zusammengefasst. Bei sehr kleinen (160×120) und sehr großen (2560×1920) Auflösungen liegen die Werte außer Tendenz. Deswegen werden die Ergebnisse von der mittleren Auflösung (640×480) weitergenutzt.

[1] Optische Verzerrung ist ein geometrischer Abbildungsfehler (kissenförmige oder tonnenförmige Verzeichnung).

Aflösung	Zeit(s)	f_x	f_y	c_x	c_y	R	T
160 × 120	2.1	100.02	100.02	77.76	73.84	2.22	−3.71
320 × 240	2.12	4923.64	4923.64	163.26	107.384	0.58	3.97
640 × 480	7.06	6857.63	6857.63	325.27	211.86	0.58	4.06
1600 × 1200	43.31	8965.34	8965.34	796.68	507.56	0.57	4.32
2560 × 1920	132.9	2517.7	2517.7	1288.41	905.62	0.09	2.5

Tabelle 6.1: Kamerakalibrierung mit verschiedene Bildauflösungen

6.2 Korrespondenzsuche

Nächster Schritt der metrischen Rekonstruktion ist die Korrespondenzsuche. Um zwei aufeinanderfolgende Bilder miteinander vergleichen zu können und die Szenengeometrie wiederherzustellen, wird die Korrespondenzsuche der zwei Bildern durchgeführt. Dieser Schritt kann auf drei Phasen verteilt werden, wie in Abbildung 6.2 dargestellt ist. Die gefundenen Korrespondenzen (Merkmale) werden später für die Bestimmung von Kamerapositionen verwendet, weswegen dieser Schritt von großer Bedeutung ist.

Abbildung 6.2: Drei Phasen des Korrespondenzsuche

6.2.1 Features Detection

Es gibt viele Techniken für die Suche von Interest-Operatoren, wie zum Beispiel Harris corner detector, SIFT, SURF, ORB und andere. Da es sich in diesem Fall um eine Anwendung für Smartphones handelt, soll immer der Ressourcenaufwand bei der Auswahl von Algorithmen beachtet werden. Um den am besten passenden Algorithmus zu wählen, werden unterschiedliche Tests durchgeführt. Es werden zwei Auswahlkriterien festgesetzt: die Ausführungszeit und die Anzahl der gefundenen Punkte. In der Abbildung (Abbildung 6.3) sind zwei Bildpaaren dargestellt, die in drei Auflösungen getestet wurden.

Die Tabelle 6.2 zeigt die Testergebnisse für das Harris corner detector-Verfahren. Für Bilder mit großer Auflösung ist dieses Verfahren nicht verwendbar, da die Merkmalerkennung sehr viel Zeit kosten kann und unerwünschte Ergebnisse liefern kann, wie z.B. bei dem ersten Bildpaar in der Auflösung 1600×1200.

Abbildung 6.3: Zwei Bildpaaren für Features Detection Tests

	Bildpaar 1			Bildpaar2		
Aflösung	**Zeit(s)**	**Punkten**	**P/s**	**Zeit(s)**	**Punkten**	**P/s**
640 × 480	2.9	25/10	6.03	2.6	29/36	12.5
1600 × 1200	440	$1.6 * 10^6/1.9 * 10^6$	4355.55	31.1	285/304	9.46
2560 × 1920	×	×	×	×	×	×

Tabelle 6.2: Harris Corner Detection: Testergebnisse

Das zweite Verfahren, das getestet wurde, ist SURF (siehe Kapitel 4.5.2). Die Ergebnisse in der Tabelle 6.3 zeigen, dass mit der Zunahme der Bildauflösung die Anzahl der gefundenen Merkmale stetig zunimmt. Dies ist ein besserer Indikator als beim Harris corner detection-Verfahren. Aber bei großen Auflösungen bleibt noch eine unakzeptable Ausführungszeit. Abbildungen 6.4 und 6.5 veranschaulichen die Ergebnisse für beide Bildpaare in der Auflösung 640×480.

Ein weiteres Verfahren, das getestet wurde, ist ORB. ORB ist eine neue Technik, die auf FAST keypoint detector [Rosten u. Drummond, 2006] und BRIEF Deskriptor [M. Calonder u. Fua., 2010] basiert. In Rahmen dieser Arbeit werden nur Ergebnisse von ORB-Detektor betrachtet. Die genaue Algorithmusbeschreibung kann dem Artikel „ORB: an efficient alternative to SIFT or SURF“ [Ethan Rublee, 2011] entnommen werden. Der Algorithmus ist in die OpenCV

	Bildpaar 1			Bildpaar 2		
Aflösung	**Zeit(s)**	**Punkten**	**P/s**	**Zeit(s)**	**Punkten**	**P/s**
640 × 480	1.3	683/690	528.1	1.66	1018/885	3585.2
1600 × 1200	8.2	2354/3165	336.5	10.4	5686/5607	542,93
2560 × 1920	24.5	5853/8870	300.5	30.3	10203/11697	361.4

Tabelle 6.3: SURF Feature Detection: Testergebnisse

Abbildung 6.4: Testergebnisse von SURF-Merkmaldetektor (erste Bildpaar, Auflösung 640 × 480)

Abbildung 6.5: Testergebnisse von SURF-Merkmaldetektor(zweite Bildpaar, Auflösung 640 × 480)

implementiert und die Suche der Merkmale ist auf 500 Punkte begrenzt. Aufgrund dieser

Begrenzung liefert ORB sehr kleine Ausführungszeit für alle Bildauflösungen (siehe Tabelle 6.4).

	Bildpaar 1			Bildpaar2		
Aflösung	**Zeit(s)**	**Punkten**	**P/s**	**Zeit(s)**	**Punkten**	**P/s**
640 × 480	0.13	500/500	3846.1	0.14	500/500	3571.4
1600 × 1200	0.61	500/500	819.8	0.75	500/500	666.7
2560 × 1920	1.5	500/500	333.3	1.52	500/500	328.9

Tabelle 6.4: ORB Feature Detection: Testergebnisse

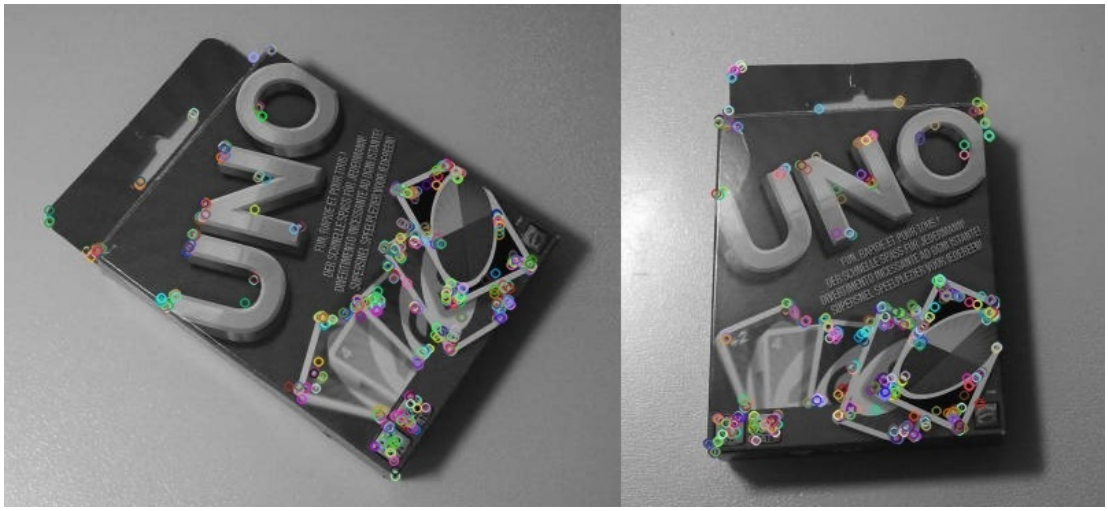

Abbildung 6.6: Testergebnisse von ORB-Merkmaldetektor (erste Bildpaar, Auflösung 640×480)

Für die Ausführung der Tests hat die kleinere Ausführungszeit größere Bedeutung als die Anzahl der gefundenen Punkte. Aber die Anzahl der Merkmale darf dabei nicht zu klein sein.

Die besprochenen Gründe führen zur Auswahl der SURF und ORB Verfahren, die mit unterschiedlichen Deskriptoren weiter getestet werden. Harris corner detection wird nicht mehr verfolgt.

6.2.2 Deskriptor Extraction und Deskriptor Matching

Ein Deskriptor beschreibt eindeutig die Umgebung eines gefundenen Merkmalpunktes in Form eines Vektors. Qualitätskriterien für Deskriptoren sind Invarianzen gegen Rotationen und Beleuchtungsänderung. In diesem Kapitel werden ORB- und SURF-Detektoren mit unterschiedlichen Deskriptoren getestet. Nach Berechnung (Extraktion) von Deskriptoren wird ein

Abbildung 6.7: Testergebnisse von ORB-Merkmaldetektor (zweite Bildpaar, Auflösung 640 × 480)

Matching[2] von Bildpaaren durchgeführt. Dafür werden zwei Techniken verwendet: FLANN [Muja u. Lowe, 2009] und Brute-force Deskriptor matcher[3].

Die erste Kombination für FLANN-Matching ist SURF-Detektor und SURF-Deskriptor. Wie die Tabelle 6.5zeigt, ist diese Kombination bei großen Auflösungen sehr aufwendig. Dabei ist die Anzahl von gefundenen Paaren ziemlich klein. Es wird nicht nur die Anzahl als Ergebnis berücksichtigt, sondern auch die Genauigkeit der gewählten Paare. Die Abbildung 6.8 zeigt, wie ungenau dieses Matching-Kombination Punktepaare gefunden hat.

	Bildpaar 1		**Bildpaar2**	
Aflösung	**Zeit(s)**	**Matches**	**Zeit(s)**	**Punkten**
640 × 480	7.1	79	12.3	8
1600 × 1200	60	167	277.23	36
2560 × 1920	×	×	×	×

Tabelle 6.5: FLANN Deskriptor Matching: Testergebnisse SURF Detector und SURF Deskriptor

Die zweite Kombination von Detektor und Deskriptor, die mit FLANN-Matching getestet wurde, ist die Kombination von ORB-Detektor und SURF-Deskriptor. Die in der Tabelle 6.6 dargestellten Zahlen zeigen, dass diese Kombination bessere Ergebnisse in der Ausführungszeit sowie auch in der Anzahl der gefundenen Paare liefert. In der Abbildung 6.9 ist zu sehen, dass die Genauigkeit der Paarenauswahl auch gestiegen ist.

[2]Matching ist ein Prozess, bei den Merkmalpunkte auf zwei Bildern vergleichen werden, um Paare zu finden. Es wird mittels Berechnung des geometrischen Abstandes der Deskriptoren durchgeführt (bessere Paar ist mit kleinsten Abstand)

[3]Brute-Force-Methoden beruht sich auf dem Ausprobieren aller möglichen Fälle.

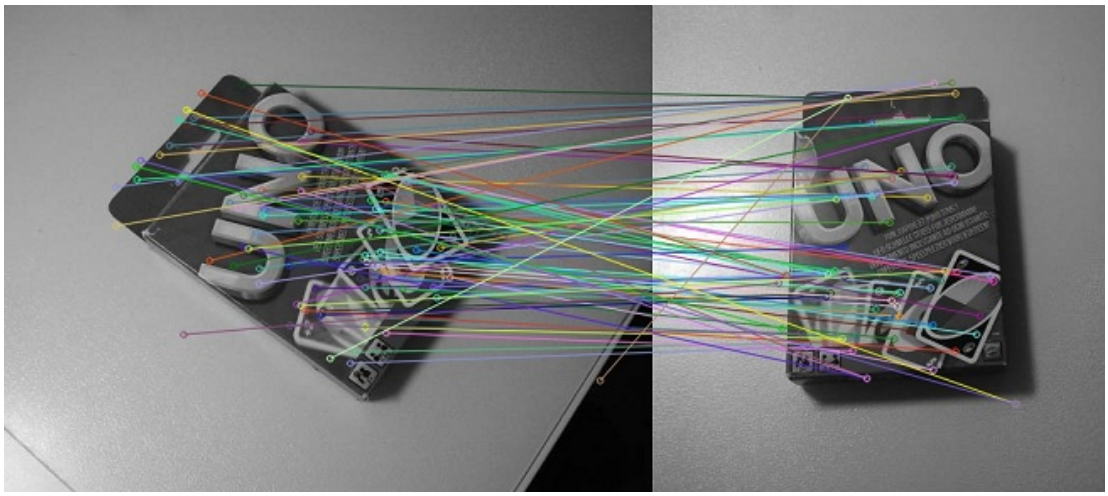

Abbildung 6.8: FLANN-Matching mit SURF-Detector und SURF-Deskriptor (Auflösung 640×480)

	Bildpaar 1		**Bildpaar2**	
Aflösung	**Zeit(s)**	**Matches**	**Zeit(s)**	**Punkten**
320 × 240	5.1	116	3.5	41
640 × 480	5.72	175	5.36	37
1600 × 1200	6.84	100	6.71	19
2560 × 1920	8.78	36	9.05	21

Tabelle 6.6: FLANN Deskriptor Matching: Testergebnisse ORB Detector und SURF Deskriptor

Es wurden noch andere Detektor-Deskriptor-Kombinationen getestet. Diese haben aber schlechtere Ergebnisse geliefert. Alle Brute-Force-Methoden wurden als ungeeignet Bezeichnet, weil sehr viele Merkmalpaare falsch zusammengestellt wurden. Daraus folgt, dass die Kombination aus ORB-Detektor und SURF-Deskriptor mit FLANN-Matching für das gestellte Ziel am besten geeignet ist.

Um die untere Grenze der Bildauflösung festzustellen, bei der die Ergebnisse noch Sinn machen, wurden Tests mit der Auflösung 320×240 durchgeführt. Da kein erheblicher Gewinn in der Ausführungszeit erzielt wurde, und Extrahieren der Textur in den letzten Phasen der 3D-Rekonstruktion auf einer so niedrigen Auflösung schwierig sein kann, wird diese Auflösung nicht weiter verwendet.

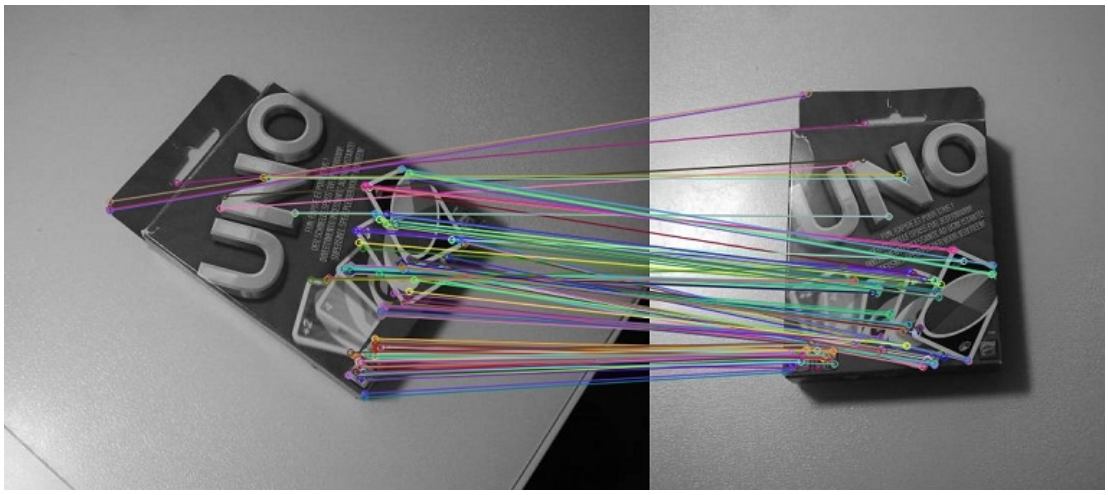

Abbildung 6.9: FLANN-Matching mit ORB-Detector und SURF-Deskriptor (Auflösung 640×480)

6.3 Triangulation

Um 3D-Punktkoordinaten bestimmen zu können, ist es notwendig Triangulation durchzuführen. Dafür bietet OpenCV Funktion triangulatePoints (projMatr1, projMatr2, points1, points2, points4D), wobei

- projMatr1: Projektionsmatrix für die Kamera ist, mit der das erste Bild gemacht wurde (siehe 4.12)
- projMatr2: Projektionsmatrix für die Kamera ist, mit der das zweite Bild gemacht wurde (siehe 4.13)
- points1: $2 \times N$ Array von markanten Punkten auf dem ersten Bild ist;
- points2: $2 \times N$ Array von Korrespondenzpunkten auf dem zweiten Bild ist;
- points4D: Array von rekonstruierten Punkten in der homogenen Koordinate ist.

Wie in dem Abschnitt Epipolare Geometrie erläutert wurde, um Projektionsmatrix für die Kamera, mit der das zweite Bild gemacht wurde, berechnen zu können, wird eine Rotationsmatrix und ein Translationsvektor notwendig. Diese Information kann aus der Essential-Matrix erhaltet werden. Um Essential-Matrix zu bestimmen, werden Kameramatrix, die durch die Kamerakalibrierung bekannt ist, und Fundamentalmatrix benötigt (siehe Formel 4.8). In OpenCV

gibt es dafür die Methode findFundamentalMat (points1, points2, method). Die zwei ersten Parameter points1 und points2 sind dieselben Punkte, für die die Triangulation durchgeführt wurde. Der dritte Parameter ist der Algorithmus für die Berechnung von Fundamentalmatrix. In dieser Arbeit wird RANSAC-Algorithmus verwendet.

Anzahl der Korrespondenzpunkten	Triangulationszeit(in Sekunden)
37	0.04
172	0.11
398	0.32

Tabelle 6.7: Abhängigkeit der Triangulationszeit von der Punktenanzahl

Die Tabelle 6.7 zeigt die Abhängigkeit der Zeit, die für die Triangulation notwendig ist, abhängig von der Anzahl der Korrespondenzpunkte. Die erwartete Tendenz, je mehr Punkte, desto mehr Zeit für Triangulation notwendig, wurde festgestellt. Dabei ist die Zeit relativ klein: ca. 1000 Punkte pro Sekunde konnten trianguliert werden.

X-Koordinate	Y-Koordinate	Z-Koordinate
880.80	854.42	1.00
508.81	328.83	1.00
657.63	316.81	1.00
960.77	321.41	1.00
887.51	298.60	1.00

Tabelle 6.8: Berechnete Koordinaten von Korrespondenzpunkten in Weltkoordinatensystem

Da die OpenCV-Funktion triangulatePoints Koordinaten von triangulierten Punkten in homogener Form liefert, sollen diese in kartesische Form transformiert werden, wie in der Abbildung 4.3 dargestellt ist. Experimentell wurde festgelegt, dass die passende Formel für Projektionsmatrix die Kamera, mit der das zweite Bild gemacht wurde, ist (siehe Formel 6.1). In den anderen drei Fällen sind die Koordinatenwerte negativ.

$$P' = UW^T V^T | - u_3 \tag{6.1}$$

Die Tabelle 6.8 zeigt Beispiele für die berechneten 3D-Koordinaten der Korrespondenzpunkte von dem ersten Bildpaar (siehe Abbildung 6.3). Die Abbildung 6.10 veranschaulicht die 2D-Projektion der 3D-Koordinaten von durch die Triangulation ermittelten Punkte für das erste Bildpaar in der Auflösung 640×480.

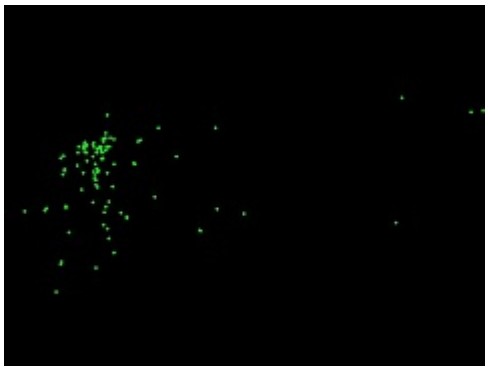

Abbildung 6.10: 2D-Projektion der triangulierte Punkte

6.4 Zusammenfassung

In diesem Kapitel wurden verschiedene Methoden für Erkennung von markanten Punkten getestet. Richard Hartley und Andrew Zisserman beschreiben in ihrem Buch „Multiple View Geometry in Computer Vision“ [Hartley u. Zisserman, 1997] die Harris Corner Detektion als am besten passenden Algorithmus für diese Aufgabe. Im Vergleich zu SURF- und ORB-Algorithmen haben die Tests aber schlechtere Ergebnisse geliefert. Es wurden auch verschiedene Kombinationen von Detektoren, Deskriptoren und Matching-Techniken, die für paarweise Verknüpfung von Korrespondenzpunkten auf zwei Bildern dienen, getestet. Aufgrund von Ausführungszeit, Anzahl gebildeter Paare und Zuverlässigkeit der Paarsuche wurden der ORB-Detektor, der SURF-Deskriptor und das FLANN-Matching gewählt. Mit Hilfe der gefundenen Punktepaare, wurden dann die Koordinaten von abgebildeten Punkten mittels Triangulation im Weltkoordinatensystem bestimmt.

7 Zusammenfassung und Ausblick

7.1 Zusammenfassung

In der vorliegenden Bachelorarbeit wurden verschiedene Verfahren für 3D-Rekonstuktion vorgestellt und auf Verwendbarkeit für eine Smartphone-basierte Anwendung untersucht. Wegen seiner Mobilität und Verfügbarkeit und wegen seiner ständig wachsenden Leistungen und Möglichkeiten ist ein Smartphone sehr gut für das gestellte Ziel geeignet.

Im Kapitel Grundlagen wurde festgestellt, dass viele Techniken für 3D-Rekonstruktion nur mit zusätzlichen Lichtquellen, Messgeräten oder Vorkenntnissen über zu rekonstruierendes Objekt verwendet werden können. Deswegen können diese Techniken nicht ohne Weiteres für Smartphones angewendet werden. Des weiteren wurden bildbasierte Rekonstruktionsmethoden untersucht und diskutiert. Eine Variante wäre Stereorekonstruktion, die aus zwei Bildern ein 3D-Modell vom abgebildeten Bereich schätzen kann. Um ein vollständiges und detailliertes 3D-Modell zu erstellen, müssen mehrere Bilder vom Objekt aufgenommen werden. Dabei entstehen folgende Probleme:

- das Bestimmen der überlappenden Bereiche in Bildsequenzen
- das extrahieren der Kameraposition aus den Bildern.

Es zeigt sich, dass im Vergleich zu Stereorekonstruktion die metrische Rekonstruktion die geforderte 3D-Rekonstruktion aus Bildsequenzen durchführen kann: korrespondierende Punkte auf den Bildern sowie Informationen über die Kameraparameter ermöglichen Rekonstruktion der Szenengeometrie und Bestimmung der dreidimensionalen Koordinaten durch Triangulation.

Im Kapitel Analyse wurde die Methode der metrischen Rekonstruktion detalliert untersucht. Die wichtigste Aufgabe bei diesem Verfahren ist die Bestimmung der Position von Bildpunkten in dem Weltkoordinatensystem. Um das zu realisieren, muss jeder Punkt auf mindestens zwei Bildern abgebildet sein. Wenn die Bestimmung der Koordinaten für jeden Pixel auf jedem Bild durchführt wird, wird dieser Prozess sehr aufwendig. Vergleichbar aufwendig ist das Bestimmen der korrespondierenden Punkte in Sequenzen von Bilder. Bei der Untersuchung von

Algorithmen zur Korrespondenzpunktesuche wurden effiziente Verfahren, wie zum Beispiel Harris corner detection oder SIFT, identifiziert um diese für eine 3D- Rekonstruktion auf einem Smartphone zu überprüfen.

Im Kapitel Realisierung wurden Testergebnisse verschiedener Algorithmen vorgestellt und diskutiert. Aufgrund dieser Ergebnisse stellt sich eine Kombination von den ORB-Deskriptor, den SURF-Detektor und den FLANN-Matching als beste heraus. Es wurden auch Triangulationsergebnisse in Form von 2D-Projektion und eine Auflistung von 3D-Koordinaten vorgestellt. Nach Zusammenfassen aller Ergebnisse wurde die Zeit, die für die Bestimmung der 3D-Koordinaten eines Objektes aus einer Bildsequenz notwendig ist, eingeschätzt. Für eine Bildsequenz aus N Bildern mit noch unbekannten Kameraparametern wird die Berechnungszeit t mit Formel 7.1 berechnet:

$$t = t_k + N(t_m + t_t) \tag{7.1}$$

mit Kalibrierungszeit t_k, Matchingzeit t_m und Triangulationszeit t_t. Zum Beispiel, für eine Bildsequenz aus zwanzig Bilder, die Auflösung 640×480 haben, wäre $t = 7s + 20(6s + 0.5s) = 137s$.

7.2 Ausblick

Im Juni 2012 wurde auf der IEEE Conference on Computer Vision and Pattern Recognition eine neue Methode für die Korrespondenzsuche namens FREAK: Fast Retina Keypoint von Alexandre Alahi, Raphael Ortiz und Pierre Vandergheynst präsentiert[1]. Da Vision-Algorithmen heutzutage oft in Smartphones eingesetzt werden, war das Ziel der FREAK-Entwickler einen Algorithmus zu entwickeln der schneller als zum Beispiel SIFT oder SURF ist und dabei invariant zur Skalierung, Drehung und anderen Störungen bleibt. Die Testergebnisse haben gezeigt, dass zum Beispiel für matching von zwei Bildern mit einer Auflösung von 800x600 22 Mal weniger Zeit benötigt wurde als bei SURF-Algorithmus mit gleicher Anzahl korrespondierender Punkte. Es ist ein Grund dafür zu versuchen diesen Algorithmus für die 3D-Rekonstruktion einzusetzen.

Mittels Triangulation bekommt man Punkte im Weltkoordinatensystem die eine Punktewolke bilden. Aus dieser Wolke lässt sich ein 3D-Modell erstellen. Dieser Prozess wird meshing genannt. Das am häufigsten benutzte meshing-Verfahren ist die Voxel-Konversionstechnik die im Rahmen dieser Bachelorarbeit nicht erläutert wurde. Nach der Modellbildung konnte das

[1] http://infoscience.epfl.ch/record/175537/files/2069.pdf

Modell in einem Format gespeichert werden, sodass es mit einem Programm, wie zum Beispiel Google SketchUp[2], weiter bearbeitet oder auf einem 3D-Drucker ausgedruckt werden konnte.

[2] http://sketchup.google.com/intl/de/

Literaturverzeichnis

[Arno Becker 2009] Arno Becker, Marcus P.: *Android Grundlagen und Programmierung*. Copyright © 2009 dpunkt.verlag GmbH, 2009

[Ebers 2004] Ebers, Olga: *Überblick über aktuelle Verfahren zur Tiefenschätzung aus 2D-Video-Sequenzen*. Studienarbeit an der TU Berlin, Institut für Telekommunikationssysteme, FG Nachrichtenübertragung, 2004

[Erschenburg 2006] Erschenburg, Jonas: Optisches Kameratracking anhand natürlicher Merkmale. In: *http://www.informatik.uni-augsburg.de/lehrstuehle/hcm/publications/2006-TR-13-esc/tr2006-13.pdf* (2006)

[Ethan Rublee 2011] Ethan Rublee, Vincent Rabaud Kurt Konolige Gary R. B.: ORB: An efficient alternative to SIFT or SURF. In: *International Conference on Computer Vision* (2011). http://www.willowgarage.com/papers/orb-efficient-alternative-sift-or-surf

[Faugeras 1993] Faugeras, O.: Three-dimensional computer vision: A geometric viewpoint. In: *The MIT press, Cambridge, MA* (1993)

[Harris u. Stephens 1988] Harris, C. ; Stephens, M.: *A combined corner and edge detector*. Plessey Research Roke Manor, United Kingdom © The Plessey Company, 1988. – 147–151 S.

[Hartley u. Zisserman 1997] Hartley, R.I. ; Zisserman, A.: *Multiple View Geometry in Computer Vision*. Second. Cambridge University Press, ISBN: 0521540518, 1997

[Hartley u. Sturm 1997] Hartley, Richard I. ; Sturm, Peter: Triangulation. (1997). http://users.cecs.anu.edu.au/~hartley/Papers/triangulation/triangulation.pdf

[http://unhive.com 2010] http://unhive.com/special/statistiken/

[http://www.bitkom.org/ 2012] http://www.bitkom.org/de/presse/8477_72316.aspx

[Huttenlocher u. Ullman. Nove] Huttenlocher, D. ; Ullman., S.: Recognizing solid objects by alignment with an image. (November 1990). Int.J.ofComputerVision,5(2):195\T1\textendash212,

[Konushin 2011] Konushin, Anton: Strukture for Motion. (2011). http://courses.graphicon.ru/files/courses/vision2/2011/lectures/cv2011_18_sfm_web.pdf

[Kutulakos u. Seitz 1999] Kutulakos, K. ; Seitz, S.: *A theory of shape by space carving.* In Proc. Sevent Int. Conf. on Computer Vision, 1999

[Lowe May] Lowe, D.: Fitting parameterized three-dimensional models to images. In: *EEE Trans. on Pattern Analysis and Machine Intelligence, 13(5):441–450* (May 1991)

[Lowe 2004] Lowe, David G.: Distinctive Image Features from Scale-Invariant Keypoints. In: *http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.2.8899* (2004)

[M. Calonder u. Fua. 2010] M. Calonder, V. Lepetit C. S. ; Fua., P.: Brief: Binary robust independent elementary features. In: *European Conference on Computer Vision* (2010)

[Muja u. Lowe 2009] Muja, Marius ; Lowe, David G.: Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration. In: *International Conference on Computer Vision Theory and Application VISSAPP'09)*, INSTICC Press, 2009, 331-340

[Pollefeys 1999] Pollefeys, Marc: *Self-Calibration and metric 3D reconstruction from uncalibrated image sequences.* http://unc.academia.edu/MarcPollefeys/Papers/51024/_pdf_. Version: 1999

[Rosten u. Drummond 2006] Rosten, E. ; Drummond, T.: Machine learning for highspeed corner detection. In: *European Conference on Computer Vision* (2006)

[Szeliski 2011] Szeliski, R.: *Computer Vision: Algorithms and Applications.* Springer-Verlag London Limited, 2011

[Wu] Wu, Changchang: *VisualSFM.* http://www.cs.washington.edu/homes/ccwu/

[www.stanford.edu 2012] www.stanford.edu: *Opencv für Android.* http://www.stanford.edu/class/ee368/Android/Tutorial-2-OpenCV-for-Android-Setup-Windows.pdf. Version: 2012

[Zhang 2000] Zhang, Zhengyou: *A Flexible New Technique for Camera Calibration.* http://research.microsoft.com/en-us/um/people/zhang/Papers/TR98-71.pdf. Version: 2000

Tabellenverzeichnis

Abbildungsverzeichnis

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 30. August 2012 Aleksej Dygoduk