Internal Report 06–01

Inkrementelle Akquisition von 3D-Objektmodellen

by

Lars Heyden

Ruhr-Universität Bochum
Institut für Neuroinformatik
44780 Bochum

IR-INI 06–01
April 2006
ISSN 0943-2752

© 2006 Institut für Neuroinformatik, Ruhr-Universität Bochum, FRG

Diplomarbeit von Lars Heyden

Inkrementelle Akquisition von 3D-Objektmodellen

LS 7 (Graphische Systeme)
Fachbereich Informatik
Universität Dortmund

erstellt am
Institut für Neuroinformatik
Ruhr-Universität Bochum

13. Dezember 2005

1. Gutachter: Prof. Dr. Heinrich Müller
2. Gutachter: Dr. Rolf Würtz

1. Betreuerin: Dr. Gabriele Peters
2. Betreuer: Dr. Rolf Würtz

Inhaltsverzeichnis

Kapitel 1

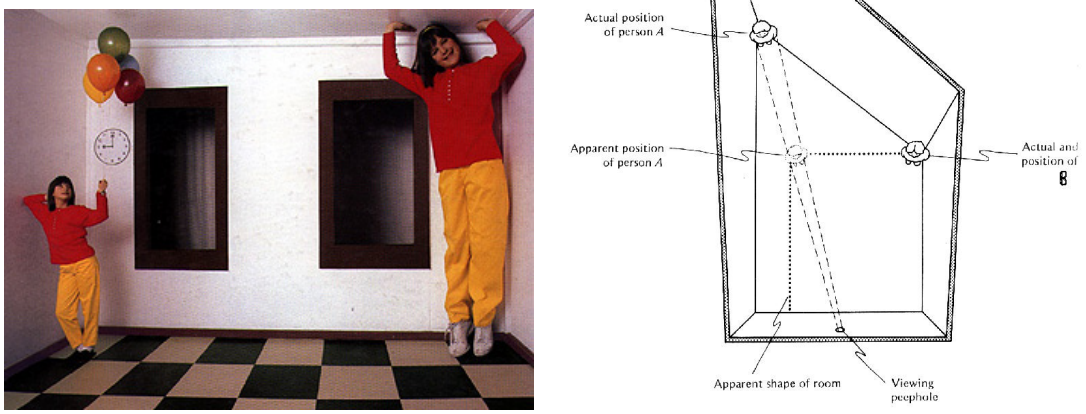
Einleitung

1.1 Hintergrund und Motivation

Thema dieser Arbeit ist das Gewinnen von Informationen über die dreidimensionale Welt aus zweidimensionalen optischen Informationen. Dieser Prozess ist etwas, das jeder Mensch sehr gut beherrscht. Auch unser optischer Sinn erhält zunächst nur zweidimensionale Informationen, nämlich die Bilder auf der Netzhaut. Dennoch haben wir eine Wahrnehmung einer dreidimensionalen Welt, die uns umgibt. Genauer gesagt erhalten Menschen, da sie zwei Augen haben, doppelte zweidimensionale Informationen, so dass die binokulare (aus den Unterschieden zwischen den Netzhautbildern gewonnene) Information aus beiden Augen zum räumlichen Sehen beiträgt – jedoch bleibt unsere Wahrnehmung der Welt auch dreidiemsional, wenn wir nur ein Auge benutzen, und wir nehmen auch in zweidimensionalen Bildern und Filmen eine dreidimensionale Welt wahr. Tatsächlich sind die binokularen nur einige von sehr vielen Informationen, die das Gehirn benutzt, um ein dreidimensionales Bild der Welt zu erstellen. Ausführliche Erläuterungen dazu finden sich in [Gol02].

Selten machen wir uns bewusst, dass die Wahrnehmung unserer Welt keine logische, eindeutige Konsequenz aus den zweidimensionalen Informationen, die unsere Augen erhalten, ist. Es ist lediglich eine Interpretation dieser Informationen – eine Interpretation, die in der realen Welt fast immer zutreffend ist. Durch optische Täuschungen jedoch können wir sie überlisten. Eine bekanntes Beispiel ist der in Abbildung 1.1(a) zu sehende Amessche Raum, in dem sich ein Zwillingspaar befindet – das sich offensichtlich sehr in der Größe unterscheidet. Man mag zunächst vermuten, dass es sich nur um eine Fotomontage handeln kann, tatsächlich interpretieren wir jedoch die Form des Raumes völlig falsch, indem wir ihn als rechteckig wahrnehmen – denn dies ist die schlüssigste Interpretation der zweidimensionalen Information, die das Bild uns liefert. Die tatsächliche Form des Raumes unterscheidet sich von der wahrgenommenen Form erheblich und ist in Abbildung 1.1(b) zu sehen. Zu näheren Informationen über diese optische Täuschung siehe [Gol02] oder [ZG96]. Ein weiteres Beispiel für den Effekt der falsch wahrgenommenen räumlichen Umgebung sind die eindrucksvollen und bekannt gewordenen Pflastermalereien von Julian Beever, die in Abbildung 1.2 zu sehen sind.

Man muss sich klar machen, dass diese Täuschungen keine Unzulänglichkeiten des Sehsinns sind, sondern die direkte Konsequenz aus unserer Fähigkeit, blitzschnell aus



(a) (b)

Abbildung 1.1: Der Amessche Raum

(a) (b)

Abbildung 1.2: Das gleiche Motiv (allerdings an unterschiedlichen Orten) aus verschiedenen Blickwinkeln betrachtet. Aus einem bestimmten Blinkwinkel, aus dem das linke Bild aufgenommen wurde, entsteht eine räumliche Illusion

nicht eindeutigen Informationen ein schlüssiges Modell unserer Umgebung zu erstellen. Durch sie wird uns bewusst, dass der trivial erscheinende Vorgang der alltäglichen Wahrnehmung weitaus komplexer ist, als man es zunächst vermutet – und das dient zur Erklärung, warum Wahrnehmung – das heißt hier also insbesondere das Transformieren der optischen 2D-Informationen zu einem 3D-Modell – zu den faszinierenden Problemen gehört, die ein Mensch ohne Schwierigkeiten lösen kann, in denen Computer dagegen ganz schlecht abschneiden. Der Prozess der Wahrnehmung ist hochkomplex, auch weit entfernt davon völlig erforscht zu sein, und schließt umfangreiches und abstraktes Wissen über die Welt mit ein, die ein Rechner schlicht und einfach nicht hat.

Das automatisierte Erstellen von 3D-Modellen kann heute bereits sehr genau geschehen, indem auf das zu erfassende Objekt bestimmte feine Muster projiziert und anschließend abgefilmt werden, so dass die Verzerrungen dieses Musters Informationen über die Form geben ([Bre03]). In meiner Arbeit geht es jedoch darum, alleine optische Informationen ohne weitere Hilfsmittel zu benutzen. Das große Thema, in dem ich diese Arbeit also einordnen möchte, ist das Erfassen der dreidimensionalen Welt, so wie Menschen es spielend machen, mit den Informationen, die auch der Mensch benutzt: mit zweidimensionalen Abbildungen dieser Welt. Das umfangreiche Weltwissen, das der Mensch dabei außerdem benutzt, bleibt aber zunächst außen vor.

1.2 Ziel der Arbeit

Anstatt gleich die dreidimensionale Beschaffenheit der ganzen Welt oder zumindest einer kompletten Umgebung zu erfassen, fangen wir kleiner an und versuchen, ein dreidimensionales Modell eines kleinen Gegenstandes zu erstellen. Dieser Prozess entspricht dabei nicht dem Erkennen einer bekannten dreidimensionalen Form sondern der Betrachtung eines zuvor unbekannten Gegenstandes. Dieses möchte ich deshalb unterstreichen, weil unsere alltägliche Wahrnehmung zum Großteil eine Wahrnehmung einer bekannten Umgebung und von bekannten Gegenständen ist, und wir meistens auf unser Wissen zurückgreifen können anstatt tatsächlich ein neues 3D-Modell zu erstellen, wie wir es hier von dem System erwarten.
Unser Modell wird aus einer Wolke von Raumpunkten bestehen. Dabei wollen wir ausschließlich optische Informationen benutzen, die aus einer einfachen Webcam gewonnen werden. Damit ist der erste Schritt vorgegeben: das Erstellen einer Bildsequenz des Objektes aus den Bildern dieser Webcam. Dabei werden wir (außer für die Segmentierung) kein Wissen über die Kamerapositionen und internen Parameter der Kamera verwenden. Dafür nehmen wir in Kauf, dass wir lediglich ein projektives Modell erstellen können (die berechneten Raumpunkte werden also gegenüber den tatsächlichen Raumpunkten projektiv verzerrt sein) – wie wir sehen werden, ist mit den gegebenen Informationen auch nicht mehr möglich. Wir werden allerdings herausstellen, mit welchen zusätzlichen Informationen wir ein metrisches Modell erstellen könnten.
Der Titel dieser Arbeit verspricht eine inkrementelle Erstellung des Modells. Das hier vorgestellte System arbeitet inkrementell, indem es mit jedem Bild, das verarbeitet wird, das Modell erweitert und ggf. anpasst. Jedoch werden wir sehen, dass das System aus zahlreichen abgrenzbaren Arbeitschritten besteht und dass jeder Arbeitsschritt für alle Bilder durchgeführt wird, bevor mit dem nächsten Arbeitsschritt begonnen wird – so ist die Arbeit auch gegliedert und für das Verständnis ist diese Vorgehensweise sicher am geeignetsten. Prinzipiell ist es aber genauso möglich, für jedes Bild – soweit nötig (wir werden sehen, dass nicht alle Bilder in allen Arbeitschritten verwendet werden) – alle Arbeitsschritte durchzuführen, bevor das nächste betrachtet wird. Das Ergebnis bleibt das gleiche – eine Ausnahme stellt nur der letzte Arbeitsschritt, das Bundle Adjustment, dar. Hier kann die Entscheidung, ob er inkrementell oder abschließend auf allen Daten durchgeführt wird, einen großen Einfluß auf das Ergebnis nehmen.
Das System ließe sich so erweitern, dass auch die Bildakquisition inkrementell durchgeführt werden könnte. Somit könnte während des Erstellens des Modells entschieden werden, welche Bereiche des Modells abgefilmt werden müssen um das Modell sinnvoll zu vervollständigen.

1.3 Resourcen

Das hier vorgestellte System wurde am Institut für Neuroinformatik der Ruhr-Universität Bochum entwickelt. Es war mir somit möglich, die dort vorhandenen Resourcen zu nutzen. Diese umfassen insbesondere die vorhandene Hardware, die zum Abfilmen der Objekte verwendet wurde (wie zum Beispiel den Roboterarm – siehe Kapitel 3), sowie die Software-Bibliothek FLAVOR (siehe hierzu [RPEvdM99]). Aus FLAVOR (kurz für „Flexible Library for Active Vision and Object Recognition“) habe ich neben Methoden zur Bildrepräsentation und mathematischen Methoden wie Matrixzerlegungen vor allem auch Methoden zum Tracking von Bildpunkten (siehe Kapitel 5) benutzt.
Als „ideelle“ Resource diente – natürlich neben der sonstigen angegebenen Literatur – vor allem [HZ04], in der die grundlegende Vorgehensweise des überwiegenden Teils des Systems sowie auch spezielle angewandte Verfahren behandelt werden.
Das System wurde komplett in C++ entwickelt.

1.4 Überblick über die Arbeit und das System

In dieser Einleitung habe ich versucht, das Thema meiner Diplomarbeit darzulegen und einzuordnen. In Kapitel 2 gehe ich auf mathematische Grundlagen ein, die für das Verständnis der Arbeit wichtig sind. Die anschließenden Kapitel spiegeln den Arbeitsablauf des Systems wider, den ich an dieser Stelle vorab zusammenfassen möchte.

Der Arbeitsablauf des Systems lässt sich in folgende Schritte gliedern:

- Abfilmen des Objektes von allen Seiten
- Entfernen des Hintergrundes in allen Bildern
- Tracking einzelner Punkte auf dem Objekt von Bild zu Bild
- Berechnen der Epipolargeometrie zwischen aufeinander folgenden Bildern
- Berechnen von Raumpunkten durch Triangulierung
- Zusammenfügen aller Raumpunkte zu einem einheitlichen, projektiven Modell
- Verbesserung des Modells durch Bundle Adjustment

Der Gegenstand, von dem ein dreidimensionales Modell erstellt werden soll, wird zunächst abgefilmt. Hierzu wird er auf eine Drehscheibe gelegt. Die Kamera ist am Ende eines Roboterarmes befestigt. Während die Drehscheibe rotiert und der Roboterarm die Höhe und Ausrichtung der Kamera verändert, kann das Objekt von allen Seiten abgefilmt werden. Hierzu werden Bilder an fest definierten Positionen entlang eines Scanpfades aufgenommen. Wir erhalten eine Sequenz von Bildern, die das Objekt aus jeweils leicht verändertem Blickwinkel zeigen. Jedes Bild ist mit einer eindeutigen Position auf einer „Scan-Hemisphäre“ assoziiert (Details siehe 3).
Zusätzlich zu den auf diese Weise gewonnen Sequenzen wird eine Sequenz aus Hintergrundbildern erstellt. Dazu werden der gleiche Scanpfad durchlaufen und in den gleichen Abständen Bilder aufgenommen wie zuvor, jedoch befindet sich kein Gegenstand auf der

Drehscheibe. Diese Sequenz kann dazu benutzt werden, aus den anderen Sequenzen den Hintergrund zu entfernen. In allen Bildern werden die Bereiche, die mit dem entsprechenden Bild (d.h., das aus der gleichen Position auf der Scan-Hemisphäre aufgenommen wurde) aus der Hintergrund-Sequenz übereinstimmen, als Hintergrund gewertet und entfernt. Dieses Verfahren wird in Kapitel 4 beschrieben.
Wir erhalten nun also Sequenzen von Bildern, die den jeweiligen Gegenstand ohne Hintergrund zeigen. Auf diesen Sequenzen findet das Tracking statt. Zwei aufeinanderfolgende Bilder einer Sequenz zeigen das Objekt aus leicht verändertem Blickwinkel. Ein bestimmter Punkt auf dem Gegenstand wird dadurch in beiden Bildern an leicht veränderten Positionen wiedergegeben. Diese Positionen werden im folgenden korrespondierende Bildpunkte genannt. Beim Tracking wird versucht, zu einem Bildpunkt den korrepondierenden Bildpunkt in nächsten Bild zu finden. In Kapitel 5 wird das hier verwendete Tracking-Verfahren erläutert.
Mit Hilfe der korrespondierenden Punkte können wir die Epipolargeometrie zwischen verschiedenen Bildern berechnen. Wissen wir von einer gewissen Anzahl an Punkten, an welche Stelle sie in verschiedenen Bildern der Sequenz projiziert werden, erhalten wir dadurch Informationen über die Veränderung der Kameraparameter – d.h., wir erhalten Kameramatrizen, die Position, Ausrichtung und interne Parameter (siehe dazu Abschnitt 2.3) der Kameras angeben. Diese Informationen sind jedoch nicht eindeutig (siehe Abschnitte 6.4 und 7.1).
Kennen wir die Kameramatrizen, können wir per Triangulierung (Abschnitt 7.2) eine Position der Punkte im Raum bestimmen. Da wir jedoch nur eine der möglichen Kameramatrix-Kombinationen benutzen, sind die so gefundenen Raumpunkte ebenso mehrdeutig, genauer gesagt unterscheiden sie sich durch eine projektive Transformation von den tatsächlichen Raumpunkten. Zudem unterscheiden sich alle Rekonstruktionen, die wir aus jeweils zwei Bildern der Sequenz gewonnen haben, untereinander um eine projektive Transformation.
Im nächsten Schritt (Abschnitt 7.3) müssen also alle Rekonstruktionen in Übereinstimmung gebracht werden, indem eine Rekonstruktion als Referenz genommen wird und alle übrigen Rekonstruktionen durch projektive Transformationen daran angepasst werden. So erhalten wir eine Rekonstruktion aller Punkte, die in mindestens zwei der ausgewählten Bilder zu sehen waren.
Wir haben nun eine 3D-Punktwolke und Kameramatrizen für alle Bilder berechnet. Dann können wir für jedes Bild berechnen, an welche Stelle jeder berechnete Punkt unseres Modells projiziert wird. Wir kennen auch den tatsächlichen, gemessenen Bildpunkt, an den der berechnete Punkt projiziert werden sollte. Somit können wir einen Fehler der Rekonstruktion berechnen.
Wir stellen also eine Fehlerfunktion auf und versuchen, um unsere Rekonstruktion zu optimieren (Kapitel 8), diese durch das Bundle Adjustment (Abschnitt 8.2) mit iterativen Verfahren (Levenberg-Marquardt-Algorithmus) zu minimieren. Als Alternative und Ergänzung werden wir auch einen evolutionären Algorithmus zur Minimierung verwenden (Abschnitt 8.3). Das Ergebnis dieses Verfahrens hängt wesentlich davon ab, wie gut die initiale Rekonstruktion war. Beispielhafte Ergebnisse sind in Kapitel 9 dargestellt. Im optimalen Fall erhalten wir eine stimmige, für alle Bilder korrekte Rekonstruktion der 3D-Punkte, die jedoch weiterhin um eine projektive Transformation von den tatsächlichen Positionen verschieden ist. Möglichkeiten, diese Mehrdeutigkeit aufzuheben, werden in Abschnitt 7.1 und in Kapitel 10 angedeutet, sind jedoch nicht mehr Teil dieses Systems.
Zum Abschluss dieser Einleitung möchte ich darauf hinweisen, dass ich in jedem Modul

des Systems versucht habe darauf zu achten, dass Aufwand und Leistung in vernünftigem Verhältnis zueinander stehen. Jedes Teil des Systems kann an sich Thema für einzelne Arbeiten diesen (oder auch erheblich größeren) Umfanges sein. Um den Gesamtumfang bewältigen zu können, habe ich jeweils möglichst wenig aufwändige Lösungen gewählt, solange sie ausreichend gute Ergebnisse geliefert haben. An anderer Stelle, insbesondere beim Tracking, konnte ich auf existierende Lösungen aus der FLAVOR-Bibliothek zurückgreifen. Aufgrund der modularen Struktur des Systems wäre es recht einfach, einzelne Module durch aufwändigere Pendants zu ersetzen. Da diese Verfahren meist auch einen höheren Rechenaufwand mit sich bringen, ist jedoch die einfachste Variante nicht immer die schlechteste. Das System soll möglichst die eigenen Schwächen in den einzelnen Arbeitsschritten offen legen können um ein sinnvolles, gezieltes Weiterentwickeln an diesen Punkten zu ermöglichen. Im Ausblick (Abschnitt 10.2) werde ich darauf näher eingehen und außerdem die mögliche Fortsetzung des Weges, den diese Arbeit beschreitet, aufzeigen.

Kapitel 2

Mathematischer Hintergrund

In diesem Kapitel möchte ich kurz die grundlegenden mathematischen Ideen darlegen, die Basis für diese Arbeit sind. Zunächst werde ich die projektive Geometrie als Erweiterung der euklidischen Geometrie erläutern. Anschließend werden projektive Transformationen beschrieben sowie Projektionen vom 3-dimensionalen in den 2-dimensionalen Raum und umgekehrt.

2.1 Projektive Geometrie

Anstatt die projektive Geometrie formal zu definieren, möchte ich sie hier als einfache Erweiterung der bekannten euklidischen Geometrie einführen (nach [HZ04]). Eine formalere Einführung bietet beispielsweise [BR92].

2.1.1 Uneigentliche Punkte

In der euklidischen 2-dimensionalen Geometrie schneiden sich zwei Geraden immer in einem Punkt – es sei denn, sie sind parallel. Durch das Aufheben dieser Ausnahme erweitern wir die euklidische Geometrie zur projektiven Geometrie. Dies geschieht durch das Hinzufügen von Punkten im Unendlichen. Diese Punkte heißen *uneigentliche Punkte*. Aus $\mathbf{R}^2$ geht auf diese Weise die projektive Ebene $\mathbf{P}^2$ hervor.
In $\mathbf{P}^2$ schneiden sich alle Geraden. Parallele Geraden schneiden sich in uneigentlichen Punkten. Anschaulich kann man sich den $\mathbf{P}^2$ so vorstellen, dass er von einem unendlich großen Kreis umgeben ist, auf dem die uneigentlichen Punkte liegen. Dieser „Kreis“ ist die *Gerade im Unendlichen* $\mathbf{l}_\infty$[1]. Analog geht aus dem euklidischen Raum $\mathbf{R}^3$ durch Hinzufügen von uneigentlichen Punkten der projektive Raum $\mathbf{P}^3$ hervor. Ihn kann man sich so vorstellen, dass er von einer unendlich großen Sphäre (Kugeloberfläche) umgeben ist, auf dem die uneigentlichen Punkte liegen. Diese „Sphäre“ ist die *Ebene im Unendlichen* Π_∞. Im $\mathbf{P}^3$ schneiden sich parallele Ebenen in uneigentlichen Linien.
In der projektiven Geometrie werden die uneigentlichen Punkte generell wie alle anderen

[1] Die Vorstellung von $\mathbf{l}_\infty$ als Kreis darf allerdings nicht zu der falschen Annahme führen, dass eine Gerade diesen „Kreis“ in zwei gegenüberliegenden Punkten schneidet. Tatsächlich schneidet jede Gerade $\mathbf{l}_\infty$ in genau einem Punkt, der durch die Richtung der Gerade bestimmt wird. Die beiden „gegenüberliegenden“ Schnittpunkte einer Gerade mit dem gedachten Kreis sind identisch.

Punkte betrachtet (das Konzept der Parallelität gibt es dadurch nicht mehr). Bei Problemen des Computersehens ist es jedoch, da wir uns mit dem realen Anschauungsraum beschäftigen, manchmal nötig, sie als Sonderfälle zu behandeln.

2.1.2 Homogene Koordinaten

Im euklidischen n-dimensionalen Raum wird ein Punkt durch ein n-Tupel von reellen Zahlen repräsentiert, beispielsweise im 2-dimensionalen Raum durch (x, y). Im projektiven Raum reicht diese Darstellung nicht aus, da damit keine uneigentlichen Punkte repräsentiert werden können. Daher werden hier *homogene Koordinaten* eingeführt.
Ein Punkt mit *inhomogenen Koordinaten* (x, y) wird hier als (kx, ky, k) dargestellt mit einer beliebigen reellen Zahl $k \neq 0$. Es wird also für jedes k der gleiche Punkt repräsentiert. Beispielsweise lässt sich der Punkt (1,2) in homogenen Koordinaten durch (1,2,1) oder auch durch (2,4,2) darstellen. Punkte werden also durch eine *Äquivalenzklasse von Koordinatenvektoren* bzw. einen Repräsentanten dieser Äquivalenzklasse dargestellt.
Um inhomogene Koordinaten (x, y) in homogene Koordinaten umzuwandeln, kann einfach $k = 1$ gewählt werden und wir erhalten die Koordinaten $(x, y, 1)$. Somit lassen sich alle Punkte des euklidischen Raumes in homogenen Koordinaten darstellen.
Um aus homogenen Koordinaten inhomogene Koordinaten zu erhalten, müssen wir die ersten beiden Koordinaten jeweils durch die letzte Koordinate teilen. Es lassen sich sich somit alle Punkte in inhomogenen Koordinaten darstellen, deren letzte Koordinate ungleich 0 ist. Punkte, deren letzte homogene Koordinate 0 ist, sind die uneigentlichen Punkte.
Analog kann im 3-dimensionalen Raum ein Punkt mit inhomogenen Koordinaten (x, y, z) durch homogene Koordinaten (kx, ky, kz, k) dargestellt werden.
Allgemein bestehen homogene Koordinaten eines Objektes mit n Freiheitsgraden aus n+1 Elementen und sind Repräsentanten einer Äquivalenzklasse von Koordinatenvektoren. Repräsentiert ein Vektor $\mathbf{a}$ ein Objekt, so repräsentiert der Vektor $k\mathbf{a}$ mit $k \in \mathbf{R}_{\neq 0}$ das gleiche Objekt. Dies gilt analog auch für Matrizen, die sich auf homogene Darstellungen eines Objektes beziehen. Auch für diese *homogenen Matrizen* ist die Skalierung irrelevant und eine $n \times m$ Matrix mit nm Einträgen hat $nm - 1$ Freiheitsgrade.

2.1.2.1 Darstellung von Geraden in homogenen Koordinaten

Im 2-dimensionalen Raum kann eine Gerade durch eine Gleichung

$$ax + by + c = 0$$

beschrieben werden und somit durch einen Vektor $\mathbf{l} = (a, b, c)^T$. Dies ist eine homogene Darstellung dieser Gerade. Da die Gleichung $ax + by + c = 0$ äquivalent ist zu jeder Gleichung $kax + kay + kc = 0$ (für $k \neq 0$), repräsentieren auch alle Vektoren $(ka, kb, kc)^T$ die gleiche Gerade.
Sei $\mathbf{x}$ ein Punkt und $\mathbf{l}$ eine Gerade, beide definiert in homogenen Koordinaten, so liegt $\mathbf{x}$ genau dann auf $\mathbf{l}$ wenn gilt:

$$\mathbf{x}^T\mathbf{l} = 0 \tag{2.1}$$

Dies folgt direkt aus der Definition der Koordinaten. Homogene Koordinaten für Geraden im 3-dimensionalen Raum werden hier nicht betrachtet.

2.1.2.2 Darstellung von Ebenen in homogenen Koordinaten

Analog zu Geraden im 2-dimensionalen Raum können im 3-dimensionalen Raum Ebenen beschrieben werden. Eine Ebene kann dargestellt werden durch die Gleichung

$$\pi_1 x + \pi_2 y + \pi_3 z + \pi_4 = 0$$

und somit durch einen Vektor $\Pi = (\pi_1, \pi_2, \pi_3, \pi_4)^T$. Alle Ebenen $k\Pi$ mit $k \in \mathbf{R}_{\neq 0}$ repräsentieren die gleiche Ebene.
Sei $\mathbf{X}$ ein Punkt und Π eine Ebene, beide definiert in homogenen Koordinaten, so liegt $\mathbf{X}$ genau dann auf Π wenn gilt:

$$\Pi^T \mathbf{X} = 0 \tag{2.2}$$

2.2 Projektive Transformationen

2.2.1 Projektive Transformationen in 2D

Eine Projektive Transformation im $\mathbf{P}^2$ ist eine Abbildung

$$h : \mathbf{P}^2 \to \mathbf{P}^2$$

bei der gilt, dass drei Punkte $\mathbf{x}_1$, $\mathbf{x}_2$ und $\mathbf{x}_3$ genau dann auf einer Geraden liegen, wenn auch die drei Punkte $h(\mathbf{x}_1)$, $h(\mathbf{x}_2)$ und $h(\mathbf{x}_3)$ auf einer Geraden liegen.
Diese Definition ist äquivalent zu folgender Aussage:

Eine Abbildung $h : \mathbf{P}^2 \to \mathbf{P}^2$ ist genau dann eine projektive Transformation, wenn eine invertierbare 3×3 - Matrix H existiert, so dass für jeden Punkt $\mathbf{x} \in \mathbf{P}^2$ gilt:

$$h(\mathbf{x}) = H\mathbf{x}$$

wobei $\mathbf{x}$ die Darstellung des Punktes durch einen Vektor mit homogenen Koordinaten ist. Ein Beweis hierzu findet sich in [HZ04].
Aus der Aussage folgt, dass also jede projektive Transformation einfach durch

$$\mathbf{x}' = H\mathbf{x} \tag{2.3}$$

beschrieben werden kann. Da $\mathbf{x}$ und $\mathbf{x}'$ homogene Vektoren sind, ist H eine homogene Matrix, also hat H acht Freiheitsgrade und alle Matrizen kH mit $k \in \mathbf{R}_{\neq 0}$ beschreiben die gleiche Transformation.
Wenn wir bestimmte Bedingungen an die Form von H stellen (unter Verringerung der Freiheitsgrade), erhalten wir spezifische Klassen von projektiven Transformationen. Beispielsweise besteht eine *euklidische Transformation* aus einer Translation und einer Rotation. Daher hat sie nur 3 Freiheitsgrade (2 für die Translation und einen Rotationswinkel). Die zughörige Transformationsmatrix hat die Form

$$H_E = \begin{pmatrix} R & \mathbf{t} \\ \mathbf{0}^T & 1 \end{pmatrix} \tag{2.4}$$

wobei R eine 2×2 - Rotationsmatrix ist (ein Freiheitsgrad) und $\mathbf{t}$ ein 2×1-Translationsvektor (zwei Freiheitsgrade). $\mathbf{0}$ ist ein 2×1-Vektor aus Nullen.

Eine weitere spezielle Transformation, der wir in dieser Arbeit begegnen werden, ist die *Ähnlichkeits-Transformation*. Sie bewirkt gegenüber der euklidischen Transformation zusätzlich eine Skalierung. Sie kann durch eine Matrix

$$H_S = \begin{pmatrix} R & \mathbf{t} \\ \mathbf{0}^T & \lambda^{-1} \end{pmatrix} \tag{2.5}$$

ausgedrückt werden mit einem Skalierungsfaktor $\lambda \in \mathbf{R}_{\neq 0}$. Offensichtlich hat eine Ähnlichkeits-Transformation 4 Freiheitsgrade.

2.2.2 Projektive Transformationen in 3D

Analog zum 2-dimensionalen Fall kann jede Transformation im $\mathbf{P}^3$ durch eine Transformationsmatrix H beschrieben werden, so dass

$$\mathbf{X}' = H\mathbf{X} \tag{2.6}$$

$\mathbf{X}$ und $\mathbf{X}'$ sind hier homogene 4×1 Vektoren und H ist eine homogene 4×4 Matrix mit 15 Freiheitsgraden.
Genau wie zuvor können wir durch Bedingungen an H die Freiheitsgrade mindern und erhalten spezifische Klassen von projektiven Transformationen. Eine euklidische Transformation hat beispielsweise im 3-dimensionalen Raum 6 Freiheitsgrade (3 für die Translation und 3 Rotationswinkel), eine Ähnlichkeits-Transformation hat mit dem zusätzlichen Skalierungsfaktor 7 Freiheitsgrade. Die entsprechenden Transformations-Matrizen haben die gleiche Form wie zuvor in (2.4) bzw. (2.5), nur ist R nun eine 3×3 Rotationsmatrix mit 3 Freiheitsgraden und $\mathbf{t}$ ein 3×1 Translationsvektor mit ebenfalls 3 Freiheitsgraden.

2.3 Von 3D nach 2D

Wir betrachten nun 2-dimensionale Repräsentationen der 3-dimensionalen Welt, wie sie zum Beispiel Fotos darstellen. Es sind Projektionen, in der eine Dimension verloren geht. Modelliert wird dieser Prozess hier durch die *Zentralprojektion*. Dieses Verfahren beschreibt das Verhalten einer einfachen Lochkamera. Es approximiert aber auch Kameras mit Linsen oder das menschliche Auge. Näheres dazu siehe [FP03].
Es wird ein 3D-Punkt als Projektionszentrum $\mathbf{C}$ ausgewählt und eine Ebene im Raum als Bildebene Π. Der Schnittpunkt dieser Bildebene mit einem Strahl, die zwischen einem Raumpunkt und dem Projektionszentrum verläuft, ist die Projektion dieses Raumpunktes auf Π.
Dieser Prozess entspricht einer Abbildung von $\mathbf{P}^3$ nach $\mathbf{P}^2$. Eine solche Abbildung kann durch eine 3×4 - Matrix P mit Rang 3 dargestellt werden. Diese Matrix heißt *Kameramatrix*. Sei also $\mathbf{X}$ ein Raumpunkt und P eine Kameramatrix, dann ist

$$\mathbf{x} = P\mathbf{X} \tag{2.7}$$

eine Projektion von $\mathbf{X}$ in eine durch P festgelegte Bildebene. $\mathbf{X}$ und $\mathbf{x}$ sind dabei wieder homogene Vektoren und so ist auch eine Kameramatrix homogen und hat im Allgemeinen 11 Freiheitsgrade. Diese 11 Freiheitsgrade können folgendermaßen erklärt werden:

- 3 Freiheitsgrade geben die Position des Projektionszentrums $\mathbf{C}$ im Raum an.
- 3 Freiheitsgrade geben die Ausrichtung der Kamera an, das heißt die Lage der Bildebene. Sie kann durch drei Rotationswinkel bzw. eine Rotationsmatrix R angegeben werden.
- 2 Freiheitsgrade geben die Brennweite und die Skalierung des zweidimensionalen Koordinatensystems (und damit die Form der nicht zwangsläufig quadratischen Pixel der Kamera) an. Die Brennweite – und damit der Abstand der Bildebene zum Projektionszentrum – wird durch einen Parameter f angegeben, die Skalierungen des Koordinatensystems durch 2 Parameter m_x und m_y (entsprechend der Anzahl an Pixeln pro Einheitslänge in der jeweiligen Richtung). Diese Parameter sind aber nicht unabhängig (und entsprechen daher nur 2 Freiheitsgraden) und werden ersetzt durch zwei Parameter $\alpha_x = fm_x$ und $\alpha_y = fm_y$, welche die Pixelgröße der jeweiligen Richtung in Bezug zur Brennweite angeben.
- 2 Freiheitsgrade geben die Koordinaten p_x und p_y in der Bildebene an, an der die optische Achse (also die Gerade, die senkrecht auf der Bildebene steht und durch $\mathbf{C}$ läuft) die Bildebene schneidet. Auch diese Koordinaten werden in Bezug zur Brennweite f gesetzt und durch $x_0 = fp_x$ und $y_0 = fp_y$ angegeben
- 1 Freiheitsgrad, parametrisiert durch s, gibt die Schrägheit zwischen den Koordinatenachsen der Bildebene an.

Ausführlichere Informationen über die Kameraparameter sind in [FP03] und [HZ04] zu finden.
Position und Ausrichtung werden *externe Kameraparameter* genannt, die übrigen fünf Parameter *interne Kameraparameter*.
Wir wollen nun betrachten, wie sich die Parameter in der Kameramatrix wiederfinden. In [HZ04] wird gezeigt, dass sich die Kameramatrix zerlegen lässt in

$$P = [\ M\ |\ -M\widetilde{\mathbf{C}}\ ] = K[\ R\ |\ -R\widetilde{\mathbf{C}}\ ] \tag{2.8}$$

Diese Notation gibt eine zusammengesetzte Matrix an. Hier besteht die 3×4 Matrix P aus dem linken 3×3 Block M und dem rechten 3×1 Block $-M\widetilde{\mathbf{C}}$, und es gilt $M = KR$. R die Rotationsmatrix, welche die Ausrichtung der Kamera angibt, $\widetilde{\mathbf{C}}$ sind die inhomogenen Koordinaten des Projektionszentrums und K ist eine 3×3 - Matrix, welche die internen Kameraparameter angibt. K hat 5 Freiheitsgrade und ist folgendermaßen aufgebaut:

$$K = \begin{pmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{pmatrix}$$

Sind die Parameter bekannt, kann P also einfach berechnet werden. Ist andererseits P bekannt, muss die Matrix zerlegt werden um K, R und $\mathbf{C}$ bestimmen zu können.
Das Projektionszentrum $\mathbf{C}$ spannt den rechten Nullraum von P auf:

$$P\mathbf{C} = \mathbf{0}$$

Es kann somit beispielsweise mit einer Singulärwertzerlegung der Matrix (siehe Abschnitt A.1.2) berechnet werden.

M ist der linke 3×3 Block von P und es gilt $M = KR$, wobei die Kalibrationsmatrix K eine obere Dreiecksmatrix ist und die Rotationsmatrix R orthogonal. Unter der zusätzlichen Bedingung, dass K positive diagonale Einträge hat, kann M mit Hilfe der RQ-Zerlegung (siehe Abschnitt A.2) eindeutig in K und R zerlegt werden.
Einen Sonderfall stellen Kameras dar, deren Projektionszentrum $\mathbf{C}$ auf Π_∞, der Ebene im Unendlichen, liegt. Dann existiert kein $\widetilde{\mathbf{C}}$ und die Zerlegung ist auf diese Weise nicht möglich. $\mathbf{C}$ selbst kann allerdings wie zuvor durch $P\mathbf{C} = \mathbf{0}$ gefunden werden.

Wir stellen an dieser Stelle fest, dass die Einträge der Kameramatrix tatsächlichen physikalischen Gegebenheiten entsprechen. Wir gehen in den folgenden Kapiteln davon aus, dass all diese Werte *unbekannt* sind. Dies ist insbesondere für die Werte der Kalibrationsmatrix K ein folgenschwere Entscheidung. Es gibt Kalibrationsverfahren (siehe [HZ04]), um eine Kamera zu kalibrieren, d.h., ihre Kalibrationsmatrix zu erfahren. Es wurde darauf bewusst verzichtet.
Wir werden stattdessen in Abschnitt 6.4 Kameramatrizen berechnen und benutzen, die mit den tatsächlichen physikalischen Werten nicht viel gemeinsam haben. Wie wir sehen werden, hat das zur Folge, dass wir nur ein projektives Modell eines 3D-Objektes erstellen werden, das allerdings völlig unabhängig von der Kalibrierung der Kamera ist. Eine ausführliche Diskussion dieser Problematik folgt in Kapitel 7.1.

2.4 Von 2D nach 3D

Im letzten Abschnitt wurde gezeigt, wie ein Punkt im Raum auf eine Ebene projiziert werden kann. $\mathbf{l}$ sei die Gerade, die einen Punkt $\mathbf{X}$ mit dem Projektionszentrum $\mathbf{C}$ verbindet. Die Projektion $\mathbf{x}$ von $\mathbf{X}$ ist der Schnittpunkt von $\mathbf{l}$ mit der Bildebene. Ist nur $\mathbf{x}$ bekannt, so kann $\mathbf{X}$ nicht eindeutig bestimmt werden. Wir wissen lediglich, dass $\mathbf{X}$ auf der Geraden $\mathbf{l}$, die von $\mathbf{x}$ zurückprojiziert wird, liegt. Es ist also offensichtlich, dass in einem Bild nicht genügend Informationen vorhanden sind, um die 3D-Struktur zu rekonstruieren. In Kapitel 7 werde ich zeigen, wie man die Informationen zweier Bilder kombinieren kann, um (bei unbekannten Kameramatrizen) zumindest eine projektive Rekonstruktion von Raumpunkten zu finden.

Kapitel 3

Generierung der Bildsequenz

Ziel dieses Moduls ist es, eine Bildsequenz von einem Objekt zu erzeugen. Diese Bildsequenz sollte dazu geeignet sein, ein Modell des abgefilmten Objektes zu erstellen. Daher soll dieses von allen Seiten aus gleicher Entfernung abgefilmt werden können. Auf das Filmen der Unterseite wird jedoch verzichtet. Die Kamera sollte also Aufnahmen machen können von Punkten aus, die in gleichem, geeigneten Abstand zum Objekt liegen und sich mindestens auf der Höhe des Objekts befinden. Diese Punkte bilden eine Halbkugel, die im folgendem „Scan-Hemisphäre“ genannt wird. Alle Positionen, von denen aus die Kamera eine Aufnahme macht, sollten eindeutig definiert sein. Die Kamera soll jeweils genau auf das Objekt ausgerichtet sein.

3.1 Modellierung der Scan-Hemisphäre

Die genannten Anforderungen wurden durch die Scan-Hemisphäre modelliert. Sie ist eine gedachte Halbkugel, in deren Mittelpunkt sich das Objekt befindet. Die Kamera soll entlang dieser Halbkugel bewegt werden können und sich an jedem Punkt zum Mittelpunkt ausrichten. Der Radius wird dabei so gewählt, dass sie dabei geeignete Aufnahmen vom Objekt machen kann (das Objekt also komplett und möglichst bildfüllend aufgenommen wird).
Die Positionen auf der Halkugel lassen sich durch sphärische Koordinaten angeben. Dabei wird ein Punkt durch die Winkel Φ und Θ bestimmt, wobei der Φ den Winkelabstand zu einem fest definierten Nullmeridian und Θ den Winkelabstand zum Äquator angibt. Ein Punkt P wird in sphärischen Koordinaten in der Form $P(\Phi|\Theta)$ notiert. Der Schnittpunkt von Äquator und Nullmeridian ist der Nullpunkt N.
Die Scan-Hemisphäre lässt sich im dreidimensionalen euklidischen Raum eindeutig definieren durch die Angabe des Nullpunktes N (durch den auch der Nullmeridian festgelegt wird) und des Mittelpunktes M, oder, wie hier geschehen, durch N und den Punkt $T(0°|90°)$, der den *Nordpol* der Hemisphäre darstellt. Es lassen sich dann zu allen Punkten $P(\Phi|\Theta)$ die kartesischen Raumkoordinaten und die entsprechende Ausrichtung der Kamera berechnen. Die Ausrichtung der Kamera umfasst den Drehwinkel um die optische Achse (eine Dimension) sowie deren Richtung (2 Dimensionen). Im Folgenden werden Position und Ausrichtung zusammengefasst, wenn von *6-dimensionalen Raumkoordinaten* die Rede ist. Punkte auf der Hemisphäre werden also durch 2-dimensionale sphärische Koordinaten parametrisiert und lassen sich eindeutig auf die 6-dimensionalen Raumkoordinaten abbilden.

Dies wird genauer beschrieben in Abschnitt 3.2.2.

3.2 Umsetzung

Der Aufbau zum Abfilmen des Objektes umfasst neben einer handelsüblichen Webcam, mit der die Aufnahmen gemacht werden, eine Drehscheibe und einen Roboterarm. Die Kamera wird am Ende des Roboterarms befestigt. Das Objekt wird auf die Mitte der Drehscheibe gelegt und von dieser gemäß des gewählten Winkels Φ ausgerichtet. Der Roboterarm bringt die Kamera auf die dem Winkel Θ entsprechende Position und richtet sie auf das Objekt aus. Auf diese Weise kann jeder Punkt auf der Scan-Hemisphäre angesteuert werden. Es kann nun ein Scanpfad angegeben werden, d.h. eine Kette von Positionen mit recht geringem Abstand. Die Aufnahmen, die von diesen Positionen gemacht werden, bilden die gewünschte Bildsequenz.

3.2.1 Steuerung der Drehscheibe

Die Drehscheibe besteht aus einem einfachen Drehteller, der mit einem Motor an die gewünschte Position gefahren werden kann. Sie hat eine Auflösung von 1010 Schritten pro 180 Grad und einen maximalen Drehwinkel von etwa 330 Grad. Eine einfache Schnittstelle zur Steuerung wird in FLAVOR bereitgestellt. Die Drehscheibe lässt sich im Bereich zwischen einer festen Grundposition bis zum maximalen Winkel durch die Angabe der gewünschten Schritte steuern.
Die aktuelle Position der Drehscheibe wird in der Anzahl von Schritten ab Grundposition gespeichert. Soll ein Winkel Φ angesteuert werden und ist s_0 die aktuelle Position der Drehscheibe in Schritten, so muss die Drehscheibe um $\Delta s = \Phi \cdot \frac{1010}{180^\circ} - s_0$ Schritte bewegt werden. Δs wird dazu auf eine Ganzzahl gerundet. Aufgrund der Auflösung beträgt die maximale Abweichung der erreichten von der gewünschten Position, die dadurch verursacht werden kann, dem halben Winkelabstand eines Schrittes, also etwa $0,09^\circ$. Die gewünschte Position wird also nicht exakt erreicht, aber die erreichte Position kann exakt ermittelt werden und wird exakt reproduziert.
Die Nullpunkt N der Scan-Hemisphäre entspricht der Grundposition der Drehscheibe von 0 Schritten. Dann kann zu jedem Φ die benötigte Drehung des Drehtellers wie beschrieben einfach berechnet werden. Leider ist die Drehscheibe wie erwähnt zu keiner kompletten Drehung um 360 Grad fähig, sondern hat einen maximalen Drehwinkel von 330 Grad. Daher war es notwendig, die fehlenden 30 Grad mit einer zusätzlichen Bewegung des Roboterarms abzudecken.

3.2.2 Steuerung des Roboterarms

Der benutzte Roboterarm ist in [Sch04] beschrieben. Er hat sieben Gelenke, die durch Verbindungsstücke miteinander verbunden sind. Jedes Gelenk hat einen Rotationsfreiheitsgrad und kann durch einen Elektromotor bewegt werden.
Die Schnittstelle zum Roboter ist der *Roboter-Server* (ausführlich beschrieben in [Sch04]). Er ermöglicht das Abfragen und Ändern aller Gelenkstellungen und somit das Bewegen des Roboterarms. Zusätzlich kann er die 7-dimensionalen Gelenkkoordinaten in 6-dimensionale Raumkoordinaten umrechnen. Die Raumkoordinaten umfassen drei

Koordinaten, die die Position des Endstücks des Roboterarms im Raum angeben und weitere drei Koordinaten, die dessen Ausrichtung angeben. Durch den zusätzlichen Freiheitsgrad der Gelenke können Raumpositionen durch verschiedene Gelenkstellungen erreicht werden, was die Flexibilität des Armes erhöht. Zudem ist der zusätzliche Freiheitsgrad dadurch motiviert, dass auch der menschliche Arm über sieben Freiheitsgrade verfügt.
Am Ende des Armes befindet sich normalerweise ein Greifer, für dieses System wird dort aber eine Kamera befestigt. Es ist also möglich, diese Kamera nun an beliebige Raumkoordinaten zu bringen, sofern sie in seiner Reichweite liegen, und beliebig auszurichten. Das Koordinatensystem ist dabei fest vom Roboter vorgeben.

3.2.2.1 Festlegung der Scan-Hemisphäre

Die Scan-Hemisphäre wird mit Hilfe der kartesischen Roboterkoordinaten definiert. Der Roboterarm wird auf gleiche Höhe wie das Objekt gefahren und die Kamera darauf ausgerichtet. Diese Position wird nun als Nullpunkt N definiert. Anschließend wird der Roboterarm genau über das Objekt gefahren und die Kamera nach unten ausgerichtet. Diese Position gibt den Nordpol T an, der jedoch automatisch noch so korrigiert wird, dass der Abstand zum Mittelpunkt M dem Anstand von N zu M entspricht. M kann zuvor berechnet werden aus der x- und y-Koordinate von T und der z-Koordinate von N. Auf diese Weise wird sichergestellt, das wir eine Halbkugel mit ebenerdiger Grundfläche erhalten.

3.2.2.2 Berechnung der Raumkoordinaten in Abhängigkeit von Θ

Wir gehen nun zunächst davon aus, dass Φ vollständig von der Drehscheibe abgedeckt werden kann. Das heißt, die Position des Roboterarms ist nur von Θ abhängig und alle Punkte, die der Arm anfahren muss, liegen auf dem Viertelkreis zwischen N und T. Die Raumkoordinaten für den gewählten Winkel werden folgendermaßen berechnet:

Der Viertelkreis zwischen N und T wird als Teil eines Einheitskreises aufgefasst und $h = \sin(\Theta)$ sowie $w = \cos(\Theta)$ geben die kartesischen Koordinaten des gesuchten Punktes auf diesem Kreis an. $\mathbf{N} = (n_1, n_2, n_3)$ seien die bekannten Raumkoordinaten des Nullpunktes, $\mathbf{T} = (t_1, t_2, t_3)$ die bekannten Raumkoordinaten des Nordpols und $\mathbf{X} = (x_1, x_2, x_3)$ die gesuchten Raumkoordinaten des Zielpunktes. Dann gilt

$$x_1 = n_1 + (1 - w) \cdot (t_1 - n_1)$$

$$x_2 = n_2 + (1 - w) \cdot (t_2 - n_2)$$

$$x_3 = n_3 + h \cdot (t_3 - n_3)$$

Die Ausrichtung der Kamera wird durch drei Rotationswinkel angegeben. Die Winkel $A_N = (\alpha_{N1}, \alpha_{N2}, \alpha_{N3})$ für den Nullpunkt und $A_T = (\alpha_{T1}, \alpha_{T2}, \alpha_{T3})$ für den Nordpol

sind bekannt. Die Winkel $A_X = (\alpha_{X1}, \alpha_{X2}, \alpha_{X3})$ an der Zielposition werden nun einfach interpoliert durch

$$\alpha_{Xi} = (1 - \frac{\Theta}{90°}) \cdot \alpha_{Ni} + \frac{\Theta}{90°} \cdot \alpha_{Ti} \tag{3.1}$$

3.2.2.3 Berechnung der Raumkoordinaten in Abhängigkeit von Φ

Da die Drehscheibe sich nur bis zu einer Stellung von 330° ab Grundposition drehen lässt, muss der Roboterarm ihr für $\Phi > 330°$ entgegen kommen.
Der Winkel, den der Roboterarm abdecken muss, beträgt $\Delta\Phi = \Phi - 330°$. Zunächst wird der Punkt $N'(\Delta\Phi|0°)$ auf dem Äquator berechnet. $\mathbf{M} = (m_1, m_2, m_3)$ seien die Raumkoordinaten des Mittelpunkts M der Scanhemisphäre, r sei der Radius. $\mathbf{N'} = (n'_1, n'_2, n'_3)$ seien die Raumkoordinaten von N'. Φ_x sei der Winkel zwischen der x-Achse des durch den Roboter vergegebenen Koordinatensystems und der Geraden zwischen N und M. Φ_y sei der Winkel zwischen der y-Achse des durch den Roboter vergegebenen Koordinatensystems und der Geraden zwischen N und M. Dann gilt

$$n'_1 = m_1 + r \cdot \cos(\Phi_x)$$

$$n'_2 = m_2 + r \cdot \sin(\Phi_y)$$

$$n'_3 = m_3$$

Um die Kamera korrekt auszurichten, können die Winkel hier nicht einfach nach (3.1) berechnet werden, sondern wir müssen einen genaueren Blick auf die Parameterisierung der Ausrichtung werfen. Der Winkel α_1 gibt die horizontale Ausrichtung, der Winkel α_2 die vertikale Ausrichtung an. Der Winkel α_3 beschreibt die Rotation des Endstücks des Armes um die eigene Achse.
Im Punkt N' ist lediglich α_1 von Punkt N verschieden und es gilt

$$\alpha_{N'1} = \alpha_{N1} + \Delta\Phi$$

Somit ist Punkt N' vollständig berechnet. Analog berechnen wir einen Punkt T', der sich von T nur durch den veränderten Rotationswinkel

$$\alpha_{T'1} = \alpha_{N'1}$$

unterscheidet[1].
Der gesuchte Zielpunkt X kann nun exakt wie in Abschnitt 3.2.2.2 beschrieben berechnet werden, indem der Viertelkreis zwischen N' und T' betrachtet wird.

[1] Diese Anpassung der Koordinaten ist abhängig von der Befestigung der Kamera am Roboterarm. Hier wurde die Kamera rechtwinklig zum Arm (und damit zur Rotationsachse des Endstücks) angebracht, da so der benötigte Bewegungsspielraum der Kamera sichergestellt werden konnte. Die restliche Argumentation ist davon unabhängig.

3.2.3 Steuerung der Kamera

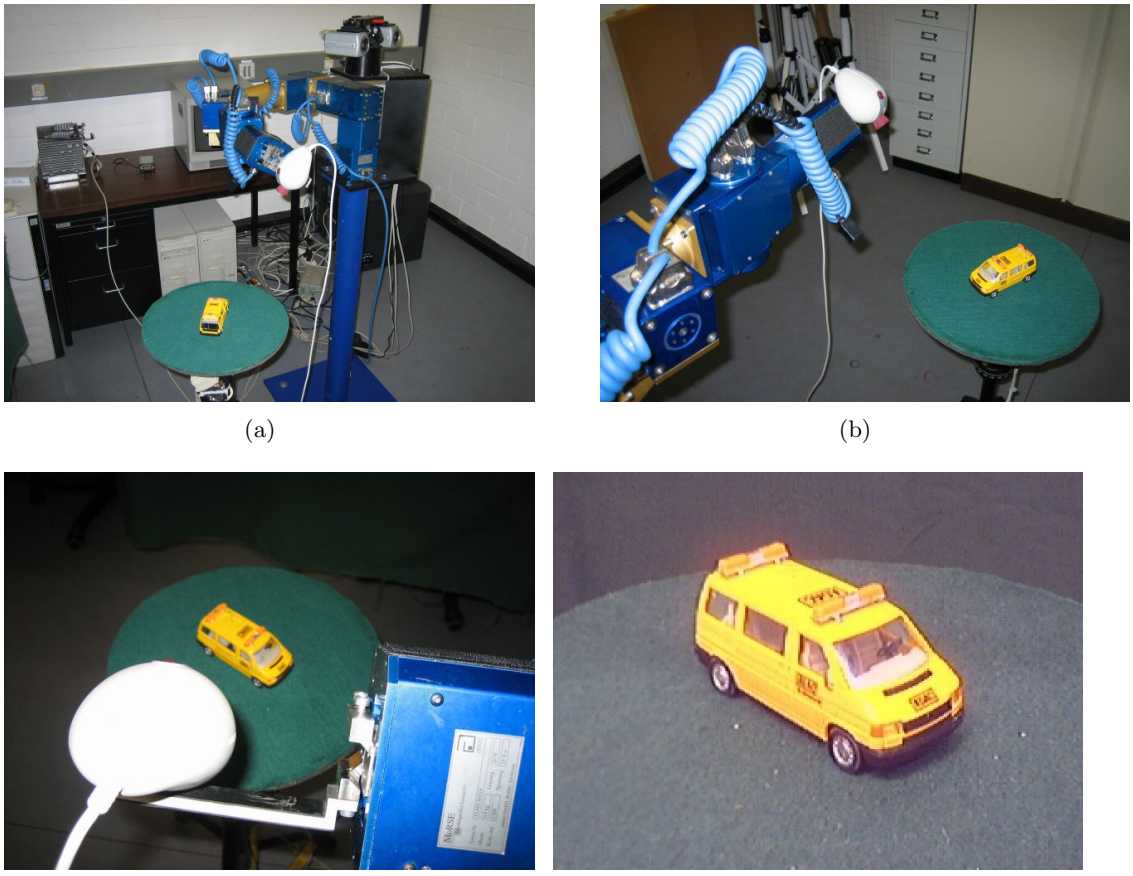
Die verwendete Kamera ist eine einfache Webcam vom Typ Philips Vesta Pro Scan (PCVC690K). Gesteuert wird sie über Schnittstellen der FLAVOR-Bibliothek. Nachdem die Kamera mit dem Roboterarm an die gewünschte Stelle gefahren worden ist und die Drehscheibe das Objekt an die richtige Stelle gedreht hat, wird das aktuelle Kamerabild aufgenommen und im Bildformat TIFF gespeichert. Als Auflösung wurden 640×480 Pixel gewählt, die Farbtiefe beträgt 24 Bit.

3.2.4 Definition eines Scanpfades

Durch Angaben von Punkten in sphärischen Koordinaten kann ein beliebig langer Scanpfad erzeugt werden. Die Schnittstelle des Systems erleichtert das, indem mehrere *Scanlinien* angegeben und verknüpft werden, die jeweils durch Start- und Endpunkt definiert werden sowie durch die Anzahl der Punkte, die auf dieser Linie aufgenommen werden sollen. Die Scanlinie verläuft natürlich nicht gerade durch den Raum, sondern auf der Oberfläche der Scan-Hemisphäre. Dabei wird nicht die kürzeste Verbindung auf der Oberfläche gewählt, sondern die kürzeste Verbindung in einer äquidistanten zylindrischen Projektion der Scan-Hemisphäre. Beispielsweise verläuft die Scanlinie zwischen zwei Punkten $(0°|45°)$ und $(180°|45°)$ entlang des 45. Breitengrades anstatt über den Nordpol.
Dieses Vorgehen begründet sich darin, dass nicht nur die Positionen von Punkten auf der Hemisphäre, sondern auch die damit verbundene Ausrichtung der Kamera berücksichtigt werden müssen. So nehmen die Punkte $(0°|90°)$ und $(180°|90°)$ die gleiche Position im Raum ein, die Ansicht ist aber um 180° rotiert, was zu völlig unterschiedlichen Bildern führt. Der Abstand dieser Punkte ist als ebenso groß zu werten wie der Abstand aller Punkte $(0°|\alpha)$ und $(180°|\alpha)$ für beliebige Winkel α, was uns unmittelbar zur äquidistanten zylindrischen Projektion führt. Zudem hat dies den Vorteil, dass zwei Punkte auf der Scan-Hemisphäre in der äquidistanten zylindrischen Projektion durch eine eindeutige Gerade verbunden sind, die dann also als Scanlinie gewählt wird.

3.2.5 Scannen und Speichern der Bildsequenz

Das Objekt wird nun an allen vorgegebenen Punkten abfotografiert und das jeweilige Bild abgespeichert. Zusätzlich werden in einer XML-Datei alle relevanten Daten des Scans gespeichert. Hier wird zu jedem Bild der Dateiname der Bilddatei und die Position auf der Scanhemisphäre, von der aus das Bild aufgenommen wurde, verzeichnet. Dies ermöglicht die Weiterverarbeitung der Sequenz in den nächsten Schritten.
In Abschnitt 1.2 wurde betont, dass diese Informationen jedoch *nicht* benutzt werden. Tatsächlich werden sie auch lediglich für die Segmentierung verwendet. Sie könnten aber auch für andere Zwecke nützlich sein. Wenn beispielsweise das System zur Erstellung von metrischen Rekonstruktionen erweitert wird, könnten sie zur Überprüfung von berechneten Kamerapositionen dienen. Allgemein sollen damit möglichst umfassende Möglichkeiten zur Nutzung der Bildsequenz sichergestellt werden.
In Abbildung 3.1 ist der Aufbau des Scans zu sehen.



(a) (b)

(c) (d)

Abbildung 3.1: Ein Spielzeugauto wird abgefilmt. Rechts unten eine Aufnahme der Kamera.

Kapitel 4

Segmentierung

Im ersten Schritt des Systems wurde eine Bildsequenz unseres Objekts erstellt. Allerdings ist auf allen Bildern nicht nur das Objekt selbst zu sehen, sondern auch die Drehscheibe und sonstiger Hintergrund. Diesen wollen wir nun entfernen, denn wir wollen ja nur das Objekt und nicht die Umgebung modellieren.
Das Segmentieren, also die Abgrenzung verschiedener Objekte in einem Bild voneinander, oder, wie in diesem Fall, die Abgrenzung eines Objektes vom Hintergrund, ist für sich ein komplexes Thema. Menschen haben damit unter normalen Bedingungen wenig Schwierigkeiten, Maschinen können es bisher nur sehr schlecht. Im Falle dieses Systems können wir jedoch mit einem sehr einfachen Verfahren brauchbare Ergebnisse erzielen.

4.1 Background Subtraction

Da wir für jede Aufnahme, die wir gemacht haben, die exakte Position der Kamera kennen, können wir von der gleichen Position aus eine weitere Aufnahme machen, in der wir das Objekt zuvor entfernen und nur den Hintergrund aufnehmen. Mit Hilfe dieser Hintergrundaufnahmen können wir in den Aufnahmen mit Objekt die Hintergrundbereiche identifizieren, es sind nämlich offensichtlich die Bereiche, die in beiden Bildern übereinstimmen. Dieses wird für jedes Pixel anhand seiner Farbwerte einzeln überprüft. Aufgrund des Kamerarauschens werden wir für ein Pixel, das in beiden Fällen zum Hintergrund gehört, nicht exakt die gleichen Farbwerte erhalten, jedoch sehr ähnliche Werte. Daher setzen wir einen Grenzwert fest. Liegt die Differenz der Farbwerte unterhalb dieses Grenzwertes, kennzeichnen wir das Pixel als Hintergrundpixel und setzen es auf die Hintergrundfarbe.
Durch das Rauschen der Kamera gibt es natürlich einzelne Pixel mit größeren Abweichungen. Um dieses Problem zu entschärfen hat es sich bewährt, an jedem Pixel auch die Farbwerte der anliegenden Pixel zu berücksichtigen. Es werden also nicht die Werte eines einzelnen Pixels verglichen, sondern mit geringerer Gewichtung auch die Farbwerte der Umgebung des Pixels.

4.1.1 Wahl des Hintergrundes

Voraussetzung für den Erfolg dieses Verfahrens ist offensichtlich, dass der Hintergrund vollkommen unverändert bleibt und die Kameraposition sehr exakt reproduziert werden kann. Zudem darf der Hintergrund an keiner Stelle die gleichen Farbwerte wie das Objekt haben. Daher ist es ratsam, einen einfarbigen Hintergrund zu wählen, dessen Farbton in den zu scannenden Objekten nicht vorkommt.

4.1.2 Weißabgleich und andere Farbunterschiede

Obwohl der Farbton eines Objektes wesentlich von der Beleuchtung abhängt, sieht er für den Menschen in unterschiedlichem Licht ziemlich gleich aus, da das menschliche Auge sich daran anpasst (für eine ausführliche Erläuterung dieses Effektes siehe [Gol02]). Auch Kameras können sich inzwischen daran anpassen, indem sie einen Weißabgleich durchführen. Der Weißabgleich führt zu einer Farbkorrektur im gesamten Bild und kann je nach gefilmtem Objekt völlig unterschiedlich ausfallen. Insbesondere können auch die Farbwerte eines Hintergrundbildes von den Farbwerten des Objektbildes abweichen. Daher ist dieses Verhalten für die Segmentierung nicht erwünscht.
Anstatt den Weißabgleich zu vermeiden, habe ich den dadurch entstandenen Unterschied in den Farbwerten aus den Bildern herausgerechnet. Dies ist möglich, da der Weißabgleich ausschließlich zu Unterschieden in den durchschnittlichen Intensitäten der einzelnen Farbkanäle führt. Um diese zu ermitteln, ist es nötig, einen Bereich zu kennen, der bis auf diese Unterschiede in beiden Bildern identisch ist. Da das Objekt mittig platziert ist, kann ein Bereich in einer Ecke des Bildes gewählt werden. In diesem Bereich wird in beiden Bildern für jeden Farbkanal die durchschnittliche Intensität gemessen und verglichen. Das Verhältnis der Intensitäten für jeden Farbton wird nun bei der Segmentierung berücksichtigt, das heißt ein Farbwert wird als „gleich“ angesehen, wenn er in jedem Farbkanal im dort gemessenen Verhältnis abweicht. Dieses recht simple Verfahren hat problemlos funktioniert und kann auch sonstige (also nicht durch den Weißabgleich entstandene) Farbunterschiede, die das gesamte Bild betreffen, beseitigen oder zumindest vermindern (z. B. könnten geringe Unterschiede durch flimmerndes Licht entstehen. Leuchtstoffröhren werden üblicherweise mit der normalen Wechselstromfrequenz von 50 Hertz betrieben und haben die Eigenschaft, in dieser Frequenz zu flimmern – für den Menschen nicht oder kaum wahrnehmbar, aber in mit kurzer Belichtungszeit aufgenommenen Bildern kann dieses Flimmern einen Unterschied bewirken). Zudem bietet dieses Verfahren den Vorteil, dass im anschließend weiterverarbeiteten Bild die durch den Weißabgleich bewirkte Farbkorrektur erhalten bleibt.
Können wir keinen Bereich im Bild angeben, der mit Sicherheit zum Hintergrund gehört, ist es möglich, verschiedene Bereiche auszuprobieren. Wir wählen einen Bereich und berechnen aufgrund seiner Farbwerte die Intensitätsverhältnisse. Liegt der gewählte Bereich komplett im Hintergrund, so sind diese Intensitätsverhältnisse richtig und er müsste komplett als Hintergrund identifiziert werden. Ist dies nicht der Fall, müssen wir einen anderen Bereich auswählen und wiederholen den Vorgang.

4.2 Clustern der Vordergrundpixel

Mit dem oben beschriebenen Verfahren lässt es sich nicht vermeiden, dass dennoch Reste des Hintergrundes im segmentierten Bild zu finden sind. Gründe sind beispielsweise minimale Änderungen des Hintergrundes oder zu starkes Kamerarauschen. Ein typisches Ergebnis der Segmentierung kann daher so aussehen wie in Abbildung 4.1(c). Hier sind zahlreiche kleine Gruppen von Pixeln übrig geblieben.
Wir können nun das tatsächliche Objekt als den Bereich des Bildes identifizieren, der aus den meisten zusammenhängenden Vordergrundpixeln besteht. Alle anderen kleinen Gruppen von Vordergrundpixeln können dann als Störungen identifizert und dem Hintergrund zugerechnet werden. Wir müssen also zusammenhängende Vordergrundpixel zu Clustern zusammenfassen.
Um diese Idee praktisch umzusetzen, können wir alle Pixel in geordneter Reihenfolge von oben nach unten und links nach rechts durchlaufen und, ist das aktuelle Pixel an Position (x, y) ein Vordergrundpixel, folgende Fallunterscheidung treffen:

- Ist genau eines der beiden Pixel $(x-1, y)$ und $(x, y-1)$ ein Vordergrundpixel, zugeordnet zu Cluster l, und der andere ein Hintergrundpixel, so ordne (x, y) Cluster l zu.
- Sind beide Pixel $(x-1, y)$ und $(x, y-1)$ Vordergrundpixel und beide dem gleichen Cluster l zugeordnet, so ordne (x, y) Cluster l zu.
- Sind beide Pixel $(x-1, y)$ und $(x, y-1)$ Vordergrundpixel und unterschiedlichen Clustern l und m zugeordnet, so vereinige die Cluster l und m zu Cluster l' und füge auch (x, y) diesem Cluster zu.
- Ist keiner der Pixel $(x-1, y)$ und $(x, y-1)$ ein Vordergrundpixel, lege einen neuen Cluster l an und füge den Pixel (x, y) diesem hinzu.

Abschließend wird der größte Cluster ermittelt und alle Vordergrundpixel, die einem anderen Cluster zugeordnet sind, als Hintergrund markiert und auf die Hintergrundfarbe gesetzt.

4.3 Umsetzung

In der zu einer Bildsequenz gehörenden XML-Datei sind alle Bilder der Sequenz verzeichnet. Außerdem ist die XML-Datei der zugehörigen Hintergrundsequenz angegeben. Da zu jedem Bild die sphärischen Koordinaten der Aufnahme gespeichert sind (siehe Abschnitt 3.2.5), kann jedes Bild dem richtigen Hintergrundbild zugeordnet werden. Dann werden mit dem in Abschnitt 4.1.2 beschriebenen Verfahren die globalen Intensitätsunterschiede in den Farbwerten der Bilder berechnet und unter deren Berücksichtigung die Segmentierung durchgeführt. Die Vordergrundpixel werden gleichzeitig wie in Abschnitt 4.2 beschrieben zu Clustern zusammengefasst. Anschließend werden alle Pixel, die nicht dem größten Cluster angehören, dem Hintergrund zugeordnet. Wir erhalten eine Sequenz von Bildern, in denen der Hintergrund auf eine einheitliche Hintergrundfarbe, beispielsweise weiß, gesetzt worden ist.

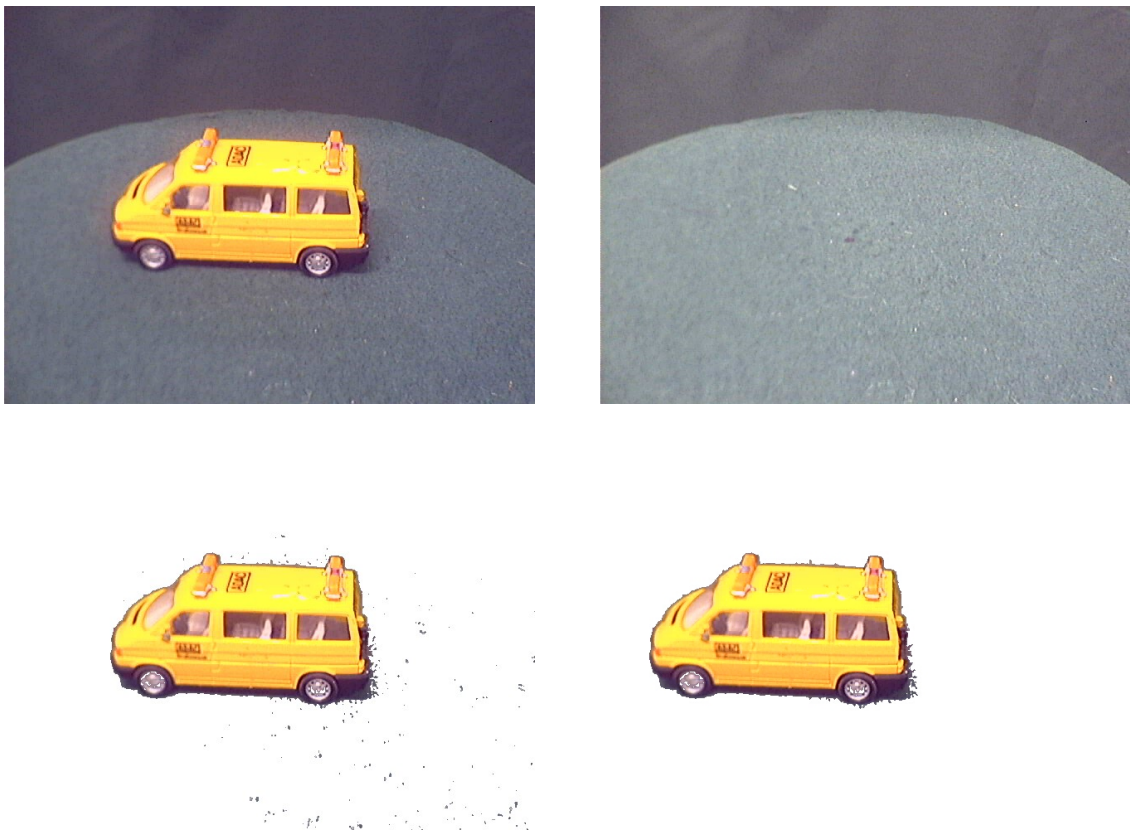


Abbildung 4.1: Ein Bild aus der Sequenz mit dem zugehörigen Hintergrundbild. In der unteren Reihe ist das Ergebnis der Segmentierung zu sehen, links das Zwischenergebnis nach der Background Subtraction, rechts ist nach Anwendung der Methode aus Abschnitt 4.2 der Hintergrund bis auf einen Rand und den Schatten vollständig entfernt.

In Abbildung 4.1 sehen wir ein Bild aus der Sequenz mit dem zugehörigen Hintergrundbild. Deutlich sind die Farbunterschiede im Hintergrund zu sehen. Diese stellten jedoch durch die Anwendung der Methode aus Abschnitt 4.1.2 bei der Segmentierung kein Problem dar. Beispielsweise ist der Hintergrund oberhalb der Drehscheibe komplett herausgerechnet worden. Problematisch war dagegen die unregelmäßige Oberfläche der Drehscheibe. Hier gibt es leider einige übrig gebliebene Fragmente im segmentierten Bild. Wie in Abbildung (d) zu sehen, konnten diese mit der Methode aus Abschnitt 4.2 vollständig entfernt werden. Das Problem, das wir mit einfachen Mitteln nicht in der Griff kriegen können, ist der Schatten des Objekts. Es zeigte sich, dass dieser aber keine gravierenden Folgen für die Weiterverarbeitung der Sequenz hatte.

Kapitel 5

Erzeugen von Punktkorrespondenzen durch Tracking

Ziel diesen Kapitels ist es, eine Anzahl von Punktkorrespondenzen zwischen den Bildern einer Sequenz zu erzeugen.
Sieht ein Mensch eine Bildsequenz (einen Film) eines sich bewegenden Gegenstandes, so kann er einen Punkt auf dem Gegenstand fixieren und der Bewegung des Punktes folgen. In jedem Bild kann er dadurch problemlos die Position dieses Punktes angeben. Prinzipiell ist dies auch einem Computer möglich, und auf diese Weise wollen wir die Punktkorrespondenzen finden.
Das hier verwendete Tracking-Verfahren werde ich nur relativ knapp beschreiben, da es im Wesentlichen nicht Teil meiner Arbeit ist, sondern ich hier vorhandene Funktionen der FLAVOR-Bibliothek benutzt habe.
Das Tracking findet auf Grauwert-Bildern statt, die Farbbilder der Sequenz werden also vorab entsprechend umgewandelt.

5.1 Auswahl der Punkte

Um Punkte aus den Bildern auszuwählen, die getrackt werden sollen, wird das Differenzbild aus zwei aufeinanderfolgenden Bildern der Sequenz erstellt. Es werden dann zufällig Bildpunkte ausgewählt, an denen – anhand des Differenzbildes ermittelbar – eine Veränderung stattgefunden hat. Die Hoffnung ist, dass diese Bildpunkte sich relativ zur Kamera bewegende Raumpunkte repräsentieren und ausreichend Struktur haben, um zum Tracking geeignet zu sein. Es wird darauf geachtet, dass neue Punkte nicht zu nah an bereits exisitierenden Punkten liegen.
Dieses Verfahren wird nach jedem Trackingschritt (d.h., nach jedem bearbeiteten Bild) benutzt, wenn die Anzahl der Punkte, die momentan getrackt werden, unterhalb einer als Parameter vorgegebenen Anzahl liegt. Da ständig Punkte entfernt werden (Erklärung dazu folgt im nächsten Abschnitt), ist das also nach fast jedem Trackingschritt der Fall.
Es gibt selbstverständlich bessere Methoden, um „interessante“, also gut geeignete, Punkte zu finden. Beispielsweise könnte ein Algorithmus wie der Ecken-Detektor von Harris

benutzt werden [HS88] oder das biologisch motivierte Verfahren von Würtz und Lourens [WL97]. Das hier benutzte Verfahren hat allerdings brauchbare Ergebnisse geliefert und hat den Vorteil, recht einfach und schnell zu sein.

5.2 Tracking eines Punktes

Voraussetzung dafür, dass ein Punkt zufriedenstellend verfolgt werden kann, ist, dass sich seine Position zwischen zwei Bildern nicht sehr verändert hat. Daher darf die Schrittweite beim Abfilmen des Objektes nicht zu groß sein, zufriedenstellende Ergebnisse brachte etwa eine Drehung des Objektes um 2° zwischen aufeinander folgenden Aufnahmen.

5.2.1 Grobe Abschätzung

Bekannt sei ein Punkt $\mathbf{x} = (x_1, x_2)^T$ im ersten Bild. Gesucht ist der korrespondierende Punkt $\mathbf{x}' = (x_1', x_2')^T$ im zweiten Bild.
Als Erstes können wir die Koordinaten grob schätzen. Da die Position sich nicht sehr verändert hat, ist eine Schätzung $\hat{\mathbf{x}}' = \mathbf{x}$ meist ausreichend. Es kann aber auch berücksichtigt werden, wie sich die Position des Punktes in vorhergehenden Trackingschritten verändert hat, und die letzten Positionsveränderungen als Prognose für die jetzige Änderung benutzt werden. Sei also $\Delta\mathbf{x}$ die Differenz von $\mathbf{x}$ zu den Koordinaten im vorhergehenden Bild, so könnten wir $\hat{\mathbf{x}}' = \mathbf{x} + \Delta\mathbf{x}$ schätzen.
Nun wird in der näheren Umgebung des Bildpunktes $\hat{\mathbf{x}}'$ die tatsächliche Position $\mathbf{x}'$ des mit $\mathbf{x}$ korrespondierenden Punktes gesucht.
Es ist nicht möglich, diese Position anhand einfacher Informationen wie der Farbwerte der Pixel zu identifizieren. Stattdessen sind komplexere „lokale Bildinformationen“, also Beschreibungen des Bildpunktes, nötig. Hierzu wird am Bildpunkt $\mathbf{x}$ eine *Gabor Wavelet Transformation* durchgeführt.

5.2.2 Gabor Wavelet Transformation

Mit einer *Fourier-Transformation* lässt sich ein Bild in seine räumlichen Frequenzen zerlegen. Erläuterungen hierzu siehe [FP03]. Es lässt sich damit feststellen, welche Frequenzen im Bild vorhanden sind. Dies sind globale Informationen, die also das gesamte Bild betreffen. Mit *Gabor-Wavelets* lässt sich eine lokale Frequenzanalyse durchführen, das heißt es lässt sich feststellen, welche räumliche Frequenzen in einem bestimmten Bereich des Bildes auftreten. Dies ist die lokale Information, die wir benutzen, um den Bildpunkt $\mathbf{x}'$ zu identifizieren. Die folgende Beschreibung der Gabor Wavelet Transformation orientiert sich an [FP03] und [Pet02].

Beispiele für Gabor-Wavelets sind in Abbildung 5.1 zu sehen. Ein Gabor-Wavelet stellt eine bestimmte räumliche Frequenz in einer bestimmten Ausrichtung dar. Die Grundlage eines Gabor-Wavelets ist also ein Grauwertbild, dessen Intensitäten in einer Richtung konstant sind und in orthogonaler Richtung durch eine Sinus- bzw. Cosinuskurve mit besagter Frequenz bestimmt werden. Dieses wird mit einer Gaußglocke multipliziert. Das Benutzen

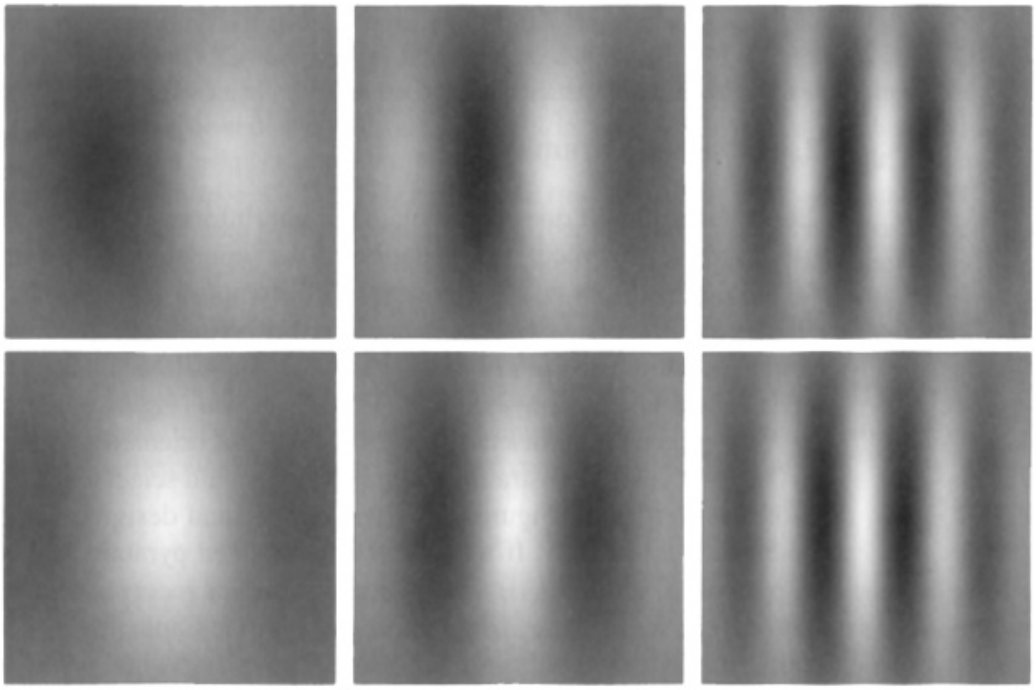


Abbildung 5.1: Einige Gabor-Wavelets unterschiedlicher Frequenz und gleicher Orientierung. In der oberen Reihe die antisymmetrischen Wavelets, unten die symmetrischen Wavelets.

der Sinuskurve führt zu einem antisymmetrischen Wavelet, das Benutzen der Cosinuskurve zu einem symmetrischen Wavelet. Meistens werden beide Varianten in Paaren benutzt.
Ein Gabor-Wavelet kann also beschrieben werden durch die Gleichungen

$$G_{symmetrisch}(\mathbf{x}) = \cos(\mathbf{k}^T\mathbf{x}) \exp\left(-\frac{\mathbf{x}^2}{2\sigma^2}\right) \tag{5.1}$$

und

$$G_{antisymmetrisch}(\mathbf{x}) = \sin(\mathbf{k}^T\mathbf{x}) \exp\left(-\frac{\mathbf{x}^2}{2\sigma^2}\right) \tag{5.2}$$

Hier gibt $\mathbf{x} = (x_1, x_2)^T$ die Bildkoordinaten an, der Parameter $\mathbf{k} = (k_1, k_2)^T$ bestimmt die Orientierung und Frequenz des Wavelets, σ ist die Varianz der Gaußfunktion und bestimmt somit die Skalierung des Filters (siehe unten). Je größer σ ist, desto größer ist der Bildbereich, der Einfluß auf das Ergebnis hat.
Alternativ lassen sich die Gabor Wavelets darstellen durch die Gleichung

$$G(\mathbf{x}) = \exp(i\mathbf{k}^T\mathbf{x}) \exp\left(-\frac{\mathbf{x}^2}{2\sigma^2}\right) \tag{5.3}$$

$G(\mathbf{x})$ ist nun komplexwertig und enthält das symmetrische Wavelet im Realteil und das antisymmetrische Wavelet im Imaginärteil.
Wir benutzen ein Gabor-Wavelet nun als Filter, den wir auf ein Bild anwenden. Wir benutzten dann den Ausdruck *Gabor-Filter* synonym für ein Gabor-Wavelet. Um die Skalierung des Gaußfilters an die Frequenz anzupassen und die Filterantwort unabhängig von

der durchschnittlichen Intensität des gesamten Bildes zu machen, erweitern wir (5.3) zu

$$G(\mathbf{x}) = \frac{\mathbf{k^2}}{\sigma^2} \exp(i\mathbf{k}^T\mathbf{x} - \exp(\sigma^2/2)) \exp\left(-\frac{\mathbf{k}^2\mathbf{x}^2}{2\sigma^2}\right) \tag{5.4}$$

Ein Filter G wird auf einen Bildpunkt $\mathbf{x}_0$ angewandt durch die *Faltung*

$$(G * I)(\mathbf{x}_0) = \int G(\mathbf{x}_0 - \mathbf{x}) I(\mathbf{x}) d^2x \tag{5.5}$$

Wird der Gabor-Filter an einen Bildpunkt auf einem Bild angewandt, erhalten wir als Antwort einen Wert, der die Stärke der in diesem Filter dargestellten Frequenz und Orientierung in der Umgebung des Bildpunktes angibt. Wir führen diesen Vorgang dann für l verschiedene Frequenzen und d Orientierungen auf einem Punkt aus, das heißt wir benutzen ld Filter und erhalten somit für jeden Bildpunkt ld Antworten. Diese Antworten sind komplexwertig und können durch Werte für die Amplitude a und Phase ϕ ausgedrückt werden, welche die betreffende Frequenz an diesem Bildpunkt hat. Diese Menge von Antworten nennen wir *Jet*. Ein Jet J besteht also aus Elementen $J_i = (a_i, \phi_i)$. Er beschreibt die Eigenschaften eines Bildpunktes in geeigneter Weise, um einen korrespondierenden Punkt mit den gleichen Eigenschaften, also einem ähnlichen Jet, zu finden.

5.2.3 Ähnlichkeitsfunktionen für Jets

Die Ähnlichkeiten von Bildpunkten kann mit der Ähnlichkeit ihrer Jets ausgedrückt werden. Eine Ähnlichkeitsfunktion für Jets ist beispielsweise

$$S_{abs}(J, J') = \frac{\sum_i a_i a_i'}{\sqrt{\sum_i a_i^2 \sum_i a_i'^2}} \tag{5.6}$$

Hier wird nur die Amplitude der Frequenz verglichen. Für nah beieinander liegende Bildpunkte liefert sie recht ähnliche Ergebnisse. Eine andere Ähnlichkeitsfunktion ist

$$S_{pha}(J, J') = \frac{1}{2}\left(\frac{\sum_i a_i a_i' \cos(\phi_i - \phi_i')}{\sqrt{\sum_i a_i^2 \sum_i a_i'^2}} + 1\right) \tag{5.7}$$

Hier werden auch die Phasen der Frequenzen verglichen. Da sich nah beieinander liegende Bildpunke in der Phase erheblich unterscheiden, liefert diese Funktion auch für solche Punkte bereits sehr unterschiedliche Werte. Der Unterschied in der Phase ist es auch, den wir bei unserem Trackingverfahren ausnutzen.

5.2.4 Finden eines korrespondierenden Punktes durch Phasenabgleich

Um den zu Punkt $\mathbf{x}$ aus dem ersten Bild korrespondierenden Punkt $\mathbf{x}'$ im zweiten Bild finden, berechnen wir zunächst die Jets J an der Stelle $\mathbf{x}$ im ersten Bild und J' an der initialen Schätzung $\hat{\mathbf{x}}'$ im zweiten Bild. Vermutlich wird unsere Schätzung nicht exakt richtig sein und $S_{pha}(J, J')$ eine geringe Ähnlichkeit ergeben. Wir versuchen nun, $\mathbf{x}'$ so zu

verschieben, dass die Phasen wieder „übereinander passen“. Wir ergänzen S_{pha} um einen Verschiebungsvektor $\mathbf{d}$, und erhalten

$$S_{disp}(J, J', \mathbf{d}) = \frac{1}{2}\left(\frac{\sum_i a_i a'_i \cos(\phi_i - \phi'_i - (\mathbf{d}^T \mathbf{k}_i))}{\sqrt{\sum_i a_i^2 \sum_i a_i'^2}} + 1\right) \tag{5.8}$$

Diese Funktion kann nun über $\mathbf{d}$ maximiert werden. Unsere Schätzung $\mathbf{x}'$ können wir dann verbessern zu einer neuen Schätzung $\mathbf{x}'_{neu} = \mathbf{x}'_{alt} + \mathbf{d}$. Wir iterieren, bis wir eine hinreichend gute Schätzung erhalten (oder ein $\mathbf{d} = \mathbf{0}$).
Die Iteration ist notwendig, da die Abschätzung von $\mathbf{d}$ nur für kleine Abweichungen präzise ist. Siehe hierzu auch [MvdM96], wo das Verfahren vorgestellt wird. Es können mit diesem Verfahren Verschiebungen bis zur halben Wellenlänge des Gabor-Filters mit der höchsten Frequenz berechnet werden.
Wird ein Punkt über mehrere Bilder verfolgt, stellt sich die Frage, ob ein Bildpunkt bzw. sein Jet mit dem Ursprungs-Jet verglichen werden soll oder mit dem Jet aus dem vorhergehenden Bild. Letzteres heißt, dass man den verbleibenden Unterschied zwischen den Jets in erster Linie nicht als durch Rauschen bewirkten Fehler, sondern als Folge der leicht veränderten Perspektive auffasst und den Jet an die dadurch bewirkten Veränderungen der Filterantworten anpasst. Ob das erwünscht ist, hängt von dem gegebenen Problem ab. Es ist auch möglich, den Jet, mit dem verglichen wird, in jedem Trackingschritt etwas an den neuen Jet anzupassen ohne ihn völlig zu übernehmen. Zu näheren Details zu diesem Verfahren siehe [Wie01] und [MvdM96].

5.3 Tracking der gesamten Punktmenge

Wir können nun in unserer Bildsequenz die Bilder nacheinander bearbeiten, indem wir zunächst wie in Abschnitt 5.1 beschrieben Bildpunkte auswählen und zu allen Punkten dann wie in Abschnitt 5.2 beschrieben einen korrespondierenden Bildpunkt im nachfolgenden Bild suchen. Zu jedem so getrackten Punkt wird die Ähnlichkeit der Bildpunkte in beiden Bildern verglichen (z.B. mit Gleichung (5.7)). Liegt sie unterhalb eines als Parameter festgesetzten Grenzwertes, wird der Punkt in den nachfolgenden Bilden nicht mehr weiterverfolgt. Da die Ähnlichkeit zu gering ist, müssen wir davon ausgehen, dass wir den Punkt, den wir eigentlich tracken wollen, verloren haben. Möglicherweise ist er im zweiten Bild nicht mehr zu sehen. Für die entfernten Punkte werden dann neue Punkte nach der Methode aus Abschnitt 5.1 ausgewählt.
Ein Punkt kann dann üblicherweise über viele Bilder der Sequenz hinweg verfolgt werden. Wir erhalten dadurch eine große Anzahl an Punktkorrespondenzen, wir kennen also zu jedem Bildpunkt eine Anzahl von Bildpunkten in anderen Bildern, die denselben Punkt im Raum abbilden.

5.4 Umsetzung

Das Tracking findet auf der in Kapitel 4 erzeugten segmentierten Bildsequenz statt. Alle Bilder der Sequenz sind in einer XML-Datei verzeichnet (siehe Abschnitt 4.3). Sie werden nun eingelesen, der Reihe nach durchlaufen und es wird, wie in Abschnitt

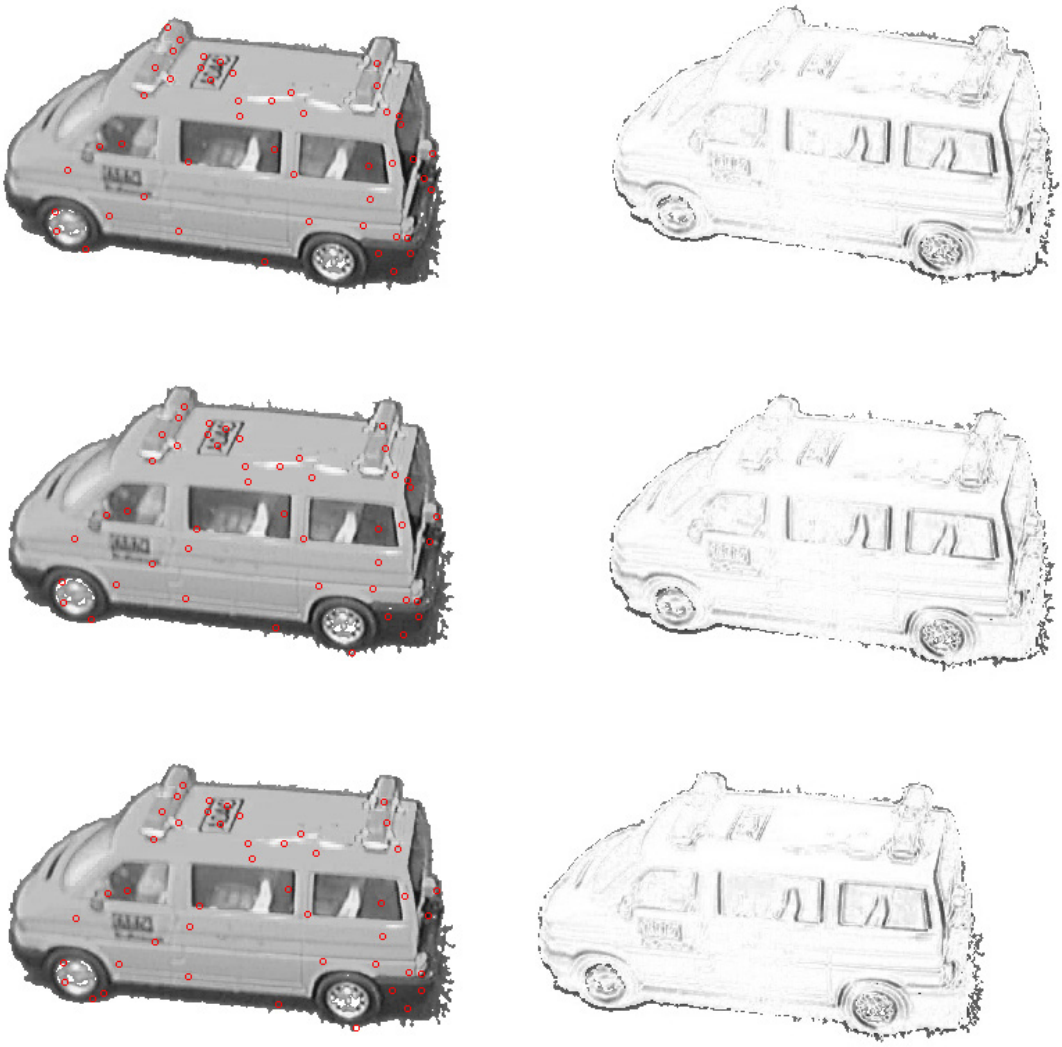


Abbildung 5.2: Auf der linken Seite sind drei aufeinanderfolgende Bilder einer Sequenz mit den getrackten Punkten zu sehen. Die rechte Seite zeigt die Differenzbilder zum jeweils vorhergehenden Bild.

5.3 beschrieben, das Tracking auf ihnen durchgeführt. Es ensteht so eine große Menge von Punktkorrespondenzen zwischen den Bildern, die in geeigneten Datenstrukturen abgespeichert werden.

Ein Auschnitt eines beispielhaften Ergebnisses ist auf der linken Seite in Abbildung 5.2 zu sehen.

Kapitel 6

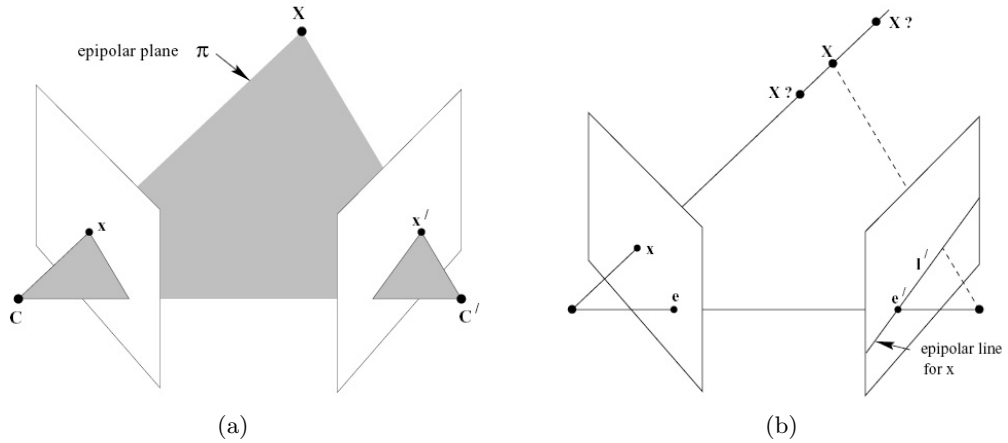
Epipolargeometrie

Wir haben nun zwischen allen Bildern unserer Sequenz eine Anzahl von Punktkorrespondenzen berechnet. Wir haben also Informationen darüber, wie die im Raum statischen Punke mit den unterschiedlichen Kameras (eine Kamera steht hier für eine Kameramatrix, die Position, Ausrichtung und interne Parameter der Kamera umfasst. Siehe Abschnitt 2.3) unterschiedlich abgebildet werden. Daraus wollen wir Informationen gewinnen, wie diese Kameras sich untereinander verändert haben und schließlich relativ zu den Kameras eine Raumposition der Punkte berechnen.
Wir betrachten im Folgenden nur zwei Bilder der Sequenz. Im Gegensatz zum Tracking müssen und sollten sie in der Sequenz nicht umittelbar aufeinander folgen (die Gründe werden in Kapitel 7 deutlich), jedoch müssen sie noch viele korrespondierende Punkte haben.
Für jedes Paar von Bildern, das wir so untersuchen, erhalten wir sowohl für die geschätzten Kameraparameter und -positionen als auch für die berechneten Raumkoordinaten der Punkte völlig unterschiedliche Ergebnisse. Der Grund dafür ist, dass die Kameramatrizen nicht eindeutig zu bestimmen sind und wir daher nur projektive Rekonstruktionen der Raumpunkte erhalten, die um projektive Transformationen voneinander abweichen. Wie diese zu einer gemeinsamen Rekonstruktion vereinheitlicht werden, wird in Kapitel 7 beschrieben.
Eine ausführlichere Erläuterung der Epipolargeometrie findet sich in [HZ04].

6.1 Einführung in die Epipolargeometrie

Die Epipolargeometrie stellt die Beziehungen zwischen zwei Ansichten einer Szene her. Gegeben seien zwei aus unterschiedlichen Blickwinkeln aufgenommene Bilder der gleichen Szene mit Kameramatrizen P und P'. Die Projektionszentren dieser Kameras seien $\mathbf{C}$ und $\mathbf{C}'$. Die Projektion $\mathbf{x}$ eines Raumpunktes $\mathbf{X}$ mit P ist also der Schnittpunkt der Geraden, die $\mathbf{X}$ und $\mathbf{C}$ verbindet, mit der Bildebene dieser Kamera. Diese Gerade ist die Rückprojektion von $\mathbf{x}$ in den Raum. Siehe hierzu auch Abschnitt 2.3.
Die Gerade $\mathbf{CC}'$ nennen wir Grundlinie. Der Schnittpunkt dieser Geraden mit der Bildebene von P ist die Abbildung von $\mathbf{C}'$ im ersten Bild, der Schnittpunkt mit der Bildebene von P' ist die Abbildung von $\mathbf{C}$ im zweiten Bild.
Diese Punkte

$$\mathbf{e} = P\mathbf{C}'$$



(a) (b)

Abbildung 6.1: Die Geometrie einer Punktkorrespondenz

und

$$\mathbf{e}' = P'\mathbf{C}$$

nennen wir *Epipole*.
Eine *Epipolarebene* ist eine Ebene, welche die Grundlinie enthält. Ein Raumpunkt $\mathbf{X}$ erzeugt also die Epipolarebene $\mathbf{XCC}'$. Siehe hierzu Abbildung 6.1(a).
Eine *Epipolarlinie* ist die Projektion einer Epipolarebene in die Bilder, also die Schnittgerade der Epipolarebene mit der jeweiligen Bildebene. Folgende Betrachtung wird für das erste Bild gemacht, gilt aber für das zweite analog:
Da $\mathbf{C}'$ in jeder Epipolarebene liegt, liegt dessen Abbildung, der Epipol $\mathbf{e}$, auf jeder Epipolarlinie. Alle Epipolarlinien schneiden sich also in $\mathbf{e}$.

Wir betrachten nun einen Raumpunkt $\mathbf{X}$. Seine Projektion in das erste Bild sei $\mathbf{x} = P\mathbf{X}$. Π sei die Epipolarebene, die von $\mathbf{C}$, $\mathbf{C}'$ und $\mathbf{X}$ aufgespannt wird. In Abschnitt 2.4 haben wir gesehen, dass wir zu einem bekannten Punkt $\mathbf{x}$ nicht feststellen können, durch welchen Raumpunkt $\mathbf{X}$ er erzeugt wurde. Wir wissen jedoch, dass er auf der Geraden liegt, die von $\mathbf{x}$ in den Raum zurückprojiziert wird. Diese Gerade liegt offensichtlich in Π. Sofern die Gerade nicht der Grundlinie entspricht, ist die Abbildung der Geraden somit genau die Abbildung von Π im zweiten Bild, also eine Epipolarlinie. Das heißt also, dass jeder Bildpunkt in einem Bild eine Epipolarlinie im anderen Bild erzeugt. Siehe hierzu Abbildung 6.1(b).
Diese Abbildung

$$\mathbf{x} \mapsto \mathbf{l}'$$

wird durch die *Fundamentalmatrix* repräsentiert.

6.2 Die Fundamentalmatrix

Die Fundamentalmatrix F ist eine 3×3 Matrix für die gilt:

$$\mathbf{l}' = F\mathbf{x} \tag{6.1}$$

und

$$\mathbf{l} = F^T \mathbf{x}' \tag{6.2}$$

wobei $\mathbf{l}$ die durch $\mathbf{x}'$ erzeugte Epipolarlinie ist und $\mathbf{l}'$ die durch $\mathbf{x}$ erzeugte Epipolarlinie, $\mathbf{x}$ und $\mathbf{x}'$ sind korrespondierende Bildpunkte (also Abbildungen desselben Raumpunktes $\mathbf{X}$). Sämtliche Punkte und Geraden sind durch homogene Koordinaten gegeben. Wir gehen hier davon aus, dass es eine derartige Matrix gibt. Eine Herleitung findet sich in [HZ04]. Da $\mathbf{x}$ auf der Geraden $\mathbf{l}$ liegt, gilt wegen (2.1) $\mathbf{x}^T\mathbf{l} = 0$ und mit (6.2)

$$\mathbf{x}^T F^T \mathbf{x}' = 0 \tag{6.3}$$

Da $\mathbf{x}'$ auf der Geraden $\mathbf{l}'$ liegt, gilt analog $\mathbf{x}'^T\mathbf{l}' = 0$ und mit (6.1)

$$\mathbf{x}'^T F \mathbf{x} = 0 \tag{6.4}$$

Wir sehen also, dass sich durch das Transponieren von F die Rollen von $\mathbf{x}$ und $\mathbf{x}'$ vertauschen.

6.2.1 Eigenschaften der Fundamentalmatrix

Aus den Eigenschaften der Fundmentalmatrix, über die wir sie gerade definiert haben, folgen weitere Eigenschaften:

- Die Fundamentalmatrix hat Rang 2. Das folgt daraus, dass sie eine lineare Abbildung aus dem 2-dimensionalen projektiven Raum (von einem Punkt $\mathbf{x}$) in den 1-dimensionalen projektiven Raum (zu einer Epipolarlinie $\mathbf{l}'$) darstellt[1]. Letzteres ist ein 1-dimensionaler Raum, da alle Epipolarlinien durch den Epipol laufen und daher eine 1-dimensionale Familie von Geraden darstellen.
  Es gilt also $\det F = 0$.

- Da $\mathbf{x}$ und $\mathbf{x}'$ sowie $\mathbf{l}$ und $\mathbf{l}'$ in homogenen Koordinaten dargestellt sind, ist auch die Fundamentalmatrix homogen. Da durch die Bedingung $\det F = 0$ ein weiterer Freiheitsgrad verloren geht, hat sie 7 Freiheitsgrade.

- Da alle Epipolarlinien den Epipol enthalten, gilt

  $$\mathbf{e}'^T F \mathbf{x} = 0 \ \forall \mathbf{x}$$

  Daraus folgt

  $$\mathbf{e}'^T F = 0 \tag{6.5}$$

  $\mathbf{e}'$ spannt also den linken Nullraum von F auf. Analog gilt, dass $\mathbf{e}$ den rechten Nullraum von F aufspannt.

[1] Anmerkung: Eine lineare Abbildung $\mathbf{P}^n \mapsto \mathbf{P}^n$ kann durch eine homogene Matrix mit Rang $n+1$ dargestellt werden, eine lineare Abbildung $\mathbf{P}^n \mapsto \mathbf{P}^{n-1}$ durch eine homogene Matrix mit Rang n. Vergleiche Kapitel 2.

6.2.2 Berechnung der Fundamentalmatrix

Gegeben sind 2 Bilder mit zahlreichen Punktkorrespondenzen. Wir werden zeigen, dass mindestens 7 Korrespondenzen notwendig sind, aber eine höhere Anzahl an Korrespondenzen für ein aussagekräftigeres Ergebnis sorgen kann.
Wir wollen nun mit Hilfe dieser Korrespondenzen die Fundamentalmatrix berechnen. Grundlage für diese Berechnung ist die Gleichung (6.4), also

$$\mathbf{x}'^T F \mathbf{x} = 0$$

die für jede Punktkorrespondenz $\mathbf{x} \leftrightarrow \mathbf{x}'$ gilt.
Sei nun $\mathbf{x} = (x, y, 1)$ und $\mathbf{x}' = (x', y', 1)$. Dann können wir die Gleichung ausschreiben zu

$$x'xf_{11} + x'yf_{12} + x'f_{13} + y'xf_{21} + y'yf_{22} + y'f_{23} + xf_{31} + yf_{32} + f_{33} = 0$$

Für n Punktkorrespondenzen erhalten wir ein Gleichungssystem mit n solcher Gleichungen, das wir durch

$$A\mathbf{f} = \mathbf{0}$$

ausdrücken können mit

$$A = \begin{pmatrix} x'_1x_1 & x'_1y_1 & x'_1 & y'_1x_1 & y'_1y_1 & y'_1 & x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x'_ix_i & x'_iy_i & x'_i & y'_ix_i & y'_iy_i & y'_i & x_i & y_i & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x'_nx_n & x'_ny_n & x'_n & y'_nx_n & y'_ny_n & y'_n & x_n & y_n & 1 \end{pmatrix}$$

und

$$\mathbf{f} = (f_{11}, f_{12}, f_{13}, f_{21}, f_{22}, f_{23}, f_{31}, f_{32}, f_{33})$$

Wir erhalten also $\mathbf{f}$ und somit F durch Lösen dieses Gleichungssystems. Zu beachten ist, dass dies ein homogenes Gleichungssystem ist. Die Skalierung der Lösung ist also beliebig und wir benötigen keine eindeutige Lösung, sondern einen 1-dimensionalen Lösungsraum. Daher reicht es aus, 8 Punktkorrespondenzen und somit 8 Gleichungen zu benutzen. Darauf beruht der 8-Punkt-Algorithmus (siehe 6.2.2.1). Außerdem muss zusätzlich sichergestellt werden, dass die Fundamentalmatrix einen Rang von 2 hat. Dies kann dazu ausgenutzt werden, mit 7 Gleichungen auszukommen (siehe 6.2.2.2).
Wir können auch mehr als 8 Gleichungen benutzen. Im Idealfall müsste die Matrix A dennoch Rang 8 behalten, da die Fundamentalmatrix nur 8 Freiheitsgrade hat. Jedoch sind die benutzten Punktkoordinaten aufgrund von Rauschen und ungenauen Messdaten sowie diskreten Bildauflösungen nicht exakt und wir erhalten im Allgemeinen ein überbestimmtes Gleichungssystem. Dann kann eine Abschätzung der Lösung völlig analog zum 8-Punkt-Algorithmus mit Hilfe der Singulärwertzerlegung gefunden werden. Die abgeschätzte Lösung kann dann nicht alle Gleichungen erfüllen – es gilt also nicht $\|A\mathbf{f}\| = 0$ – es wird aber $\|A\mathbf{f}\|$ minimert unter der Bedingung $\|\mathbf{f}\| = 1$. Siehe hierzu Abschnitt A.1.2.

6.2.2.1 Der 8-Punkt-Algorithmus

Benutzen wir 8 Punktkorrespondenzen, erhalten wir ein Gleichungssystem mit 8 Gleichungen. Wir können es mit Hilfe der Singulärwertzerlegung (siehe A.1.2) von A lösen und erhalten einen Lösungsvektor $\mathbf{f}$. Dieser spannt einen 1-dimensionalen Lösungsraum auf, das heißt für jedes $k \in \mathbf{R}_{\neq 0}$ ist $k\mathbf{f}$ eine mögliche Lösung. Die Singulärwertzerlegung liefert einen Vektor $\mathbf{f}$ mit $\|\mathbf{f}\| = 1$. Wir können nun aus $\mathbf{f}$ (bzw. $k\mathbf{f}$, es gibt aber keinen Grund ein $k \neq 1$ zu wählen) die Fundamentalmatrix F erstellen. Abschließend wird wiederum mit Hilfe der Singulärwertzerlegung F auf Rang 2 gebracht (siehe dazu A.1.1).
Damit dieser simple Algorithmus zufriedenstellende Ergebnisse liefern kann, ist es unerlässlich, die Daten, also die Bildkoordinaten, angemessen zu normalisieren. Wir wenden vor Aufstellung des Gleichungssystems auf alle Punkte $\mathbf{x}_i$ aus dem ersten Bild eine Transformation T und auf alle Punkte $\mathbf{x'}_i$ aus dem zweiten Bild eine Transformation T' an, die den Ursprung des jeweiligen Koordiantensystems auf das Mittel der jeweiligen Punktmenge verschiebt und die Koordinaten so skaliert, dass der durchschnittliche Abstand der Punkte zum Ursprung $\sqrt{2}$ beträgt.
Die Fundamentalmatrix $\hat{F}$, die wir dann berechnen, bezieht sich nun auf Punktkorrespondenzen $T\mathbf{x} \leftrightarrow T'\mathbf{x}'$. Es gilt also

$$(\mathbf{x}'^T T'^T)\hat{F}(T\mathbf{x}) = 0$$

und damit

$$\mathbf{x}'^T(T'^T\hat{F}T)\mathbf{x} = 0$$

und wir erhalten mit

$$F = T'^T\hat{F}T$$

die gesuchte Fundamentalmatrix, welche sich auf die ursprünglichen Punktkorrespondenzen $\mathbf{x} \leftrightarrow \mathbf{x}'$ bezieht .

Die Normalisierung hat folgenden Grund, der sie unverzichtbar macht:
Aufgrund von Rauschen, ungenauen Daten und der diskreten Bildauflösung erhalten wir nie eine absolut korrekte Fundamentalmatrix, sondern lediglich eine Matrix, die möglichst gut auf unsere Daten passt. Der Fehler, der durch sie entsteht, ist abhängig von der Wahl der Koordinatensysteme der beiden Bilder, mit deren Punkten sie erstellt wurde. Diese Abhängigkeit von einer willkürlichen Wahl ist an sich nicht erwünscht und wird durch die Normalisierung eliminiert. Zudem ist das Koordinatensystem, das durch die Normalisierung erzwungen wird, besonders geeignet, um zu einem möglichst guten Ergebnis zu führen. Dies wird mit Blick auf die Matrix A offensichtlich. Würden wir nicht normalisierte Daten benutzen, würden x und y sowie x' und y' den ursprünglichen Bildkoordinaten entsprechen und beispielsweise Werte um 100 annehmen. Einige Einträge der Matrix würden somit Werte im Bereich von 10^4 annehmen. Dies würde zu einer großen Verzerrung in der Bewertung der Fehler führen. Insbesondere hätte die letzte Koordinate kaum Einfluss auf das Ergebnis der Gleichungen. So würde eine Abweichung in der letzten Koordinate (z.B. $(100, 100, 0.5)$ statt $(100, 100, 1)$), die einen erheblichen Unterscheid bewirkt, völlig unterbewertet.
Durch die Normalisierung ist sichergestellt, dass alle Einträge etwa im gleichen Bereich liegen und diese Verzerrung vermieden wird.

6.2.2.2 Der 7-Punkt-Algorithmus

Der 7-Punkt-Algorithmus funktioniert nach dem gleichen Prinzip wie der 8-Punkt-Algorithmus. Wir benutzen jedoch nur 7 Gleichungen und erhalten durch die Singulärwertzerlegung keinen 1-dimensionalen Lösungsraum, sondern einen 2-dimensionalen Lösungsraum, der durch zwei Vektoren $\mathbf{f}_1$ und $\mathbf{f}_2$ aufgespannt wird. Wir benutzen nun die Bedingung, dass die Fundamentalmatrix einen Rang von 2 haben und somit $\det F = 0$ gelten muss. Wir bilden die Matrizen F_1 aus $\mathbf{f}_1$ und F_2 aus $\mathbf{f}_2$ und suchen dann die Matrix $F = \alpha F_1 + (1-\alpha)F_2$, für die $\det F = 0$ gilt – wir müssen also die Gleichung

$$\det(\alpha F_1 + (1-\alpha)F_2) = 0$$

nach α lösen. Dies ist eine kubische Gleichung, wir erhalten also für α bis zu drei mögliche Werte und daher bis zu drei mögliche Lösungen für die Fundamentalmatrix.

Es ist auch hier mit der gleichen Argumentation notwendig, den Algorithmus auf normalisierten Daten durchzuführen, wie es in Abschnitt 6.2.2.1 beschrieben ist.

6.2.2.3 RANSAC

Beim 7- und 8-Punkt-Algorithmus werden nur 7 bzw. 8 Korrespondenzen zur Berechnung von F benutzt. Für diese Korrespondenzen ist die Gleichung $\mathbf{x}'^T F \mathbf{x} = 0$ erfüllt. Da im 8-Punkt-Algorithmus die Matrix im Nachhinein modifiziert wird, um Rang 2 zu erhalten, treten dort allerdings auch schon bei den benutzten Korrespondenzen Abweichungen auf. Wir erwarten, dass die Gleichung auch für die anderen, nicht benutzten Korrespondenzen gilt. Bei perfekten Daten wäre dies der Fall und bei sehr guten Daten sind die Fehler tatsächlich gering. Im Allgemeinen sind wir aber darauf angewiesen, dass gerade die ausgewählten Korrespondenzen besonders gut sind. Leider kann beim Tracking nicht garantiert werden, dass alle Korrespondenzen gut sind – wir müssen auch mit völlig falschen Korrespondenzen rechnen (siehe Abbildung 6.2, in der die blau gekennzeichneten Korrespondenzen offensichtlich nicht korrekt sind).
RANSAC ist eine im Grunde einfache aber sehr wirksame Methode, um Ausreißer, also fehlerhafte Korrespondenzen, zu entlarven und eine gute Auswahl an Punkten für den 7- oder 8-Punkt-Algorithmus zu treffen. Es ist eine allgemeine Methode, die auf verschiedene Probleme angewendet werden kann, zum Beispiel wird sie auch in Abschnitt 7.3.1 bei der Berechnung von 3D-Transformationen benutzt.

RANSAC, vorgestellt in [FB81], steht kurz für „RANdom SAmple Consensus“, also etwa „Konsens durch Zufallsauswahl“. Das einfache Prinzip besteht darin, zufällig eine Auswahl an Daten – in diesem Falle also Punktkorrespondenzen – zu treffen, aufgrund dieser Daten eine Lösung zu berechnen und die Qualität der Lösung auf allen Daten zu testen. Hier werden wir also die aus 7 oder 8 zufällig ausgewählten Punktkorrespondenzen berechnete Fundamentalmatrix auf allen Punktkorrespondenzen testen, d.h. die Abweichung berechnen, den diese Korrespondenzen in Gleichung 6.4 produzieren. Unter der Annahme, dass wir eine gute Auswahl getroffen haben – also alle ausgewählten Korrespondenzen sehr gute Korrespondenzen sind – erhalten wir für alle weiteren guten Korrespondenzen geringe Abweichungen. Schlechte Korrespondenzen dagegen erzeugen

große Abweichungen. Damit können wir sie als Ausreißer, also als fehlerhafte Daten, entlarven.
Wir gehen nun davon aus, dass die guten Punktkorrespondenzen deutlich in der Überzahl sind. Dann werden wir bei einer Auswahl der guten Korrespondenzen nur wenig Ausreißer erhalten, wenn dagegen schlechte Korrespondenzen in der Auswahl sind, werden wir viele (vermeintliche) Ausreißer erhalten. Wir werden nun so oft aus einer Zufallsauswahl eine Fundamentalmatrix berechnen und diese testen, bis wir mit der Anzahl an Ausreißern zufrieden sind. Diese „Zufriedenheit“ können wir klar definieren.
Wir sind dann zufrieden, wenn wir sehr sicher sind (mit einer fest definierten Wahrscheinlichkeit p, z. B. $p = 0.99$), in einer der Zufallsauswahlen keinen (tatsächlichen) Ausreißer benutzt zu haben. Die Anzahl der dann benötigten Iterationen lässt sich wie folgt berechnen:
s bezeichne die Anzahl an jeweils benutzten Korrespondenzen (also 7 beim 7-Punkt-Algorithmus) und ε den Anteil an Ausreißern. Dann berechnet sich die Anzahl an nötigen Iterationen N durch

$$N = \frac{\log(1-p)}{\log(1-(1-\varepsilon)^s)} \tag{6.6}$$

Da ε unbekannt ist, machen wir hier eine worst-case Schätzung: Wir setzen ε auf die geringste Anteil an Ausreißern, die eine Auswahl uns geliefert hat, es wird also nach jeder Iteration gegebenenfalls aktualisiert. Dann ist ε eine obere Schranke für den tatsächlichen Anteil an Ausreißern.

6.3 Umsetzung und Ergebnisse

In meinem System habe ich den 7-Punkt-Algorithmus in Verbindung mit RANSAC benutzt. Die Verwendung des 7-Punkt-Algorithmus hat gegenüber dem 8-Punkt-Algorithmus zum einen den Vorteil, dass die Fundamentalmatrix nicht nachträglich auf Rang 2 gebracht werden muss. Zum anderen ist aber vor allem in Verbindung mit RANSAC die Laufzeit wesentlich besser, da aus (6.6) folgt, dass wir bei einer geringeren Anzahl an benutzten Korrespondenzen weniger Iterationen benötigen.
Da der 7-Punkt-Algorithmus drei Lösungen liefern kann, werden in diesem Fall alle drei Lösungen auf allen Daten getestet und die beste (also die Matrix, die am wenigsten Ausreißer produziert) gegebenenfalls übernommen.

In Abbildung 6.2 sind zwei Bilder aus der Sequenz und ihre gemeinsamen Punkte zu sehen. Ein kleiner Kreis kennzeichnet die Position eines getrackten Punktes. Der Strich deutet zu der Position des korrespondierenden Punktes im jeweils anderen Bild, zeigt also die vermutete Bewegung des Raumpunktes relativ zur Kamera. Aufgrund dieser Korrespondenzen wurde nun auf oben beschriebene Weise eine Fundamentalmatrix erstellt. Die Korrespondenzen, die zur Berechnung der besten und daher ausgewählten Fundamentalmatrix benutzt wurden, sind grün eingezeichnet. Rot sind die Korrespondenzen, die mit dieser Fundamentalmatrix übereinstimmen. Die als Ausreißer entlarvten Punkte sind blau gekennzeichnet. Hier ist deutlich zu sehen, dass sie beim Tracking verrutscht sind.

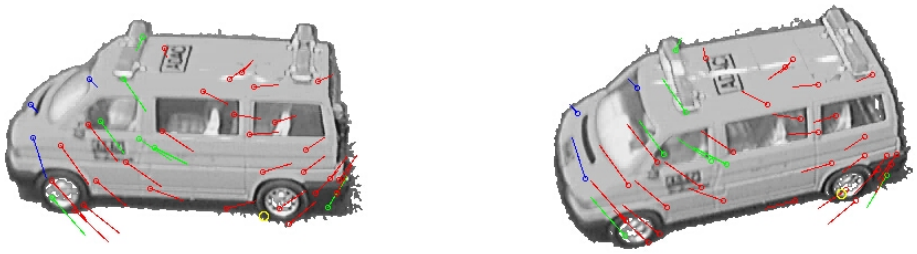


Abbildung 6.2: Punktkorrespondenzen zwischen zwei Bildern einer Sequenz

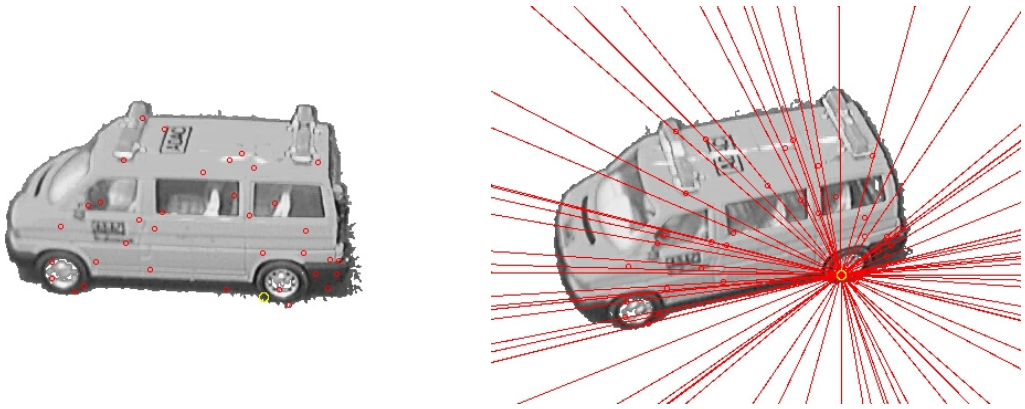


Abbildung 6.3: Als korrekt klassifizierte Punkte mit Epipolarlinien im zweiten Bild

Abbildung 6.3 zeigt alle als korrekt akzeptierten Punkte in beiden Bildern und im zweiten Bild zusätzlich die durch sie erzeugten Epipolarlinien. Es ist zu sehen, dass alle Punkte auf oder sehr nah an der durch ihre Korrespondenz erzeugten Epipolarlinie liegen. Abbildung 6.4 zeigt die Ausreißer und die durch sie erzeugten Epipolarlinien. Hier ist zu sehen, dass die Epipolarlinien jeweils den Punkt schneiden, der die korrekte Korrespondenz dargestellt hätte. Dies ist ein Hinweis darauf, dass wir eine gute Fundamentalmatrix gefunden haben.
Wenn wir uns die als korrekt klassifizierten Korrespondenzen ansehen, können wir feststellen, dass auch einige dieser Punkte beim Tracking verrutscht sind. Betrachten wir die Punkte an der Reflektion auf dem Dach des Autos. Reflektionen sind ein großes Problem beim Tracking, da sie scheinbar eine Textur des Objektes darstellen. Daher sind zwei Punkte auf einer Reflektion geblieben und hätten als Ausreißer identifiziert werden sollen. Doch in Abbildung 6.3 sehen wir, warum das nicht passiert ist: Die Verschiebung vom korrekten Punkt folgte entlang der Epipolarlinie und konnte daher nicht ermittelt werden. Solche fehlerhaften Korrespondenzen verschlechtern die gefundene Fundamentalmatrix und daraus konstruierte Kameramatrizen nicht, führen aber später in der Triangulierung (siehe Kapitel 7) zu fehlerhaften Ergebnissen. Der Grund wird bei Betrachtung von Abbildung 6.1(b) offensichtlich: Solange die Punkte auf (oder sehr nah bei) der jeweiligen durch die vermeintliche Korrespondenz erzeugten Epipolarlinie liegen, entsprechen sie der Epipolargeometrie unter der Annahme, dass der abgebildete

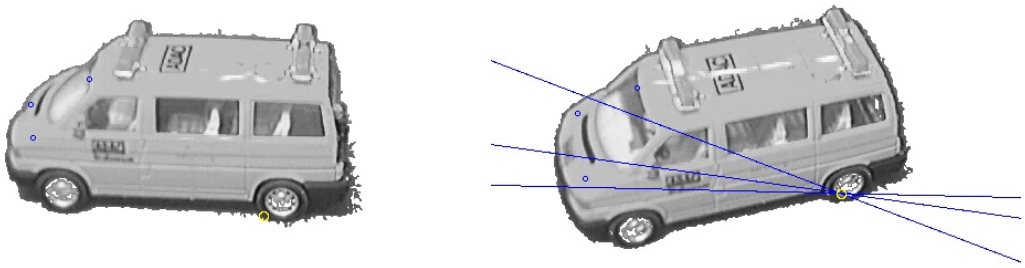


Abbildung 6.4: Als Ausreißer klassifizierte Punkte mit Epipolarlinien im zweiten Bild

Raumpunkt eine völlig andere Position im Raum hat als es tatsächlich der Fall ist. Wenn uns keine weiteren Informationen zur Verfügung stehen, kommen wir zwangsläufig zu einer falschen Vermutung über die Raumposition des dargestellten Punktes und haben an dieser Stelle keine Möglichkeit, dies zu vermeiden.

Allgemein kann man sagen, dass typische Fehlerquellen beim Tracking – Reflektionen, Schatten oder gleichförmige Oberflächen, die ein „Stehenbleiben“ des Punktes bewirken – tückische Auswirkungen auf die Berechnung der Fundamentalmatrix haben; denn sie bewirken keinen zufällig verteilten, sondern einen systematischen Fehler. Die falschen Korrespondenzen korrelieren miteinander – und zum Teil auch mit vielen korrekten Korrespondenzen. Daher ist es wenig aussagekräftig, dass in einer Simulation mit gleichverteiltem Fehler hohe Ausreißer-Quoten bewältigt werden können. Ein Fehler, der nicht mit anderen Daten korreliert, wird immer als Ausreißer klassifiziert und führt, sobald er zur Berechnung der Matrix verwendet wird, zu sehr vielen Ausreißern, so dass diese Matrix sofort verworfen würde. Die systematischen, typischen Fehler, die häufig auftreten, haben ein ganz anderes Verhalten und stellen ein Gegenmodell zur tatsächlichen Bewegung zwischen den Bildern dar. Daher benötigen wir deutlich mehr gute Korrespondenzen als Ausreißer, um eine korrekte Fundamentalmatrix zu erhalten.

6.4 Berechnen von Kameramatrizen

Die Fundamentalmatrix stellt die Beziehungen zwischen zwei Ansichten einer Szene dar und steht daher offensichtlich in direktem Bezug zu den Kameramatrizen der beiden beteiligten Bilder. In [HZ04] wird gezeigt, dass eine Fundamentalmatrix eindeutig aus den Kameramatrizen berechnet werden kann. Es können auch die Kameramatrizen aus der Fundamentalmatrix berechnet werden und das ist das Ziel dieses Abschnitts.
Die Berechnung der Kameramatrizen aus der Fundamentalmatrix ist jedoch leider alles andere als eindeutig, was schon daran deutlich wird, dass die Kameramatrizen deutlich mehr Freiheitsgrade als die Fundamentalmatrix haben. Jede Kameramatrix hat 11 Freiheitsgrade, zusammen also 22. Die Fundamentalmatrix hat dagegen nur 7 Freiheitsgrade. Die Differenz von 15 Freiheitsgraden entspricht den Freiheitsgraden einer 3-dimensionalen projektiven Transformation (siehe Abschnitt 2.2.2). Tatsächlich lässt sich zeigen (siehe dazu [HZ04]), dass eine Fundamentalmatrix, die sich auf ein Paar von Kameramatrizen

(P, P') bezieht, genau dann auch für ein anderes Paar von Kameramatrizen $(\tilde{P}, \tilde{P}')$ gilt, wenn eine invertierbare 4×4 Matrix H (also eine projektive Transformation) existiert, so dass $\tilde{P} = PH$ und $\tilde{P}' = P'H$.
Es gibt also eine große Anzahl von möglichen Paaren von Kameramatrizen. Üblicherweise wird eine spezifische *kanonische Form* dieses Paares gewählt. Gegeben sei eine Fundamentalmatrix F. Die erste Kameramatrix wird unabhängig von F gewählt durch

$$P = [\ I \mid \mathbf{0}\ ] \tag{6.7}$$

Hier ist I die 3×3 Identitätsmatrix und $\mathbf{0}$ ein 3-Vektor aus Nullen. Zur Zulässigkeit dieser Wahl siehe [HZ04]. Wir haben durch die Festlegung der ersten Matrix 11 Freiheitsgrade festgelegt und es bleiben 4 Freiheitsgrade, um die zweite Kameramatrix P' zu wählen . In [HZ04] wird gezeigt, dass P' gewählt werden kann durch

$$P' = [\ [\mathbf{e}']_\times F + \mathbf{e}'\mathbf{v}^T \mid \lambda \mathbf{e}'\ ] \tag{6.8}$$

mit $\lambda \in \mathbf{R}_{\neq 0}$ (ein Freiheitsgrad), $\mathbf{v}$ ist ein beliebiger 3-Vektor (3 Freiheitsgrade). Die Matrix $[\mathbf{e}']_\times$ ist dabei die dem Vektor $\mathbf{e}'$ entsprechende schiefsymmetrische Matrix und ist definiert durch

$$[\mathbf{e}']_\times = \begin{pmatrix} 0 & -e'_3 & e'_2 \\ e'_3 & 0 & -e'_1 \\ -e'_2 & e'_1 & 0 \end{pmatrix}$$

Eine übliche Wahl für die Parameter ist $\lambda = 1$ und $\mathbf{v} = \mathbf{0}$. Dies führt allerdings dazu, dass das Projektionszentrum von P' auf der Ebene im Unendlichen Π_∞ liegt. Dies stellt zwar für den weiteren Ablauf des Systems kein Problem dar, dennoch wollte ich dies vermeiden. So hat eine Kamera mit Projektionszentrum in Π_∞ den Nachteil, dass sie sich nicht durch (2.8) zerlegen lässt. Siehe hierzu Abschnitt 2.3.
Ich habe also $\lambda = 1$ und $\mathbf{v} = \mathbf{1} = (1, 1, 1)^T$ gewählt, somit wird die zweite Kamera berechnet durch

$$P' = [\ [\mathbf{e}']_\times F + \mathbf{e}'\mathbf{1}^T \mid \mathbf{e}'\ ] \tag{6.9}$$

Kapitel 7

Rekonstruktion von 3D-Punkten

In diesem Kapitel wollen wir 3D-Punkte rekonstruieren mit Hilfe der im letzten Kapitel berechneten Kameramatrizen. Da diese Kameramatrizen nicht eindeutig sind, erhalten wir keine eindeutige Rekonstruktion, sondern lediglich eine projektive Rekonstruktion. Im Abschnitt 7.1 werden wir diese Mehrdeutigkeit etwas näher beleuchten. Anschließend werden wir in Abschnitt 7.2 durch Triangulierung eine projektive Rekonstruktion einer Punktmenge erstellen. In Abschnitt 7.3 wird gezeigt, wie wir verschiedene solcher Rekonstruktionen zu einer einheitlichen Rekonstruktion zusammenfügen können.

7.1 Die Mehrdeutigkeit bei der Rekonstruktion

Um zu verstehen, warum wir nur eine projektive Rekonstruktion erstellen können, erinnern wir uns, dass eine Abbildung vom Raum ins Bild durch Gleichung (2.7), also $\mathbf{x} = P\mathbf{X}$ dargestellt wird. Gegeben sei nun eine Punktmenge von n Raumpunkten $\mathbf{X}_i$ und $\mathbf{x}_i$ seien deren Abbildungen, es gilt also

$$\mathbf{x}_i = P\mathbf{X}_i \ \forall i = 1..n$$

Sei H eine projektive Transformation, die wir auf die gesamte Punktmenge – also auf jeden einzelnen Raumpunkt – anwenden. Dann gilt

$$\mathbf{x}_i = (PH^{-1})H\mathbf{X}_i \ \forall i = 1..n$$

Wir erhalten also unter Verwendung einer entsprechend transformierten Kameramatrix PH^{-1} für jeden Punkt exakt die gleiche Projektion wie zuvor. Wenn die Kameramatrix keinen weiteren Beschränkungen unterliegt und wir P und die Punkte $\mathbf{X}_i$ also lediglich aufgrund ihrer Abbildungen $\mathbf{x}_i$ berechnet haben, ist keine Rekonstruktion „richtiger“ als eine beliebig transformierte Rekonstruktion.
Die Verwendung mehrerer Bilder hilft uns zunächst nicht weiter. Benutzen wir m Bilder, also auch m Kameramatrizen, so gilt für die Projektion $\mathbf{x}_{ij}$ des i-ten Punktes ins j-te Bild

$$\mathbf{x}_{ij} = (P_jH^{-1})H\mathbf{X}_i \ \forall i = 1..n, j = 1..m \tag{7.1}$$

für eine beliebige projektive Transformation H.
Werfen wir einen Blick in Abschnitt 6.4, so sehen wir, dass wir hier für ein berechnetes

Kamerapaar P, P' auch jedes Kamerapaar $PH, P'H$ wählen können. Offensichtlich beziehen diese sich dann auf eine projektive Rekonstruktion $H^{-1}\mathbf{X}$ der Raumpunkte, wenn $\mathbf{X}$ eine Rekonstruktion in Bezug auf P, P' ist. Wenn wir diese Rekonstruktion also in Abschnitt 7.2 durchführen werden, muss uns bewusst sein, dass die Rekonstruktion von den gewählten Kameramatrizen abhängt und jede projektiv transformierte Rekonstruktion ebenso richtig wäre. Wir müssen zur Kenntnis nehmen, dass wir mit den gegebenen Daten auch nicht mehr erreichen können – die Informationen, die wir benutzen, reichen lediglich für eine projektive Rekonstruktion aus.

Wir können uns nun fragen, welche weiteren Informationen wir benötigen, um eine eindeutige Rekonstruktion zu erhalten – falls sie überhaupt möglich ist. Würden wir die „wahren“ Kameramatrizen kennen und zur Rekonstruktion benutzen, würden wir eine „wahre“, eindeutige Rekonstruktion erhalten. Jedoch sind in den Kameramatrizen Position und Ausrichtung der Projektionszentren in der Welt enthalten – relativ zu einem Weltkoordinatensystem. Doch weder gibt es ein allgemeines Weltkoordinatensystem, auf das wir uns beziehen könnten, noch könnten wir aus optischen Informationen die Position eines Objektes in einem derartigen Koordinatensystem bestimmen. Die absolute Position des Objektes in der Welt interessiert uns aber eigentlich auch nicht – eine Translation und auch eine Rotation des Objektes in der Welt können und wollen wir also nicht ermitteln. Außerdem können wir die Skalierung eines Objektes nicht bestimmen. Sehen wir uns das Spielzeugauto an, dessen Bildsequenz wir als Beispiel in dieser Arbeit benutzen: Es ist für den Betrachter der Bilder unmöglich, auch nur annähernd die Größe des Spielzeugautos zu ermitteln. Dies ist eine Information, die, ebenso wie die absolute Position und Rotation des Objektes in der Welt, in den Bildern einfach nicht enthalten ist.

Sehen wir uns die Kameramatrizen (siehe Abschnitt 2.3) an, so stellen wir fest, dass es also Position und Ausrichtung ($\mathbf{C}$ und R) sind, die wir nicht absolut bestimmen können. Bestimmen könnten wir die Unterschiede dieser Werte zwischen den Kameramatrizen, jedoch nicht die Skalierung dieser Unterschiede. Die Kalibrationsmatrizen dagegen enthalten physikalische Eigenschaften der Kamera. Diese könnten prinzipiell ermittelt werden.
Was wir dann erreichen können, ist eine *metrische Rekonstruktion*. Sei die Punktmenge $\{\mathbf{X}_i\}$ eine metrische Rekonstruktion, dann ist $\{H_S\mathbf{X}_i\}$ eine ebenso gültige metrische Rekonstruktion, wobei H_S eine Ähnlichkeits-Transformation nach (2.5) darstellt. In einer metrischen Rekonstruktion ist also die Form eines Objektes eindeutig, die Position und Ausrichtung sowie die Skalierung des Objektes nicht.
Um an die Kalibrationsmatrizen zu gelangen, wäre es am naheliegensten, die Kamera zu kalibrieren. Dies ist z. B. durch Aufnahme von geeigenten *Kalibrationsobjekten*, deren Geometrie bekannt ist, möglich, siehe hierzu [HZ04]. Dieses Ansatz habe ich nicht gewählt, da es einige Einschränkungen mit sich bringt und ich das System möglichst allgemein und offen halten wollte. Die Bedingung, nur Bildsequenzen zu verwenden, die mit kalibrierten Kameras und bekannten Kalibrationsmatrizen aufgenommen wurden, ist eine recht harte Einschränkung. Wie eine metrische Rekonstruktion mit Hilfe bekannter Kalibrationsmatrizen erstellt werden kann, ist in [HZ04] beschrieben.
Ein vielversprechender Ansatz ist die Idee der automatischen Kalibrierung. Wir können Bedingungen an die Kalibrationsmatrix stellen, zum Beispiel dass alle oder bestimmte Parameter konstant sind. Dann werden mit jedem Bild, das wir benutzen, die Freiheitsgrade der Rekonstruktion verringert und bei einer ausreichenden Zahl von benutzten Bildern

können wir eine metrische Rekonstruktion erstellen. Siehe hierzu auch Abschnitt 10.2.2.1. Für den Augenblick finden wir uns aber damit ab, dass wir lediglich eine projektive Rekonstruktion erstellen.

7.2 Triangulierung

Die hier beschriebene Triangulierung und insbesondere der Algorithmus in Abschnitt 7.2.1 sind aus [HZ04] übernommen.
Gegeben sei eine Punktkorrespondenz $\mathbf{x} \leftrightarrow \mathbf{x}'$ und die zu den jeweiligen Bildern gehörenden Kameramatrizen P und P'. Dann sind die von $\mathbf{x}$ und $\mathbf{x}'$ zurückprojizierten Geraden bekannt. Da beide Punkte Projektionen desselben Raumpunktes $\mathbf{X}$ darstellen, sollte $\mathbf{X}$ auf beiden Geraden liegen und somit den Schnittpunkt der Geraden darstellen (siehe auch Abb. 6.1). Auf diese Weise kann der Punkt $\mathbf{X}$ eindeutig bestimmt werden. Er sollte die Gleichungen

$$\mathbf{x} = P\mathbf{X} \tag{7.2}$$

und

$$\mathbf{x}' = P'\mathbf{X} \tag{7.3}$$

exakt erfüllen. Entsprechen P und P' den tatsächlichen Kameramatrizen, so ist $\mathbf{X}$ die tatsächliche Rekonstruktion des Raumpunktes. Entsprechen lediglich die Kalibrationsmatrizen von P und P' den tatsächlichen physikalischen Gegebenheiten der Kameras, so ist $\mathbf{X}$ eine metrische Rekonstruktion. Werden die Kameras jedoch wie in Abschnitt 6.4 gewählt, so ist $\mathbf{X}$ lediglich eine projektive Rekonstruktion, die also um eine projektive Transformation vom tatsächlichen Raumpunkt veschieden ist.

Da unsere Daten nicht exakt sind, schneiden sich die zurückprojizierten Geraden leider in der Regel nicht tatsächlich, sondern sind windschief zueinander und laufen knapp aneinander vorbei, wie in Abbildung 7.1 dargestellt.

In den Bildebenen stellt sich das Problem so dar wie in Abbildung 7.2 gezeigt. Die Punkte liegen nicht exakt auf der durch ihre Korrespondenz erzeugten Epopolarlinie. Siehe hierzu auch Abbildungen 6.3 und 6.4 in Abschnitt 6.3. Während wir dort die Ausreißer als fehlerhafte Korrespondenzen eingestuft haben, an denen $\mathbf{x}$ und $\mathbf{x}'$ also tatsächlich unterschiedliche Raumpunkte darstellen, können wir auch bei den als richtig eingestuften Korrespondenzen nicht davon ausgehen, dass sie exakt der Epipolargeometrie entsprechen, sich die zurückprojizierten Geraden also schneiden.
Eine Triangulierung auf realen Daten kann also nicht exakt sein, sondern nur eine Abschätzung des wahrscheinlichsten Raumpunktes.

7.2.1 Ein lineares Triangulierungsverfahren

Da wir keinen Punkt finden werden, der die Gleichungen (7.2) und (7.3) exakt erfüllt, müssen uns mit einem Punkt zufrieden geben, der beide Gleichungen möglichst gut erfüllt. Wir wollen also $\mathbf{X}$ so wählen, dass $d(\mathbf{x}, P\mathbf{X})$ und $d(\mathbf{x}', P'\mathbf{X})$ minimiert werden.

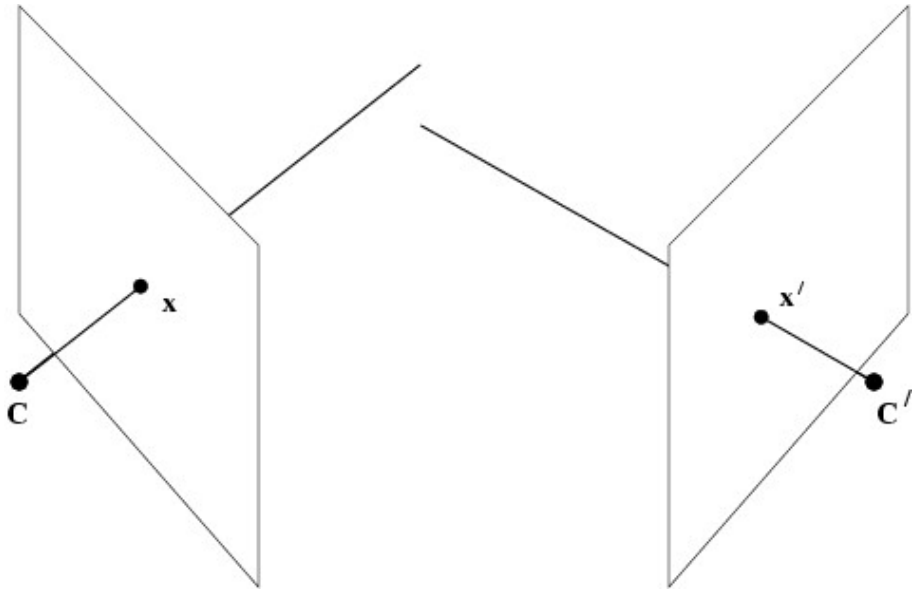


Abbildung 7.1: Die von den Bildpunkten zurückprojizierten Geraden schneiden sich leider nur in der Theorie

Abbildung 7.2: Die Abbildungen der Situation aus Abb. 7.1 in den Bildebenen

Dies sind die Distanzen der vom rekonstruierten Punkt $\mathbf{X}$ in die Bilder projizierten, also berechneten, Bildpunkte zu den gemessenen Bildpunkten (also Abbildungen des tatsächlichen Raumpunktes). Diese Vorgehensweise hat den Vorteil, projektiv unabhängig von den gewählten Kameramatrizen zu sein. Das heißt: Wird auf einen mit Kameras P und P' rekonstruierten Punkt $\mathbf{X}$ eine projektive Transformation H angewandt, so enspricht dieser Punkt $H\mathbf{X}$ exakt dem Punkt, den wir durch Verwendung der Kameras PH^{-1} und $P'H^{-1}$ erhalten würden (dies folgt aus (7.1)).

Wir erinnern uns, dass $\mathbf{x}$ und $P\mathbf{X}$ (sowie $\mathbf{x}'$ und $P'\mathbf{X}$) homogene Koordinaten sind. Die Gleichungen (7.2) und (7.3) gelten also unabhängig von den Skalierungen beider Seiten und können daher durch Kreuzprodukte ausdrückt werden, es gilt also

$$\mathbf{x} \times P\mathbf{X} = \mathbf{0} \tag{7.4}$$

und

$$\mathbf{x}' \times P\mathbf{X}' = \mathbf{0} \tag{7.5}$$

Daraus erhalten wir jeweils drei Gleichungen, für (7.4) mit $\mathbf{x} = (x, y, 1)$:

$$x(\mathbf{p}^{3T}\mathbf{X}) - (\mathbf{p}^{1T}\mathbf{X}) = 0$$

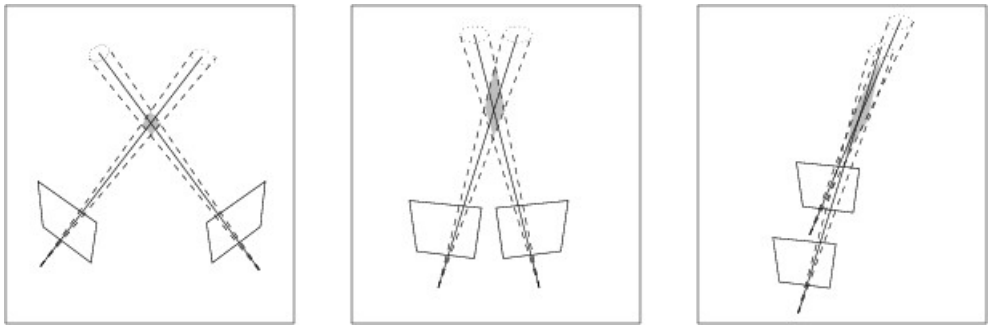


Abbildung 7.3: Die Qualität der Rekonstruktion sinkt, je kleiner der Winkel zwischen den zurückprojizierten Geraden ist

$$y(\mathbf{p}^{3T}\mathbf{X}) - (\mathbf{p}^{2T}\mathbf{X}) = 0$$
$$x(\mathbf{p}^{2T}\mathbf{X}) - y(\mathbf{p}^{1T}\mathbf{X}) = 0$$

Hier ist p^{iT} die i-te Reihe von P. Für (7.5) mit $\mathbf{x}' = (x', y', 1)$ erhalten wir analog

$$x'(\mathbf{p}'^{3T}\mathbf{X}) - (\mathbf{p}'^{1T}\mathbf{X}) = 0$$
$$y'(\mathbf{p}'^{3T}\mathbf{X}) - (\mathbf{p}'^{2T}\mathbf{X}) = 0$$
$$x'(\mathbf{p}'^{2T}\mathbf{X}) - y'(\mathbf{p}'^{1T}\mathbf{X}) = 0$$

Wählen wir jeweils zwei der Gleichungen, so sind diese voneinander linear unabhängig und die dritte Gleichung von beiden linear abhängig. Wir fügen nun die jeweils ersten beiden der Gleichungen zusammen zu einem Gleichungssystem $A\mathbf{X} = \mathbf{0}$ mit

$$A = \begin{pmatrix} x\mathbf{p}^{3T} - \mathbf{p}^{1T} \\ y\mathbf{p}^{3T} - \mathbf{p}^{2T} \\ x'\mathbf{p}'^{3T} - \mathbf{p}'^{1T} \\ y'\mathbf{p}'^{3T} - \mathbf{p}'^{2T} \end{pmatrix}$$

Dies ist ein Gleichungssystem mit 4 Gleichungen und 4 homogenen Unbekannten und ist daher überbestimmt. Wir lösen es mit der Singulärwertzerlegung (siehe A.1.2) und erhalten einen Lösungsvektor für $\mathbf{X}$, der $\|A\mathbf{X}\|$ und somit $d(\mathbf{x}, P\mathbf{X})$ und $d(\mathbf{x}', P'\mathbf{X})$ minimiert unter der Bedingung $\|\mathbf{X}\| = 1$. Auf diese Weise kann nun zu jeder Punktkorrespondenz $\mathbf{x} \leftrightarrow \mathbf{x}'$ ein Raumpunkt $\mathbf{X}$ berechnet werden.

7.2.2 Wahl der Bilder für die Triangulierung

Für das Tracking war es notwendig, dass zwischen zwei nachfolgenden Bildern unserer Bildsequenz keine große Bewegung stattgefunden hat, die Kameraposition sich also nicht sehr verändert hat. Für die Rekonstruktion dagegen gilt das Gegenteil. je größer der Winkel zwischen den Projektionsgeraden von den Bildebenen zur vermuteten 3D-Position des Raumpunktes ist, desto geringer wird der Fehler bei der Rekonstruktion ausfallen. Der Grund wird aus Abbildung 7.3 ersichtlich. Ein kleiner Winkel zwischen den Projektionsgeraden bewirkt einen größeren (schraffiert dargestellten) Fehlerbereich, in welchem der Raumpunkt mit einer gewissen Wahrscheinlichkeit liegt. Im letzten Abschnitt haben wir

gesehen, dass wir den Raumpunkt nicht exakt bestimmen können, sondern nur versuchen, den Fehler, den der berechnete Raumpunkt in den Bildern bewirkt, zu minimieren. Hier wird deutlich, dass auch ein kleiner Fehler in den Bildern bei ungünstigem Winkel einen großen Fehler im Raum bedeuten kann. Daher muss darauf geachtet werden, dass zur Rekonstruktion immer Bilder benutzt werden, die zwar noch viele Korrespondenzen haben, aber möglichst weit auseinander liegen.

7.3 Das Erstellen einer projektiven Rekonstruktion aus der gesamten Sequenz

Mit Hilfe der Epipolargeometrie und der Triangulierung können wir nun für Bildpaare aus der Sequenz eine projektive Rekonstruktion erstellen. Diese Rekonstruktion umfasst jeweils die Punkte, die in beiden Bildern sichtbar sind. Unser Ziel ist aber, eine einheitliche Rekonstruktion von Punkten aus allen Bildern der Sequenz zu gewinnen.
Wir können leider nicht einfach alle rekonstruierten Punkte zu einer Gesamtmenge von Raumpunkten zusammenfügen, da alle Rekonstruktionen projektiv sind und sich somit auch untereinander um eine projektive Transformation unterscheiden. So können für einen Raumpunkt in verschiedenen Rekonstruktionen völlig unterschiedliche Werte berechnet werden. Wir müssen die Rekonstruktionen also zunächst durch entsprechende Transformationen vereinheitlichen.
Im nachfolgenden Abschnitt werde ich beschreiben, wie man eine projektive Transformation zwischen 3D-Punktmengen berechnen kann. Als Voraussetzung benötigen wir eine Anzahl von Punktkorrespondenzen $\mathbf{X}_i \leftrightarrow \mathbf{X'}_i$. Wir berechnen dann die Transformation, die jedes $\mathbf{X'}_i$ auf $\mathbf{X}_i$ abbildet. Eine derartige Transformation ist ab 5 Punkten eindeutig, da durch jeden 3D-Punkt 3 der 15 Freiheitsgrade einer projektiven Transformation festgelegt werden.
Wir müssen also darauf achten, in beiden Punktemengen eine ausreichend große Anzahl (also möglichst größer als 5, da wir auch hier mit RANSAC arbeiten werden) an gemeinsamen Punkten zu haben. Dies sind offensichtlich zunächst genau die Punkte, die in allen beteiligten Bildern sichtbar sind.

7.3.1 Berechnen einer projektiven Transformation zwischen 3D-Punktmengen

Gegeben sei eine Menge von 3D-Punktkorrespondenzen $\mathbf{X} \leftrightarrow \mathbf{X'}$. Gesucht ist eine projektive Transformation H so dass für alle Korrespondenzen

$$\mathbf{X} = H\mathbf{X'} \tag{7.6}$$

gilt. Auf den ersten Blick ähnelt dieses Problem dem Problem der Triangulierung. Dort haben wir aus den Gleichungen (7.4) und (7.5) ein lineares Gleichungssystem erstellt, das wir lösen konnten. Das Problem ist, dass die Koordinaten homogen sind, bei beliebigen Skalierungen beider Seiten die Gleichheit also immer noch gilt. Bei der Triangulierung konnten wir dieses Problem durch die Verwendung des Kreuzprodukts lösen. Da wir hier vierdimensionale Koordinaten haben, besteht diese Möglichkeit nun leider nicht. Da es nicht möglich ist, die beliebige Skalierung aus den Gleichungen zu entfernen, ohne dass das

entstehende Gleichungssystem nicht mehr linear wäre, machen wir genau das Gegenteil: Wir fügen eine zusätzliche Unbekannte hinzu und berücksichtigen die beliebige Skalierung der Gleichung explizit durch einen Skalierungsfaktor k. Somit erhalten wir aus (7.6):

$$k\mathbf{X} - H\mathbf{X}' = \mathbf{0} \tag{7.7}$$

Die Gleichung (7.7) entspricht den vier Gleichungen

$$kx_1 - h_{11}x'_1 - h_{12}x'_2 - h_{13}x'_3 - h_{14}x'_4 = 0$$

$$kx_2 - h_{21}x'_2 - h_{22}x'_2 - h_{23}x'_3 - h_{24}x'_4 = 0$$

$$kx_3 - h_{31}x'_3 - h_{32}x'_3 - h_{33}x'_3 - h_{34}x'_4 = 0$$

$$kx_4 - h_{41}x'_4 - h_{42}x'_4 - h_{43}x'_3 - h_{44}x'_4 = 0$$

Dieses Gleichungssystem können wir schreiben als

$$A\mathbf{h}^* = \mathbf{0}$$

mit

$$A = \begin{pmatrix} x_1 & -\mathbf{x}'^T & \mathbf{0}^T & \mathbf{0}^T & \mathbf{0}^T \\ x_2 & \mathbf{0}^T & -\mathbf{x}'^T & \mathbf{0}^T & \mathbf{0}^T \\ x_3 & \mathbf{0}^T & \mathbf{0}^T & -\mathbf{x}'^T & \mathbf{0}^T \\ x_4 & \mathbf{0}^T & \mathbf{0}^T & \mathbf{0}^T & -\mathbf{x}'^T \end{pmatrix}$$

Der Vektor $\mathbf{0}$ jeweils ein Vektor aus Nullen. Der Vektor $\mathbf{h}^*$ ist der aus den Einträgen von H erzeugte Vektor, erweitert um den Skalierungsfaktor k, also

$$\mathbf{h}^* = (k, h_{11}, h_{12}, ..., h_{43}, h_{44})$$

Wir bilden nun ein Gleichungssystem aus nicht nur einer, sondern aus 5 Korrespondenzen, da wie oben festgestellt 5 Korrespondenzen zur Bestimmung einer 3D-Transformation notwendig sind. Aus jeder Korrespondenz erhalten wir 4 Gleichungen und einen zusätzlichen Skalierungsfaktor. Der Vektor $\mathbf{h}^*$ besteht dann aus den 16 Einträgen von H und 5 Skalierungsfaktoren, hat also 21 Einträge. Die Matrix A ist aus 20 Gleichungen zusammengesetzt und ist eine 20×21 - Matrix.
Dieses Gleichungssystem lösen wir mit Hilfe der Singulärwertzerlegung (siehe Abschnitt A.1.2) und erhalten $\mathbf{h}^*$ als Lösungsvektor. Wir können dann aus den entsprechenden Einträgen von $\mathbf{h}^*$ die Transformationsmatrix H erstellen. Die gleichzeitig berechneten Skalierungsfaktoren werden nicht benötigt und müssen nicht beachtet werden.

Wie bei der Berechnung der Fundamentalmatrix ist aus den gleichen Gründen auch hier eine Normalisierung der Daten vor Anwendung des Algorithmus zu empfehlen. Die Normalisierung der Punktmengen kann völlig analog erfolgen, siehe hierzu Abschnitt 6.2.2.1. Für die Berechnung der Transformation für die Korrespondenzen $\mathbf{X}_i \leftrightarrow \mathbf{X}'_i$ werden also zunächst die Normalisierungs-Transformationen T und T' für die Punktmengen berechnet. Dann wird die Transformation $\hat{H}$ für die Korrespondenzen $T\mathbf{X}_i \leftrightarrow T'\mathbf{X}'_i$ berechnet, die also

$$T\mathbf{X}_i = \hat{H}T'\mathbf{X}'_i$$

erfüllt. Die Abbildung auf die ursprünglichen Punkte $\mathbf{X}_i$ ist dann gegeben durch

$$\mathbf{X}_i = T^{-1}\hat{H}T'\mathbf{X}'_i$$

und die gesuchte Transformation H durch

$$H = T^{-1}\hat{H}T'$$

Zur Berechnung der Transformation kann und sollte völlig analog zu Abschnitt 6.2.2.3 RANSAC benutzt werden. Lediglich die Bewertung der Ausreißer unterscheidet sich. Bei der Erstellung der Fundamentalmatrix wurde ein Ausreißer als eine falsche Korrespondenz bewertet, so konnten fehlerhafte Daten entlarvt und entfernt werden. Bei der Erstellung der Transformation bedeutet ein Ausreißer, dass die Raumpunkte der beiden Rekonstruktionen sich scheinbar nicht entsprechen, da es durch die Transformation nicht gelingt, sie aufeinander abzubilden. Dies muss nicht an fehlerhaften Ausgangsdaten liegen, sondern könnte auch auf ein schlechtes Ergebnis der Triangulierung zurückzuführen sein. Andererseits könnte der Fehler durch eine Scheinkorrespondenz, die durch eine Verschiebung des Bildpunktes entlang der Epipolarlinie entstand (wie in Abschnitt 6.3 beschrieben), verursacht werden. In jedem Fall würden diese Punkte im weiteren Verlauf des Systems Probleme verursachen, wenn sie korrespondierende Raumpunkte in den nächsten Rekonstruktionen haben. Daher ist es unvermeidlich, diese Punkte als fehlerhafte Daten zu werten und aus der Rekonstruktion zu entfernen.

7.3.2 Überblick über das Erstellen der Gesamt-Rekonstruktion

Wir können nun die gesamte Vorgehensweise zum Erstellen einer projektiven Rekonstruktion aller 3D-Punkte zusammenfassen:

1. Generiere aus der Bildsequenz eine „spärliche Sequenz“, d. h. eine Sequenz, die nur einige Bilder aus der Original-Sequenz enthält, aber den gesamten Scan-Vorgang abdeckt. Diese Sequenz sollte so viele Bilder wie nötig enthalten, so dass es in *jeweils drei* aufeinander folgenden Bildern ausreichend Punktkorrespondenzen gibt, und so wenig wie möglich, um gute Ergebnisse bei der Triangulierung zu erhalten und die Anzahl der nötigen Transformationen gering zu halten.

2. Gruppiere die Bilder der Sequenz zu Paaren aus aufeinanderfolgenden Bildern, so dass aufeinanderfolgende Paare jeweils ein Bild gemeinsam haben.

3. Berechne für das erste Bildpaar wie in Kapitel 6 beschrieben die Epipolargeometrie, also Fundamentalmatrix und die Kameramatrizen, und entferne Punktkorrespondenzen, die dabei als Ausreißer identifiziert werden können.

4. Berechne, wie in diesem Kapitel beschrieben, für alle in beiden Bildern sichtbaren Punkte durch Triangulierung eine Raumposition. Wir erhalten eine projektive Rekonstruktion dieser Punkte, an die wir die folgenden Rekonstruktionen anpassen wollen. Die Rekonstruktion enthält neben der Punktmenge auch die benutzten Kameramatrizen. Wir nennen sie hier $\Gamma_0 = (\{P_j\}, \{\mathbf{X}_i\})$.

5. Berechne analog für alle weiteren Bildpaare die Fundamentalmatrix, Kameramatrizen und eine Rekonstruktion Γ_l und berechne wie in Abschnitt 7.3.1 beschrieben

jeweils eine Transformation H, welche alle Punkte dieser Rekonstruktion an Γ_0 angleicht. Transformiere dann jeden Punkt aus Γ_l, der keine Korrespondenz in Γ_0 hat, zu $\mathbf{X}_H = H\mathbf{X}$ und füge ihn Γ_0 hinzu. Transformiere die zweite Kameramatrix P' zu $P'_H = P'H^{-1}$ und füge sie Γ_0 hinzu.

Da wir darauf geachtet haben, dass es zwischen jeweils drei aufeinander folgenden Bildern ausreichend Korrespondenzen gibt, ist sichergestellt, dass es in jeder Rekonstruktion ausreichend Punkte mit Korrespondenz in Γ_0 gibt. Denn wenn in drei aufeinander folgenden Bildern Bildpunkte existieren, die denselben Raumpunkt darstellen, sind Rekonstruktionen dieser Punkte in zwei aufeinander folgenden Rekonstruktionen Γ_{l-1} und Γ_l vorhanden. Da bei Berechnung von Γ_l alle Punkte aus Γ_{l-1} bereits transformiert und Γ_0 hinzugefügt worden sind, sind nun auch in Γ_0 Rekonstruktionen dieser Punkte vorhanden.
Auf diese Weise ist es möglich, eine einheitliche Rekonstruktion von Punkten zu erstellen, die jeweils nur in einem Teil der Sequenz zu sehen sind. Allerdings ist diese Vorgehensweise nicht ganz unproblematisch. Wenn wir Punkte aus einer Rekonstruktion Γ_l auf ihre Korrespondenzen in Γ_0 abbilden, können diese selbst bereits transformiert worden sein. Da jede Transformation einen Fehler mit sich bringen kann, multiplizieren sich diese Fehler nun. Punkte einer Rekonstruktion Γ_{l+1} werden wiederum mit Hilfe der transfomierte Punkte aus Γ_l transfomiert. Das bedeutet, dass eine misslungene Transformation oder auch eine misslungene Rekonstruktion alle folgenden Transformationen stark beeinträchtigt, bzw. ein akzeptables Ergebnis gar unmöglich macht.
Im nächsten Kapitel werden wir zunächst eine Methode kennenlernen, um das Gesamtergebnis unserer Rekonstruktion bewerten und nachträglich verbessern zu können. Anschließend werden wir, um die gerade genannte Problematik zu reduzieren, diese Methode nicht erst nachträglich, sondern bereits iterativ während des Erstellens der Rekonstruktion anwenden.

Kapitel 8

Optimierung der Rekonstruktion

Wie in Abschnitt 7.3.2 zusammengefasst haben wir nun unser Ziel erreicht – eine 3D-Rekonstruktion der Punkte, die wir in den vorigen Abschnitten beobachtet und getrackt haben. Wir haben festgestellt, dass uns auf dem Weg zu dieser Rekonstruktion einige Fehlerquellen begegnet sind und wir nicht davon ausgehen können, dass unser Ergebnis wirklich eine zufriedenstellende projektive Rekonstruktion ist. Wir müssen nun einen Weg finden, um die Güte unserer Rekonstruktion zu messen. Anschließend werden wir versuchen, diese Güte nachträglich zu verbessern.

8.1 Bewertung der Rekonstruktion durch den Rückprojektionsfehler

Die einzige Information, die wir verwendet haben, ist die Position von bestimmten Punkten in den Bildern und welche Bildpunkte miteinander korrespondieren, also den gleichen Raumpunkt darstellen. An unser Modell kann also auch nur das unsere Anforderung sein: Die rekonstruierten Punkte sollen durch die rekonstruierten Kameras genau an die Stellen abgebildet (also zurückprojiziert) werden, an denen wir die zugehörigen Bildpunkte gemessen haben. Die Abweichungen können wir messen und erhalten einen *Rückprojektionsfehler*.
Der Rückprojektionsfehler eines Punktes ist uns bereits in Abschnitt 7.2.1 begegnet. Er lässt sich ausdrücken durch

$$e_{\hat{\mathbf{X}}} = d(\hat{P}\hat{\mathbf{X}}, \mathbf{x})$$

Hier ist also $\mathbf{x}$ der gemessene Bildpunkt, $\hat{\mathbf{X}}$ der rekonstruierte Raumpunkt und $\hat{P}$ die rekonstruierte Kameramatrix.
Wir betrachten nun die gesamte Rekonstruktion $\Gamma = (\{\hat{P}_j\}, \{\hat{\mathbf{X}}_i\})$ mit den berechneten Raumpunkten $\hat{\mathbf{X}}_i$ und den berechneten Kameramatrizen $\hat{P}_j$. $\mathbf{x}_{ij}$ sei der im j-ten Bild gemessene Bildpunkt, der den i-ten Raumpunkt abbildet. Dann summiert sich der Rückprojektionsfehler zu

$$e_\Gamma = \sum_{i,j} d(\hat{P}_j\hat{\mathbf{X}}_i, \mathbf{x}_{ij})^2 \tag{8.1}$$

Hier drückt d die geometrische Bilddistanz aus, also den Abstand des zurückprojizierten zum gemessenen Punkt im Bild.
Da die Messungen nicht exakt sind, wird selbst bei einer optimalen Rekonstruktion ein Fehler übrig bleiben. In der Annahme, dass der Messfehler ein Gauß-verteiltes Rauschen ist, suchen wir eine *Maximum-Likelihood*-Schätzung, daher wird in Formel 8.1 jeder einzelne Fehler quadriert.
Wir können die Güte unseres Modells also durch eine handliche Formel ausdrücken. Offensichtlich wünschen wir einen möglichst kleinen Rückprojektionsfehler. Daraus entsteht das Minimierungsproblem

$$min_{\hat{P}_j,\hat{\mathbf{X}}_i} e_\Gamma = min_{\hat{P}_j,\hat{\mathbf{X}}_i} \sum_{i,j} d(\hat{P}_j\hat{\mathbf{X}}_i, \mathbf{x}_{ij})^2 \tag{8.2}$$

Dieses Minimierungsproblem wollen wir durch *Bundle Adjustment* lösen, im Deutschen etwa „Bündelanpassung“. Der Name begründet sich durch die Vorstellung, dass durch die Minimierung die Strahlenbündel zwischen den 3D-Punkten und den Kamerazentren gleichzeitig angepasst werden. Die Minimierung wird hier mit dem Levenberg-Marquardt-Algorithmus durchgeführt. Eine Alternative, die zumindest eine sinnvolle Ergänzung darstellen kann, ist die Verwendung eines evolutionären Algorithmus.

8.2 Bundle Adjustment

Wir werden kurz einen Blick auf die allgemeine Funktionsweise von interativen Miniermungsmethoden wie den Levenberg-Marquard-Algorithmus werfen und anschließend die spezielle Anwendung beim Bundle Adjustment betrachten. Mehr Details zur numerischen Minimierung im Allgemeinen und zum Bundle Adjustment im Besonderen sind in [TMHF00] zu finden. Die hier angewandte Vorgehensweise und insbesondere der Algorithmus aus Abschnitt 8.2.3 sind aus [HZ04] übernommen.

8.2.1 Iterative numerische Minimierungsmethoden

Gegeben sei eine Vektorfunktion $\mathbf{f}(\mathbf{p})$, die optimert werden soll. Es gibt einen Zielvektor $\mathbf{z}$ und der Parametervektor $\mathbf{p}$ soll so gewählt werden, dass $\mathbf{z}$ durch $\mathbf{f}(\mathbf{p})$ approximiert wird, es soll also gelten

$$\mathbf{f}(\mathbf{p}) \approx \mathbf{z} \tag{8.3}$$

Daraus ergibt sich die Fehlerfunktion

$$\boldsymbol{\epsilon}(\mathbf{p}) = \mathbf{f}(\mathbf{p}) - \mathbf{z} \tag{8.4}$$

Die Funktion $e(\mathbf{p})$ sei die quadrierte Fehlernorm von $\boldsymbol{\epsilon}(\mathbf{p})$ und ist definiert durch

$$e(\mathbf{p}) = \frac{1}{2}\left\|\boldsymbol{\epsilon}(\mathbf{p})\right\|^2 = \frac{1}{2}\boldsymbol{\epsilon}(\mathbf{p})^T\boldsymbol{\epsilon}(\mathbf{p}) \tag{8.5}$$

Wir wollen nun $e(\mathbf{p})$ minimieren.
Ein Startwert $\mathbf{p}_0$ ist vorgegeben. e wird an dieser Stelle lokal approximiert durch eine lineare oder quadratische Funktion und mit Hilfe deser Abschätzung versuchen wir, einen

Wert $\mathbf{p}$ zu finden, der den Funktionswert gegenüber $\mathbf{p}_0$ verringert. Die Veränderung des Parametervektors, die wir berechnen wollen, ist $\Delta = \mathbf{p} - \mathbf{p}_0$. Wir fahren iterativ fort, bis wir zu einem lokalen Minimum gelangen. Wir können allerdings nicht entscheiden, ob das erreichte Minimum auch ein globales Minimum ist.

8.2.1.1 Gradientenabstieg

Die Ableitung von $\boldsymbol{\epsilon}$ an der Stelle $\mathbf{p}_0$ entspricht der Ableitung von $\mathbf{f}$ an der Stelle $\mathbf{p}_0$ und ist gegeben durch die Jacobi-Matrix J, die sich, stehen keine anderen Möglichkeiten zur Verfügung, numerisch berechnen lässt. Der Gradient von e (also (8.5)) an der Stelle $\mathbf{p}_0$ ist dann gegeben durch

$$\mathbf{g} = J^T \boldsymbol{\epsilon}(\mathbf{p}_0) \tag{8.6}$$

Die negative Gradientenrichtung $-\mathbf{g}$ ist die Richtung der größten Verminderung des Funktionswertes von e. Wir erhalten also für eine geeignete Schrittlänge $\lambda \in \mathbf{R}$ mit

$$\Delta = -\lambda \mathbf{g} \tag{8.7}$$

einen neuen Parameterwert $\mathbf{p} = \mathbf{p}_0 + \Delta$, der den Wert der Fehlerfunktion vermindert.
Der große Vorteil des Gradientenabstiegs ist, dass wir – sofern wir uns nicht bereits in einem Minimum befinden – auf jeden Fall eine Verminderung des Funktionswerts erhalten. Jedoch entspricht der Gradientenabstieg einer linearen Approximierung der Funktion, was eine langsame Konvergenz mit sich bringt.

8.2.1.2 Newton-Verfahren

Das Newton-Verfahren beruht auf der Taylor-Entwicklung von e. Eine quadratische Abschätzung von e ist gegeben durch das Taylorpolynom zweiter Ordnung und damit durch

$$e(\mathbf{p}) = e(\mathbf{p}_0) + \mathbf{g}\Delta + \frac{1}{2}\Delta^T H \Delta \tag{8.8}$$

Der Gradient $\mathbf{g}$ ist dabei definiert durch (8.6) und H ist die Hessematrix (also die Matrix der zweiten Ableitungen) an der Stelle $\mathbf{p}_0$.
Da unsere Approximierung quadratisch ist, hat sie ein eindeutiges Minimum, das wir erhalten, indem wir die Nullstelle der Ableitung berechnen. Die Ableitung von (8.8) nach Δ ist

$$e'(\mathbf{p}) = \mathbf{g} + \Delta H \tag{8.9}$$

Indem wir sie auf Null setzen und nach Δ umformen erhalten wir

$$\Delta = -H^{-1}\mathbf{g} \tag{8.10}$$

Das Newton-Verfahren beruht auf der Hoffnung, dass sich die Funktion in der Nähe des Minimums durch eine quadratische Funktion gut approximieren lässt und führt im Allgemeinen zu schnellerer Konvergenz als der Gradientenabstieg. Nachteile sind, dass eine Verbesserung des Funktionswertes nicht garantiert ist, falls eine quadratische Approximation an der Stelle $\mathbf{p}_0$ nicht angemessen ist, und dass die Berechnung der Hessematrix sehr aufwändig ist.

8.2.1.3 Gauß-Newton-Verfahren

Das Gauß-Newton-Verfahren ist eine Modifikation des Newton-Verfahrens, um das Problem der aufwändigen Berechnung der Hessematrix zu umgehen. Die Hessematrix lässt sich berechnen durch die Ableitung von (8.6) und somit durch

$$H = J^T J + J'^T \boldsymbol{\epsilon}(\mathbf{p}_0) \tag{8.11}$$

J' ist hier ein 3-dimensionales Feld aus den Ableitungen von J und das Produkt $J'^T \boldsymbol{\epsilon}(\mathbf{p}_0)$ ist die Summe über die jeweils mit ihrer Hessematrix multiplizierten Komponenten von $\boldsymbol{\epsilon}(\mathbf{p}_0)$.
Im Gauß-Newton-Verfahren wird der zweite Term weggelassen und die Hessematrix abgeschätzt durch

$$H = J^T J \tag{8.12}$$

Da nun die Hessematrizen von $\boldsymbol{\epsilon}(\mathbf{p}_0)$ nicht berechnet werden müssen, bringt dies eine erhebliche Einsparung an Rechenzeit und die Approximierung ist meist ausreichend.

8.2.1.4 Levenberg-Marquardt-Algorithmus

Mit dem Levenberg-Marquardt-Algorithmus wird versucht, die Vorteile des Gradientenabstiegs und des Gauß-Newton-Verfahrens zu verbinden. Die Update-Gleichung im Gradientenabstieg (8.7) mit (8.6) lässt sich ausdrücken durch

$$\lambda \Delta = -J^T \boldsymbol{\epsilon}(\mathbf{p}_0)$$

In dieser Gleichung bewirkt ein kleines λ eine große Schrittweite, es ist das multiplikative Inverse des λ aus (8.7). Die Update-Gleichung im Gauß-Newton-Verfahren (8.10) mit (8.6) und (8.12) lässt sich ausdrücken durch

$$J^T J \Delta = -J^T \boldsymbol{\epsilon}(\mathbf{p}_0)$$

Beides wird nun kombiniert zu

$$(J^T J + \lambda I)\Delta = -J^T \boldsymbol{\epsilon}(\mathbf{p_0}) \tag{8.13}$$

I ist die Identitätsmatrix. Durch den Parameter λ bewegt sich diese Gleichung zwischen einem Gradientenabstieg, dessen Schrittweite gleichzeitig durch λ bestimmt wird, und dem Gauß-Newton-Verfahren. Kann der Funktionswert in einem Iterationsschritt nicht vermindert werden, wird λ erhöht, um in die Nähe des Gradientenabstiegs zu kommen, der für eine ausreichend kleine Schrittweite eine Verringerung garantiert. Die Schrittweite wird durch Erhöhung von λ gleichzeitig verkleinert. Nach einem erfolgreichen Iterationsschritt dagegen wird λ gesenkt, um von der schnelleren Konvergenz des Gauß-Newton-Verfahrens zu profitieren.
Für die Anpassung von λ existieren verschiedene Strategien. Üblich ist z. B. ein Startwert $\lambda = 0.001$ und eine Verringerung bzw. Erhöhung jeweils um den Faktor 10.

8.2.2 Anwendung des Levenberg-Marquardt-Algorithmus beim Bundle Adjustment

Der Levenberg-Marquardt-Algorithmus wurde in Abschnitt 8.2.1 so beschrieben, dass er sich auf das durch (8.2) gegebene Minimierungsproblem problemlos anwenden lässt. Die Parameter, die wir anpassen wollen, sind die in einer initialen Rekonstruktion (nach Abschnitt 7.3.2) berechneten Kameras und Raumpunkte. Im Folgenden bezeichnen wir mit *Kameraparameter* die berechneten Einträge der Kameramatrizen und mit *Punktparameter* die berechneten Koordinaten der Raumpunkte. Die Anzahl der verwendeten Bilder und damit Kameramatrizen sei m, die Anzahl der berechneten Raumpunkte n. Wir fassen die Kamera- und Punktparameter in einem Parametervektor $\mathbf{p}_0$ der Länge $12m + 4n$ zusammen. Aus den inhomogenen Koordinaten aller gemessener Bildpunkte erstellen wir den Zielvektor $\mathbf{z}$ der Länge $2mn$. In $\mathbf{z}$ sind zu *allen* Raumpunkten die gemessenen Bildpunkte in jedem Bild verzeichnet. Ist ein Raumpunkt in einem Bild nicht sichtbar, werden die entsprechenden Einträge im Vektor auf Null gesetzt. Wir verwenden im Vektor der Bildkoordinaten inhomogene Koordinaten, da der Algorithmus ansonsten unter zusätzlichem Rechenaufwand die Skalierungen der homogenen Bildkoordinaten angleichen würde.
Wir benötigen eine Routine, die aus dem Parametervektor $\mathbf{p}$ einen Vektor $\mathbf{f}(\mathbf{p})$ berechnet, in dem die inhomogenen Koordinaten aller aus den Parametern berechneten Bildpunkte stehen, in der gleichen Reihenfolge wie im Zielvektor $\mathbf{z}$. Diese Routine lässt sich einfach erstellen. Zur Berechnung eines Bildpunktes werden aus dem Parametervektor die Einträge der entsprechenden Kameramatrix und die Koordinaten des entsprechenden Raumpunktes gewonnen und durch (2.7) die homogenen Koordinaten der Projektion berechnet und in inhomogene Koordinaten umgerechnet. Die Bildkoordinaten eines nicht sichtbaren Raumpunktes werden auch hier auf Null gesetzt.
Damit haben wir die Gleichung (8.3) modelliert und können für die initiale Rekonstruktion (8.4) und (8.5) einfach berechnen. Die Fehlernorm $e(\mathbf{p})$ aus (8.5) entspricht dann bis auf den konstanten Faktor 1/2 genau dem Rückprojektionsfehler (8.1), da die Summe der quadrierten euklidischen Distanzen genau der Summe der quadrierten Differenzen der einzelnen Koordinaten entspricht.
Die Jacobi-Matrix J berechnen wir numerisch. Ihre Spalten bestehen aus partiellen Ableitungen

$$\frac{\partial \mathbf{f}}{\partial p_i}$$

die numerisch berechnet werden durch

$$\frac{\mathbf{f}(\mathbf{p}) - \mathbf{f}(\mathbf{p'}_i)}{\delta}$$

mit $\delta \in \mathbf{R}$, δ sollte in der Größenordnung 10^{-4} bewählt werden. $\mathbf{p'}_i$ geht aus $\mathbf{p}$ hervor, indem der i-te Parameter um δ auf $p'_i = p_i + \delta$ erhöht wird. Die Einträge der Jacobi-Matrix, die auf die Projektionen nicht sichtbarer Raumpunkte bezogen sind, werden auf Null gesetzt.
Somit können wir Gleichung (8.13) aufstellen, nach Δ lösen und damit den Levenberg-Marquardt-Algorithmus anwenden. Diese Berechnung ist jedoch aufgrund der großen Jacobi-Matrix J sehr aufwändig.

8.2.3 Eine schnelle Variante des Levenberg-Marquardt-Algorithmus für das Bundle Adjustment

Der in Abschnitt 8.2.2 vorgestellte Algorithmus soll nun so modifiziert werden, dass er in akzeptabler Laufzeit ausgeführt werden kann. Dies wird dadurch möglich, dass die Jacobi-Matrix eine spärlich besetzte Matrix ist und wir das ausnutzen können.
Im Folgenden sei $\mathbf{x}$ unser Zielvektor, also der Vektor der gemessenen Bildpunkte, und $\hat{\mathbf{x}} = \mathbf{f}(\mathbf{p})$ sei der Vektor der aus den Parametern berechneten Bildpunkte. $\mathbf{x}$ und $\hat{\mathbf{x}}$ sind also zusammengesetzt aus 2×1 - Vektoren $\mathbf{x}_{ij}$ bzw. $\hat{\mathbf{x}}_{ij}$, die die gemessenen bzw. berechneten Bildpunkte im j-ten Bild zum i-ten Raumpunkt repräsentieren. Der Fehlervektor an der Stelle $\mathbf{p}$ wird kurz als $\boldsymbol{\epsilon}$ bezeichnet und ist aus Vektoren $\boldsymbol{\epsilon}_{ij}$ zusammengesetzt, die die Differenz zwischen $\mathbf{x}_{ij}$ und $\hat{\mathbf{x}}_{ij}$ angeben.
Der erste Schritt des Algorithmus liegt in einer Partitionierung der Parameter.

8.2.3.1 Partitionierung der Parameter

Wir partitionieren den Parametervektor $\mathbf{p}$ durch $\mathbf{p} = (\mathbf{a}^T, \mathbf{b}^T)^T$ in zwei Vektoren $\mathbf{a}$ und $\mathbf{b}$, so dass $\mathbf{a}$ die Kameraparameter enthält und $\mathbf{b}$ die Raumkoordinaten der Punkte. Dadurch partitioniert sich auch Δ in $\Delta = (\Delta_{\mathbf{a}}^T, \Delta_{\mathbf{b}}^T)^T$. Die Jacobi-Matrix $J = \frac{\partial \mathbf{f}}{\partial \mathbf{p}}$ hat dann die Form $J = [\ A \mid B\ ]$ mit

$$A = \frac{\partial \mathbf{f}}{\partial \mathbf{a}}$$

und

$$B = \frac{\partial \mathbf{f}}{\partial \mathbf{b}}$$

Dadurch nimmt Gleichung (8.13) folgende Form an:

$$\left[\begin{pmatrix} A^TA & A^TB \\ B^TA & B^TB \end{pmatrix} + \lambda I \right] \begin{pmatrix} \Delta_{\mathbf{a}} \\ \Delta_{\mathbf{b}} \end{pmatrix} = - \begin{pmatrix} A^T\boldsymbol{\epsilon} \\ B^T\boldsymbol{\epsilon} \end{pmatrix}$$

Dies können wir schreiben als

$$\begin{pmatrix} U^* & W \\ W^T & V^* \end{pmatrix} \begin{pmatrix} \Delta_{\mathbf{a}} \\ \Delta_{\mathbf{b}} \end{pmatrix} = - \begin{pmatrix} \boldsymbol{\epsilon}_A \\ \boldsymbol{\epsilon}_B \end{pmatrix} \tag{8.14}$$

mit $U = A^TA$, $V = B^TB$, $W = A^TB$, $\boldsymbol{\epsilon}_A = A^T\boldsymbol{\epsilon}$ und $\boldsymbol{\epsilon}_B = B^T\boldsymbol{\epsilon}$. U^* und V^* gehen aus U und V hervor, indem ihre diagonalen Einträge mit dem Faktor $1 + \lambda$ multipliziert werden. Um (8.14) zu lösen, definieren wir eine Matrix $Y = WV^{*-1}$ und multiplizieren links an beide Seiten von (8.14) die Matrix

$$\begin{pmatrix} I & -Y \\ 0 & I \end{pmatrix}$$

und erhalten

$$\begin{pmatrix} U^* - YW^T & 0 \\ W^T & V^* \end{pmatrix} \begin{pmatrix} \Delta_{\mathbf{a}} \\ \Delta_{\mathbf{b}} \end{pmatrix} = - \begin{pmatrix} \boldsymbol{\epsilon}_A - Y\boldsymbol{\epsilon}_B \\ \boldsymbol{\epsilon}_B \end{pmatrix} \tag{8.15}$$

Die Gleichungen der oberen Zeile aus (8.15) sind nun unabhängig von $\Delta_{\mathbf{b}}$ und können geschrieben werden als

$$S\Delta_{\mathbf{a}} = -\boldsymbol{\epsilon}_A + Y\boldsymbol{\epsilon}_B \tag{8.16}$$

mit

$$S = U^* - YW^T$$

und nach $\Delta_{\mathbf{a}}$ aufgelöst werden. Die Gleichungen der unteren Zeile von (8.15) können geschrieben werden als

$$W^T\Delta_{\mathbf{a}} + V^*\Delta_{\mathbf{b}} = -\boldsymbol{\epsilon}_B$$

Wir formen sie um zu

$$V^*\Delta_{\mathbf{b}} = -\boldsymbol{\epsilon}_B - W^T\Delta_{\mathbf{a}}$$

und können daraus nach Einsetzen der für $\Delta_{\mathbf{a}}$ berechneten Werte auch $\Delta_{\mathbf{b}}$ berechnen.

8.2.3.2 Ausnutzen der spärlichen Struktur der Jacobi-Matrix

Das Partitionieren der Daten an sich hat uns noch keinen Vorteil gebracht, eröffnet uns aber Möglichkeiten, die spärliche Struktur der Jacobi-Matrix auszunutzen.
Die Jacobi-Matrix hat $4n + 12m$ Spalten. Eine Zeile entpricht der Ableitung einer Koordinate eines projizierten Bildpunktes und ist daher nur von einer Kameramatrix und einem Raumpunkt, also 16 Parametern abhängig. Alle anderen Einträge der Zeile bestehen aus Nullen.
Durch jeden Bildpunkt entsteht ein 2×12-Block mit den Ableitungen nach den Kameraparametern und ein 2×4-Block mit den Ableitungen nach den Koordinaten des zugehörigen Raumpunktes. Diese Blöcke werden für jeden Bildpunkt berechnet. Wir erhalten also nm Ableitungsmatrizen A_{ij} für die Kameraparameter durch

$$A_{ij} = \frac{\partial \hat{\mathbf{x}}_{ij}}{\partial \mathbf{a}_j}$$

und nm Ableitungsmatrizen B_{ij} für die Raumpunkte durch

$$B_{ij} = \frac{\partial \hat{\mathbf{x}}_{ij}}{\partial \mathbf{b}_i}$$

Wir berechnen außerdem durch

$$\boldsymbol{\epsilon}_{ij} = \hat{\mathbf{x}}_{ij} - \mathbf{x}_{ij}$$

zu jedem Bildpunkt einen 2×1-Fehlervektor, der die Abweichung des berechneten zum tatsächlichen Bildpunkt enthält. Analog zu den Matrizen U, V, W, Y und den Vektoren $\boldsymbol{\epsilon}_A$ und $\boldsymbol{\epsilon}_B$ aus Abschnitt 8.2.3.1 berechnen wir nun

- je m Matrizen $U_j = \sum_i A_{ij}^T A_{ij}$ und $U_j^* = U_j + \lambda I$
- je n Matrizen $V_i = \sum_j B_{ij}^T B_{ij}$ und $V_i^* = V_i + \lambda I$

- nm Matrizen $W_{ij} = A_{ij}^T B_{ij}$
- nm Matrizen $Y_{ij} = W_{ij} V_i^{*-1}$
- m Vektoren $\boldsymbol{\epsilon}_{\mathbf{a}_j} = \sum_i A_{ij}^T \boldsymbol{\epsilon}_{ij}$
- n Vektoren $\boldsymbol{\epsilon}_{\mathbf{b}_i} = \sum_j B_{ij}^T \boldsymbol{\epsilon}_{ij}$

Wir können dann die Matrix S aus (8.16) berechnen, indem wir sie als $m \times m$ Block-Matrix definieren mit Blöcken

$$S_{jk} = \begin{cases} -\sum_i Y_{ij} W_{ik}^T & \text{für } j \neq k \\ -\sum_i Y_{ij} W_{ik}^T + U_j^* & \text{für } j = k \end{cases}$$

Die rechte Seite der Gleichung stellen wir dar durch einen Vektor $\mathbf{e} = (\mathbf{e}_1^T, ..., \mathbf{e}_m^T)^T$ mit

$$\mathbf{e}_j = -\boldsymbol{\epsilon}_{\mathbf{a}_j} + \sum_i Y_{ij} \boldsymbol{\epsilon}_{\mathbf{b}_i}$$

Wir können dann die Gleichung

$$S \Delta_{\mathbf{a}} = \mathbf{e}$$

nach $\Delta_{\mathbf{a}} = (\Delta_{\mathbf{a}_1}^T, ..., \Delta_{\mathbf{a}_m}^T)^T$ lösen.
Anschließend berechnen wir $\Delta_{\mathbf{b}} = (\Delta_{\mathbf{b}_1}^T, ..., \Delta_{\mathbf{b}_m}^T)^T$ durch

$$\Delta_{\mathbf{b}_i} = V_i^{*-1}(-\boldsymbol{\epsilon}_{\mathbf{b}_i} - \sum_j W_{ij}^T \Delta_{\mathbf{a}_j})$$

Wir sind völlig analog zu Abschnitt 8.2.3.1 vorgegangen. Das am Ende berechnete $\Delta = (\Delta_{\mathbf{a}}^T, \Delta_{\mathbf{b}}^T)^T$ ist identisch. Anstatt jedoch wie zuvor mit sehr großen Matrizen zu rechnen, wurden hier jeweils nur die Blöcke berechnet, die tatsächlich Daten enthalten. Dies bringt enorme Verbesserungen der Laufzeit mit sich.
Sollte der Rückprojektionsfehler $e(\mathbf{p}') = e(\mathbf{p} + \Delta)$ gegenüber $e(\mathbf{p})$ nicht verringert worden sein, wird λ erhöht und wir berechnen damit ein neues Δ. Alle von λ unabhängigen Werte, also alle A_{ij}, B_{ij}, $\boldsymbol{\epsilon}_{ij}$, U_j, V_i, W_{ij}, $\boldsymbol{\epsilon}_{\mathbf{a}_j}$ und $\boldsymbol{\epsilon}_{\mathbf{b}_i}$, müssen dabei nicht neu berechnet werden. Nach einer erfolgreichen Iteration, also einer Verminderung des Rückprojektionsfehlers, wird der neue Parametervektor $\mathbf{p}' = \mathbf{p} + \Delta$ übernommen und λ für den nächsten Iterationsschritt gesenkt. Wir iterieren, bis der Rückprojektionsfehler in einem Minimum konvergiert.
Für eine ausführlichere Erläuterung des Algorithmus siehe [HZ04].

8.3 Optimierung durch einen evolutionären Algorithmus

Vor allem bei einer schlechten initialen Rekonstruktion hat das Bundle Adjustment den Nachteil, dass der Rückprojektionsfehler nur bis zu einem lokalen Minimum verringert wird, das meistens weit entfernt von einer optimalen Lösung ist. Daher lohnt es sich, auch eine alternative Methode zur Optimierung zu berücksichtigen, die lokale Minima überwinden kann – einen evolutionären Algorithmus.

8.3.1 Kurzer Überblick über evolutionäre Algorithmen

Es soll hier nur kurz und allgemein die Funktionsweise evolutionärer Algorithmen erläutert werden. Eine ausführliche Einführung und tiefgehende Behandlung bietet beispielsweise [Jan03].
Evolutionäre Algorithmen sind randomisierte Suchheuristiken, deren Vorbild – wie der Name verrät – die natürliche Evolution ist. Gegeben ist eine *Population* von Datenpunkten, beispielsweise repräsentiert durch Vektoren $\mathbf{x} \in \mathbf{R}^n$, und eine *Fitnessfunktion* $f : \mathbf{R}^n \to \mathbf{R}$, die diese Vektoren bewertet, indem sie ihnen einen *Fitnesswert* zuordnet. Ein evolutionärer Algorithmus läuft dann im Allgemeinen in folgenden Phasen ab:

1. **Bewertung der Anfangspopulation.** Alle Vektoren der Population werden durch die Fitnessfunktion bewertet.

2. **Selektion zur Reproduktion.** Aufgrund der Ergebnisse der Bewertungen werden Vektoren zur Reproduktion – also zur Erzeugung neuer Vektoren – ausgewählt. Je höher der Fitnesswert eines Vektors, desto größer die Wahrscheinlichkeit, dass er ausgewählt wird.

3. **Reproduktion und Variation**. Aus den ausgewählten Vektoren werden Nachkommen erzeugt, indem die Daten zweier Vektoren miteinander kombiniert werden und / oder indem die Daten eines Vektors mutiert werden, d.h. zufällig ausgewählte Einträge leicht verändert werden.

4. **Bewertung der Nachkommen.** Die Nachkommen werden mit der Fitnessfunktion bewertet.

5. **Selektion zur Ersetzung.** Aufgrund der Bewertungen der Population und der Nachkommen werden Vektoren aus der Population ausgewählt, die entfernt werden, sowie Nachkommen, die der Population hinzugefügt werden. Je kleiner der Fitnesswert eines Vektors, desto größer die Wahrscheinlichkeit, dass er entfernt wird. Je größer der Fitnesswert eines Nachkommen, desto größer die Wahrscheinlichkeit, dass er in die Population übernommen wird. Im Allgemeinen wird die Größe der Population konstant gehalten.

6. **Testen auf Abbruchkriterium.** Die Phasen ab 2. werden wiederholt, bis ein Abbruchkriterium erfüllt wird (z.B. keine Nachkommen mit höherem Fitnesswert erzeugt werden).

Es gibt zahlreiche Varianten der evolutionären Algorithmen, die sich in unterschiedlichen Strategien der Selektion und Variation unterscheiden sowie durch Parameter wie unterschiedlichen Populationsgrößen und Anzahl an Nachkommen. Es wird gehofft, dass die Vektoren sich mit der Zeit an die Fitnessfunktion „anpassen“, die Fitnesswerte der Vektoren also steigen und insbesondere der maximale Fitnesswert eines Vektors in der Population ständig erhöht wird. In einem Minimierungsproblem kann die zu minimierende Funktion als Fitnessfunktion benutzt werden und der Fitnesswert analog zur beschriebenen Vorgehensweise minimiert anstatt maximiert werden. Evolutionäre Algorithmen haben den Vorteil, dass sie auf beliebige Funktionen angewendet werden können, über die nichts bekannt ist und die keinen weiteren Vorgaben unterliegen – sie müssen lediglich einen Wert liefern, der als Fitnesswert interpretiert werden kann. Ist jedoch mehr über die Funktion

bekannt – so wie im Fall des Bundle Adjustments – sind Optimierungsstrategien wie der Levenberg-Marquardt-Algorithmus deutlich überlegen und im Allgemeinen viel schneller. Evolutionäre Algorithmen haben jedoch einen zweiten großen Vorteil: Sie können über lokale Minima hinwegkommen. Das macht sie auch für bekannte Funktionen, für die eigentlich schnellere Optimierungsverfahren existieren, interessant.

8.3.2 Ein einfacher evolutionärer Algorithmus als Alternative zum Bundle Adjustment

Wir benutzen einen recht einfachen evolutionären Algorithmus, der zumindest für den entscheidenden Punkt – die Überwindung lokaler Minima – ausreichend ist. Er funktioniert nach dem in Abschnitt 8.3.1 beschriebenen Prinzip und zeichnet sich vor allem dadurch aus, dass die Population nur aus einem einzelnen Vektor besteht. Er ist eine Variante des *(1+1) ES* - Algorithmus (siehe [Jan03] - ES für Evolutionsstrategie) und läuft folgendermaßen ab:

- Als Fitnessfunktion wird (8.1) verwendet, sie soll minimiert werden. Ein kleiner Wert der Funktion steht also für eine hohe „Fitness“ des Vektors.

- Die Population besteht aus einem einzelnen Vektor $\mathbf{p}$, der aus allen berechneten Kameramatrizen und Raumpunkten besteht und initial durch die in Abschnitt 7.3.2 beschriebene Rekonstruktion gewonnen wird.

- Der Vektor ist partitioniert in Kamera- und Punktparameter, die mit unterschiedlichen Wahrscheinlichkeiten variiert werden. Der Parameter m_P gibt die Wahrscheinlichkeit an, dass ein Kameraparameter variiert wird. Der Parameter $m_{\mathbf{X}}$ gibt die Wahrscheinlichkeit an, dass ein Punktparameter variiert wird. Die Varianzen der Mutationen sind durch Parameter v_P und $v_{\mathbf{X}}$ gegeben. Eine initiale Wahl der Parameter ist beispielsweise $m_P = m_{\mathbf{X}} = 0.01$ und $v_P = v_{\mathbf{X}} = 1$

- Aus dem Vektor $\mathbf{p}$ wird ein neuer Vektor $\mathbf{p}'$ erzeugt, indem jedes Element des Vektors mit einer durch m_P bzw. $m_{\mathbf{X}}$ gegebenen Wahrscheinlichkeit mutiert. Ein zur Mutation ausgewähltes Element p_i wird ersetzt durch eine normalverteilte Zufallszahl p_i' mit Erwartungswert p_i und Varianz $p_i \cdot v_P$ bzw. $p_i \cdot v_{\mathbf{X}}$

- Die Vektoren $\mathbf{p}$ und $\mathbf{p}'$ werden anhand des durch sie erzeugten Rückprojektionsfehlers, also Funktionwertes von (8.1), verglichen und der Vektor, der den kleineren Fehler erzeugt, wird übernommen.

- Die Parameter m_P, $m_{\mathbf{X}}$, v_P und $v_{\mathbf{X}}$ werden bei jeder Iteration ebenfalls mutiert und ersetzt durch normalverteilte Zufallszahlen mit ihren alten Werten als Erwartungswert. Diese Mutationen werde dann übernommen, wenn im jeweiligen Iterationsschritt eine deutlich überdurchschnittliche Verminderung des Rückprojektionsfehlers erreicht wurde.

- Es werden abwechselnd nur Kameraparameter, nur Punktparameter oder beide Parameterarten variiert.

8.4 Iterative Optimierung

Während des Erstellens der Rekonstruktion nach Abschnitt 7.3.2 produzieren wir ständig Fehler, die sich multiplizieren. Daher bietet es sich an, eine Optimierung der Rekonstruktion nicht erst am Ende auf die Gesamtrekonstruktion anzuwenden, sondern bereits nach jedem Rekonstruktionsschritt. Die Rekonstruktion Γ, die wir erstellen, wird ständig erweitert. Die Optimierungsmethoden können jederzeit darauf angewendet werden und bis auf die zusätzliche Rechenzeit spricht nichts dagegen, das nach jedem Hinzufügen einer Teilrekonstruktion zu tun.

Kapitel 9

Ergebnisse

In Anhang B ist eine Testreihe von etwa 150 Versuchen angegeben. Es wurden in jedem Versuch die gleichen Punktkorrespondenzen verwendet, die sonstigen Bedingungen wurden variiert. Für eine Rekonstruktion wurden maximal acht Bilder, also sieben Teilrekonstruktionen, verwendet. Die verwendete Bildsequenz beschreibt eine Drehung des Objektes von etwa 270° mit variierender Höhe der Kamera. In vielen Versuchen wurde nur ein Teil dieser Sequenz benutzt.
Es wurden entweder nur eine oder beide Optimierungsmethoden verwendet. Der Levenberg-Marquardt-Algorithmus wurde iteriert, bis er ein Minimum gefunden hatte, allerdings wurden maximal 100 Iterationen durchgeführt. Die Anzahl an Iterationen des evolutionären Algorithmus variierte und wurde, da der Algorithmus im Allgemeinen nicht konvergiert, zufällig gewählt, im Schnitt wurden 30.000 Iterationen durchgeführt. Wurden beide Optimierungsmethoden angewandt, so wurde zunächst der Levenberg-Marquardt-Algorithmus benutzt, anschließend der evolutionäre Algorithmus und schließlich erneut der Levenberg-Marquardt-Algorithmus.
In etwa der Hälfte der Versuche wurde iterativ optimiert. Bei der iterativen Anwendung beider Verfahren wurde der Levenberg-Marquardt-Algorithmus nach jedem Schritt dreimal im Wechsel mit dem evolutionären Algorithmus durchgeführt, der evolutionäre Algorithmus wurde hier im Schnitt je 8.000 Mal iteriert. Die Ergebnisse können folgendermaßen zusammengefasst werden:

- Die Fehler der Rekonstruktionen variieren sehr. Dies erklärt sich dadurch, dass viele Zufallselemente (vor allem RANSAC) Einfluss auf das Ergebnis nehmen.
- Je mehr Bilder und damit Teilrekonstruktionen verwendet werden, desto größer ist die Gefahr, dass der Fehler sehr groß wird.
- Hat der Levenberg-Marquardt-Algorithmus ein lokales Minimum gefunden, kann der evolutionäre Algorithmus den Fehler in der Regel weiter vermindern. Wird anschließend erneut der Levenberg-Marquardt-Algorithmus angewandt, kann auch er erneut zur Minimierung beitragen. Es sieht also so aus, als könnten sich die Algorithmen gut ergänzen.
- Eine Optimierung kann das Ergebnis im Allgemeinen deutlich verbessern. In den Tabellen in Anhang B ist das am Unterschied der Fehler vor und nach der abschließenden Optimierung bei den Versuchen ablesbar, die nicht iterativ optimiert wurden.

- Auch eine Optimierung kann leider nicht verhindern, dass in einer Vielzahl der Versuche erhebliche Fehler entstehen.

- Eine iterative Optimierung kann keinen geringeren Fehler garantieren, vor allem bei einer höheren Anzahl an verwendeten Bildern kann sie aber die Chancen, den Fehler in einem mäßigen Rahmen zu halten, erhöhen.

Als Fazit muss man feststellen, dass das System momentan leider nicht zuverlässig ein gutes projektives Modell erstellen kann. Die wesentliche Ursache dafür dürfte in einer unzuverlässigen Rekonstruktion der 3D-Punkte durch die hier benutzte Trianguliuerungsmethode liegen. Diese Problematik wird durch die mehrfachen Transformationen der 3D-Punkte verschärft. Ein Ausweg wäre die Verwendung des *Trifocal Tensor* (siehe dazu Abschnitt 10.2.1.5), mit dessen Hilfe eine wesentlich genauere Rekonstruktion möglich seien könnte.
Wir wollen hier beispielhaft zwei eher geglückte Rekonstruktionen mit vergleichsweise geringem Fehler herausgreifen – eine Rekonstruktion in der fünf Bilder verwendet wurden sowie eine Rekonstruktion aus acht verwendeten Bildern. Beide Rekonstruktionen wurden durch iterative Anwendung des Levenberg-Marquardt- und des evolutionären Algorithmus gewonnen.

9.1 Eine beispielhafte Rekonstruktion aus fünf verwendeten Bildern

In Abbildung 9.1 sind die mit den berechneten Kameramatrizen in die jeweiligen Bilder projizierten berechneten Raumpunkte als kleine Kreise dargestellt. Die Punkte werden allerdings nur in den Bildern dargestellt, in denen ein zugehöriger Bildpunkt, der zur Berechnung des Raumpunktes verwendet wurde, vorhanden ist. Im optimalen Fall müsste der Raumpunkt genau an diesen Bildpunkt zurückprojiziert werden. Ein Strich deutet zu dessen tatsächlicher Position. Die Länge eines Strichs drückt also den jeweiligen Rückprojektionsfehler aus. Das Ergebnis sieht recht erfreulich aus, denn nur bei wenigen Punkten gibt es deutliche Abweichungen.

In Abbildung 9.2 sind die Projektionen des gesamten Modells durch die berechneten Kameramatrizen in alle Bilder zu sehen. Hier wird deutlich, dass vor allem im ersten und im letzten Bild viele Punkte nicht auf dem abgebildeten Objekt liegen. Hier gelangen wir jedoch an die Grenzen des Systems - wir haben aus den gegebenen Informationen ein mögliches Modell erstellt, dass diesen Informationen weitgehend entspricht. Um das Modell weiter zu verbessern, wären weitere Informationen – zum Beispiel zusätzliche Ansichten der Raumpunkte in weiteren Bildern – notwendig. Die Frage, welche Informationen genau benötigt werden, um ein korrektes Modell zu erhalten, kann diese Arbeit nicht beantworten.

9.2 Eine beispielhafte Rekonstruktion aus acht verwendeten Bildern

Abbildung 9.3 stellt in der gleichen Art wie Abbildung 9.1 den Rückprojektionsfehler einer aus acht Bildern gewonnenen Rekonstruktion dar. Es ist mit Abstand die beste Rekonstruktion der Versuchsreihe, die aus allen Bildern erstellt wurde und damit den größten Teil des Objektes abdeckt. Doch auch diese Rekonstruktion kann nicht wirklich zufrieden stellen. Der Rückprojektionsfehler ist zwar in den meisten Bildern recht gering, jedoch führt der größere Fehler in einigen Bildern - vor allem im sechsten Bild - dazu, dass das gesamte Modell nicht mehr konsistent ist. Dies wird in Abbildung 9.4 deutlich, in der das gesamte Modell mit den berechneten Kameras dargestellt ist.

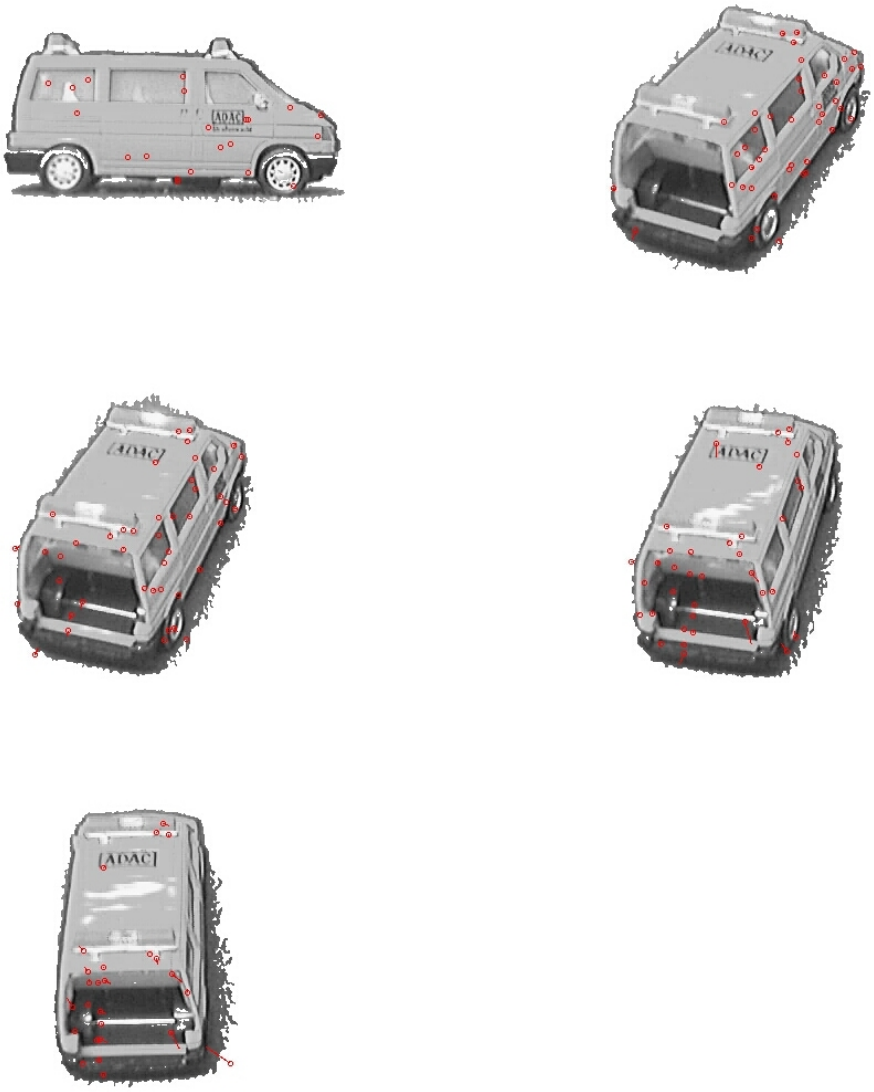


Abbildung 9.1: Die Rückprojektionsfehler einer Rekonstruktion aus fünf verwendeten Bildern.

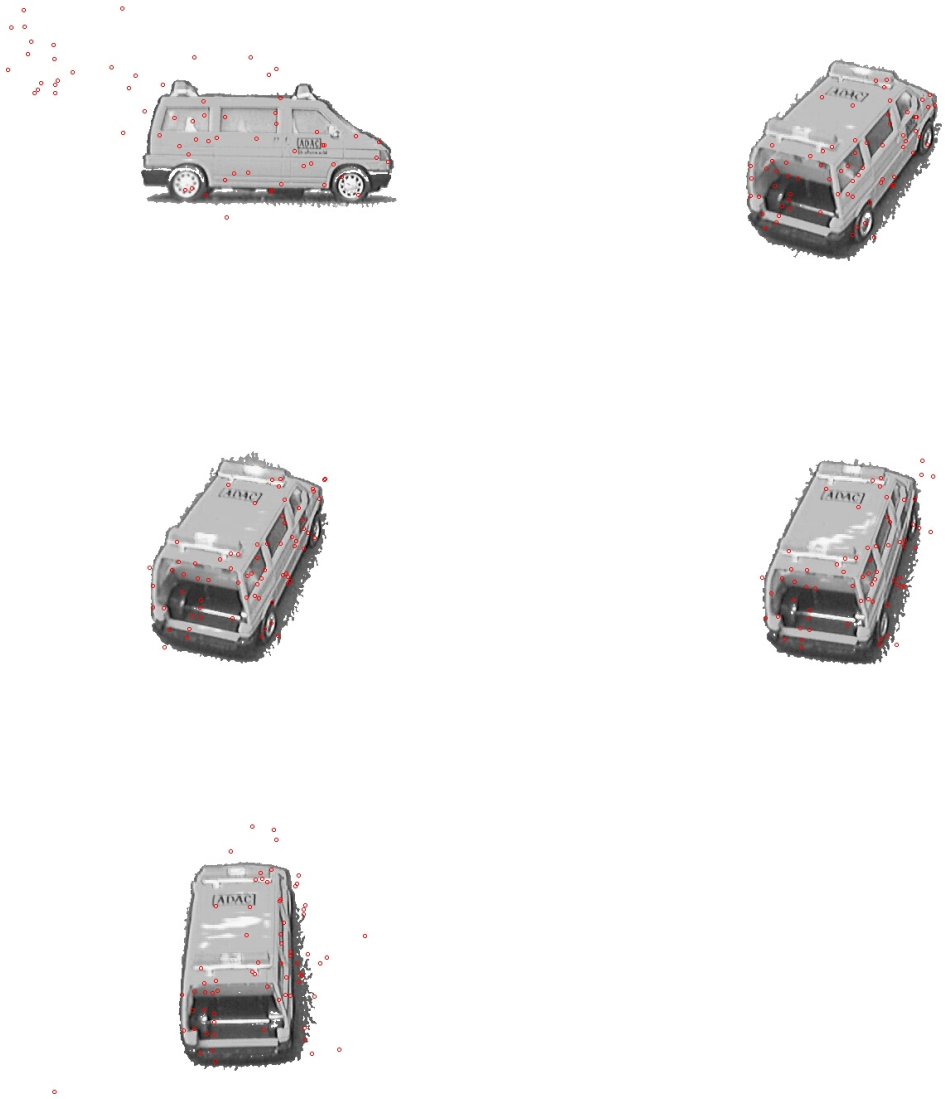


Abbildung 9.2: Projektion des gesamten Modells in die rekonstruierten Kameras.

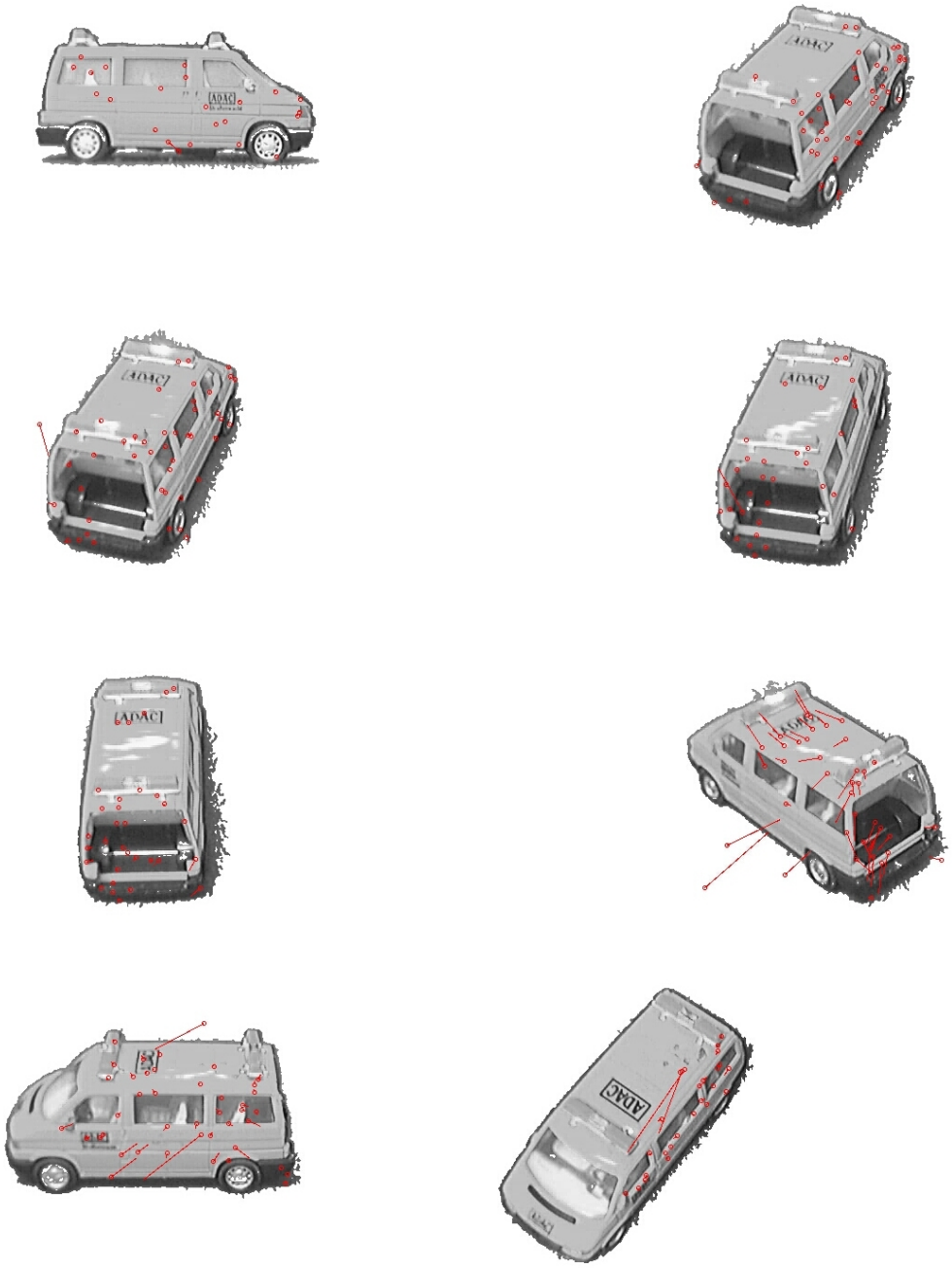


Abbildung 9.3: Die Rückprojektionsfehler einer Rekonstruktion aus acht verwendeten Bildern.

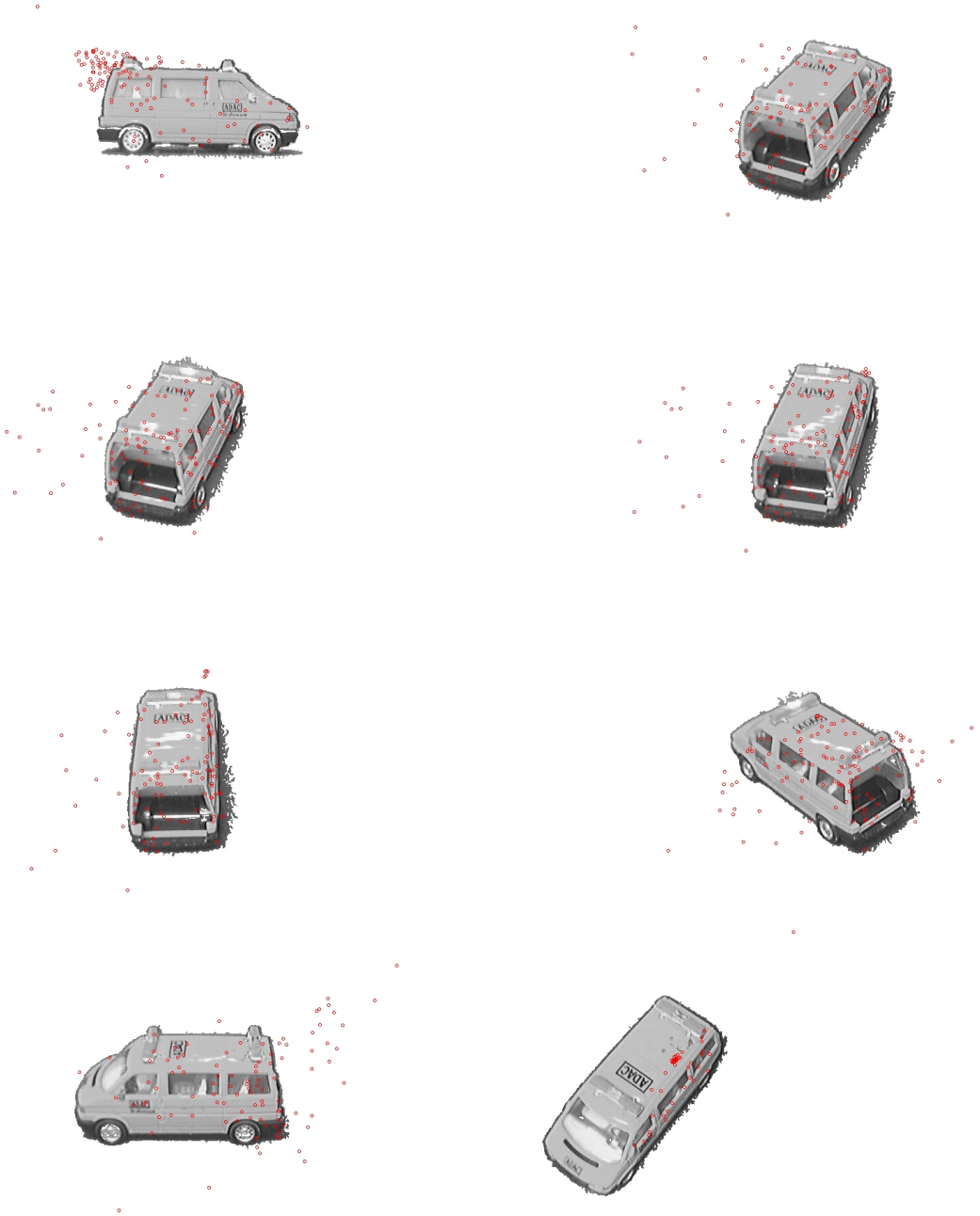


Abbildung 9.4: Projektion des gesamten Modells in die rekonstruierten Kameras.

Kapitel 10

Zusammenfassung und Ausblick

Das Ziel dieser Arbeit war es, alleine aus optischen Informationen ein 3D-Modell eines Objektes zu erstellen. Im Rahmen dieser Arbeit hieß das: ein projektives Modell in Form einer Punktwolke. Dass es grundsätzlich nur möglich ist, ein metrisches Modell zu erstellen (dass also eine gewisse Mehrdeutigkeit nie vermieden werden kann) und warum hier nur ein projektives Modell erstellt wurde, wurde im Abschnitt 7.1 ausführlich erläutert. Es wurden dort auch Bedingungen aufgezeigt, unter denen ein metrischen Modell erstellt werden kann. An dieser Stelle wollen wir nun in der Zusammenfassung das Erreichte diskutieren und im Ausblick einen Blick auf das noch zu Erreichende werfen.

10.1 Zusammenfassung

Das hier vorgestellte System deckt die gesamte Entstehung des Modells beginnend mit der Akquirierung der Bildsequenz ab. Die einzelnen Arbeitsschritte, die hier in jeweils eigenen Kapiteln abgehandelt wurden, lassen sich gut trennen und jeweils durch andere – insbesondere durch überlegene – Verfahren ersetzen. Da das Hauptaugenmerk der Arbeit auf der Entwicklung des Gesamtsystems lag, sind für jeden Arbeitsschritt aufwändigere Verfahren denkbar und zum Teil empfehlenswert (in Abschnitt 10.2.1 werden wir näher darauf eingehen).
Die Ergebnisse der Arbeit sind zufriedenstellend in der Hinsicht, dass tatsächlich ein zusammenhängendes Modell eines Objektes geschaffen werden kann und Probleme, die in den jeweiligen Arbeitschritten auftreten, identifiziert werden und ihre Folgen in den nächsten Arbeitsschritten beobachtet werden können. Das System kann in der jetzigen Form für eine längere Bildsequenz, bei der die Gesamtrekonstruktion aus vielen Teilrekonstruktionen zusammengesetzt werden muss, jedoch noch kein befriedigendes Resultat liefern. Der gemessene Rückprojektionsfehler ist meist erheblich und kann mit den Optimierungsmethoden reduziert, aber nicht auf ein akzeptables Maß gebracht werden. Zudem ist der Rechenaufwand dafür insgesamt recht groß.
Das System arbeitet nicht deterministisch. Auch bei einer konstanten Menge von Punktkorrespondenzen führen die nächsten Arbeitsschritte zu sehr unterschiedlichen Ergebnissen, da mit RANSAC und dem evolutionären Algorithmus zahlreiche Zufallselemente zum Resultat beitragen.

10.2 Ausblick

Der Ausblick lässt sich in zwei Punkte gliedern: Zum einen besteht bei vielen Arbeitsschritten Potential, das Ergebnis dieses Schrittes und damit des Gesamtsystems zu verbessern. In Abschnitt 10.2.1 werde ich für verschiedene Arbeitsschritte Möglichkeiten der Optimierung herausstellen.
Zum anderen kann das System ausgebaut werden, beispielsweise um eine metrische Rekonstruktion zu erstellen. Diese Möglichkeiten werden in Abschnitt 10.2.2 angesprochen.

10.2.1 Optimierung des Systems

Da jeder Arbeitsschritt auf die vorigen aufbaut, ist ein gutes Ergebnis eines jeden Moduls essentiell für ein gutes Endergebnis. Es wird zwar immer versucht, in einem Arbeitsschritt zu berücksichtigen, dass die vorigen Schritte Fehler produziert haben könnten. So wird versucht, bei der Berechnung der Epipolargeometrie fehlerhafte Korrespondenzen zu finden. Bei der Erstellung der 3D-Transformationen, um Rekonstruktionen anzugleichen, können nicht konsistente Rekonstruktionen einzelner Punkte gefunden werden. Die Optimierung versucht generell, den entstandenen Fehler zu minimieren. Dennoch ist das System darauf angewiesen, dass alle Module des Systems zuverlässig arbeiten und jedes Modul kann, liefert es fehlerhafte Ergebnisse, ein brauchbares Endresultat unmöglich machen. An dieser Stelle wollen wir uns einige kritische Stellen im System ansehen, die Probleme bereiten oder bereiten könnten und bessere Ergebnisse verhindern, und untersuchen wie sich der jeweilige Arbeitsschritt optimieren ließe.

10.2.1.1 Verbesserung der Segmentierung

Die Segmentierung ist in diesem System ein weniger kritischer Arbeitsschritt, da das in Kapitel 4 angewandte Segmentierungsverfahren für einen bekannten Hintergrund ein ausreichend gutes Ergebnis liefert. Lediglich der Schatten stellt ein Problem dar, das mit einfachen Mitteln nicht zu beheben ist. Punkte auf dem Schatten, die zum Tracking ausgewählt werden, könnten zu fehlerhaften Korrespondenzen führen. Es gibt Ansätze, einen Schatten in Bildern automatisiert zu erkennen und zu entfernen, beispielsweise mit Hilfe des Retinex-Algorithmus (siehe [FHD02]). Inwieweit sich dieses Verfahren auf die gegebenen Bilder anwenden lassen könnte, müsste sich jedoch erst erweisen.

10.2.1.2 Verbesserte Auswahl interessanter Punkte

Die in Abschnitt 5.1 benutzte Methode liefert Punkte, die das Objekt meist gut abdecken und die zum Tracking in der Regel gut geeignet sind. Es könnte sich dennoch lohnen, hier andere Verfahren (wie die dort bereits angesprochenen Verfahren aus [HS88] und [WL97]) anzuwenden. Eventuell ließen sich so Punkte finden, die im Tracking ein noch besseres Ergebnis liefern. Zudem könnte überlegt werden, ob einige Punkte (z. B. Eckpunkte) auf jeden Fall Teil des Modells sein sollten, um dieses aussagekräftiger zu machen.

10.2.1.3 Verbesserung des Trackings

Das Tracking (Kapitel 5) ist vielleicht der kritischte Abschnitt des Systems, da hier die Punktkorrespondenzen erstellt werden, die nicht nur für die weitere Rekonstruktion, sondern auch für deren Bewertung Grundlage sind. Offensichtlich führt eine falsche Korrespondenz zu einer unbrauchbaren Rekonstruktion eines Raumpunktes, was durch die Transformationen zwischen den Teilrekonstruktionen auch auf weitere Punkte kritische Auswirkungen haben kann. Gleichzeitig ist gerade das Tracking immer mit einer gewissen Unsicherheit behaftet, so dass fehlerhafte Korrespondenzen nie ganz ausgeschlossen werden können und auch korrekte Korrespondenzen nie hundertprozentig genau bestimmt werden. Glücklicherweise können falsche Korrespondenzen bei der Berechung der Epipolargeometrie häufig entlarvt werden.
Das Tracking wurde auf Grauwertbildern durchgeführt. Ein Anwenden des hier benutzten Verfahrens auf Farbbilder hieße, das Tracking in jedem Schritt auf den drei Farbkanälen einzeln durchzuführen die Ergebnisse zu kombinieren. Dies könnte ein Ansatz sein, um das Ergebnis zu verbessern. Ein weiterer Ansatz wäre es, ein Modell für die Bewegung des Bildpunktes von Bild zu Bild zu benutzen, das über die Annahme konstanter Positionsveränderung hinausgeht. Siehe hierzu [FP03].
Das Finden von Punktkorrespondenzen könnte außerdem durch Informationen, die die Epipolargeometrie uns liefert, unterstützt werden (siehe dazu folgenden Abschnitt).

10.2.1.4 Verbesserung zur Berechnung der Epipolargeometrie

Zur Berechnung der Fundamentalmatrix wird ein linearer Algorithmus in Verbindung mit RANSAC verwendet. Der große Vorteil dieses Verfahrens ist es, dass gleichzeitig Ausreißer, also fehlerhafte Korrespondenzen, ermittelt werden können. Wir könnten anschließend aus den als korrekt klassifizierten Korrespondenzen die Fundamentalmatrix mit nichtlinearen Verfahren neu berechnen, die unter der nun erfüllten Bedingung, dass keine fehlerhaften Korrespondenzen in den Daten enthalten sind, das Resultat verbessern würden. Beispielsweise könnte der Fehler, der durch die Fundamentalmatrix entsteht, durch den Levenberg-Marquardt-Algorithmus (siehe Abschnitt 8.2.1) minimiert werden.
Die Epipolargeometrie liefert zu allen Bildpunkten eine Epipolarlinie im anderen Bild, auf welcher der korrespondierende Punkt liegen sollte. Wir könnten diese Information in Ergänzung zum Tracking benutzen, um Punktkorrespondenzen zu berechnen oder zu korrigieren. Die Berechnung der Punktkorrespondenzen und der Epipolargeometrie würde sich dann gegenseitig bedingen und müsste iteriert werden, bis die Korrespondenzen und die Werte der Fundamentalmatrix stabil bleiben.
Eine weitere Optimierung könnte darin liegen, den Fehler der Fundamentalmatrix nicht durch die in Gleichung 6.4 erzeugte Abweichung zu messen, sondern mit der Fundamentalmatrix bereits eine Triangulierung durchzuführen und den durch die so erzeugten Raumpunkte bewirkten Rückprojektionsfehler zu verwenden.
Zu diesen Verbesserungen siehe [HZ04]. Bei allen genannten möglichen Verbesserungen müsste Nutzen und der vergrößerte Aufwand abgewogen werden.

10.2.1.5 Verbesserung der Rekonstruktion

Bei der Triangulierung in Abschnitt 7.2 haben wir ein einfaches lineares Verfahren verwendet. Auch hier gibt es aufwendigere nichtlineare Verfahren, die besser Ergebnisse liefern

könnten, siehe [HZ04].
Der entscheidende Unterschied, der die Ergebnisse dieses Arbeitsschrittes erheblich verbessern könnte – und damit vermutlich das größte Problem im bestehenden System mindern könnte – wäre jedoch die Verwendung des *Trifocal Tensor*.
Der Trifocal Tensor (siehe [HZ04]) ist die Erweiterung der Fundamentalmatrix auf drei Bilder. Es werden also Informationen aus drei Bildern kombiniert, um die wechselseitige Geometrie zwischen diesen Bildern zu berechnen und schließlich Raumpunkte zu erzeugen. Die damit erzeugte Rekonstruktion könnte wesentlich besser sein als die durch zwei Bilder berechnete Rekonstruktion. Dies legen z.B. [HZ04] und [FZ98a] nahe. Außerdem könnten Transformationen zwischen den Rekonstruktionen zuverlässiger erstellt werden, da nun Dreier-Gruppen von Bildern betrachtet würden und aufeinander folgende Gruppen, aus denen Rekonstruktionen erstellt werden, eine Überlappung von zwei Bildern haben könnten. Siehe hierzu [FZ98b].
Im hier vorgestellten System wurde zugunsten der Vervollständigung des Gesamtsystems auf die Implementierung des Trifocal Tensor verzichtet. Bei einer Optimierung des Systems läge hier wohl das größte Potential, um die Resultate zu verbessern.

10.2.1.6 Verbesserung der Optimierung

Die hier verwendeten Optimierungsverfahren lassen sich in vielfältiger Weise variieren. So könnte sowohl der Levenberg-Marquardt-Algorithmus im Bundle Adjustment als auch der evolutionäre Algorithmus auf inhomogene Parameter, also auf inhomogene Koordinaten der Raumpunkte und eine inhomogene Darstellung der Kameramatrizen, angewandt werden. Die Zahl der Parameter würde sich somit unter positiven Auswirkungen auf die Laufzeit geringfügig verringern und vermutlich würde sich auch das Verhalten der Algorithmen ändern. Es könnten auch verschiedene Kostenfunktionen angewandt werden (siehe [HZ04]). Vor allem beim evolutionären Algorithmus gibt es zahlreiche Strategien, die unter Umständen bessere Ergebnisse möglich machen könnten – beispielsweise durch eine größere Population und der Anwendung von Rekombination. Ob dies in der Gesamtlaufzeit Vorteile bringen könnte, müsste sich allerdings erst erweisen.
Ein weiterer Ansatz, um die Optimierung zu variieren, wären verschiedene Strategien, wann und worauf die Optimierung angewandt wird. Eine Alternative zur iterativen Optimierung wäre eine hierarchische Vorgehensweise. Die Sequenz könnte in kleine Teilsequenzen aufgeteilt werden, aus denen Teilrekonstruktionen erstellt würden, die jeweils optimiert und zu immer größeren und schließlich zur Gesamtrekonstruktion zusammengefügt werden könnten. Eine derartige Vorgehensweise wird in [FZ98b] vorgestellt.
Allgemein ist zu sagen, dass es in der Optimierung einen großen Spielraum an möglichen Veränderungen gibt, über deren Auswirkungen vorab jedoch nur spekuliert werden kann.

10.2.2 Weitere Entwicklung

Wurde ein projektives Modell in Form einer Punktwolke mit zufriedenstellender Qualität erzeugt, stehen zahlreiche Wege für eine weitere Entwicklung offen, die hier nur in aller Kürze erwähnt werden sollen.

10.2.2.1 Erstellen einer metrischen Rekonstruktion

Um eine metrische Rekonstruktion zu erstellen, müssen entweder die Kalibrationsmatrizen der Kameras bekannt sein oder es kann versucht werden, eine automatische Kalibrierung anhand der Bilder durchzuführen. Es gibt Verfahren zur automatischen Kalibrierung, die das Wissen über eine spezielle Art der Bewegung zwischen den Bildern nutzen. In [FCZ98] wird beispielsweise vorgestellt, wie aus Invarianten zwischen Bildern, die aus der Bewegung einer Drehscheibe entstehen, Wissen über die Kalibrierung gewonnen werden kann. Da diese Verfahren jedoch auf allgemeine Bewegungen nicht erweiterbar sind, sind andere Verfahren, die z.B. Wissen über einzelne Kameraparameter oder über deren Konstanz berücksichtigen, vielversprechender (siehe z.B. [HZ04]).

10.2.2.2 Erstellen verschiedener Modellformen

In einer metrischen Rekonstruktion sind – bis auf die durch eine Ähnlichkeits-Transformation gegebene Mehrdeutigkeit – auch die Kameramatrizen bekannt. Mit ihrer Hilfe lassen sich verschiedene Verfahren wie *Volume Intersection* auf die Bilder anwenden, um realitätsnahe Modelle der Objekte zu erhalten, siehe z.B. [SCMS01].
Zudem könnten aus der Punktwolke selbst z.B. durch Aufspannen von Polygonen weitere Modellformen erzeugt werden.

10.2.2.3 Erfassen einer Umgebung

Kann die Form eines Objektes korrekt erfasst werden, wäre ein möglicher weiterer Schritt der Übergang zur Modellierung einer komplexeren Szene und Umgebung. Die in dieser Arbeit besprochene Vorgehensweise ist in keiner Weise auf das Erfassen von Objekten spezialisiert sondern ist dazu geeignet, für allgemeinere und komplexere Aufgaben erweitert zu werden.

Anhang A

Matrix - Zerlegungen

A.1 Singulärwertzerlegung

Die Singulärwertzerlegung (oder kurz SVD für Singular Value Decomposition) ist ein mächtiges Werkzeug für numerische Berechnungen. Es wird hier gezeigt, wie sie in den Fällen, die für diese Arbeit eine Rolle spielen, angewendet werden kann. Dabei folge ich [HZ04]. Wie die SVD funktioniert und implementiert werden kann, wird an dieser Stelle nicht gezeigt. Hierzu siehe [GL89] bzw. [PTVF88] oder auch [Gro01]. Zur Durchführung der SVD standen Methoden der FLAVOR-Bibliothek zur Verfügung.

Eine $m \times n$ Matrix A mit $m \geq n$ kann zerlegt werden in $A = UDV^T$, so dass

- U eine $m \times n$ Matrix mit orthogonalen Spalten ist (d.h., es gilt $U^T U = I_{n \times n}$)
- D eine $n \times n$ Diagonalmatrix ist, deren Einträge nicht negativ und in absteigender Reihenfolge angeordnet sind
- V eine orthogonale $n \times n$ Matrix ist.

Die Einträge in D sind die *Singulärwerte* von A. Die Spalten von U sind die *linken Singulärvektoren* (Spalte U_i ist der zu d_{ii} gehörende Singulärvektor), die Spalten von V (also Zeilen von V^T) die *rechten Singulärvektoren*. Da die linken Singulärvektoren für uns nicht von Interesse sind, ist im Folgenden mit *Singulärvektor* immer ein rechter Singulärvektor gemeint.
Diese Matrixzerlegung hat folgende nützliche Eigenschaften:

- Die Anzahl an von Null verschiedenen Einträgen in D (also Singulärwerten) entspricht dem Rang der Matrix A.
- Hat die Matrix A einen Rang von $r < n$, so sind $n - r$ Singulärwerte 0. Die dazu gehörenden Singulärvektoren spannen den $n - r$ - dimensionalen Kern von A auf.

A.1.1 Verringern des Ranges einer Matrix mit Hilfe der Singulärwertzerlegung

Da die Anzahl an Singulärwerten, also Einträge in D, den Rang der Matrix bestimmen, können wir einfach Einträge von D auf 0 setzen, um den Rang der Matrix - wenn wir

sie anschließend wieder zusammensetzen - zu verringern. Sei also D' die so modifizierte Diagonalmatrix, r der Rang von A und k die Anzahl an Einträgen von D, die wir auf Null setzen. Dann ist $A' = UD'V^T$ eine Matrix von gewünschtem Rang $r - k$. Die Matrix A' soll sich natürlich möglichst wenig von A unterscheiden. Daher setzen wir immer die kleinsten Einträge von D auf 0, da diese bzw. die zugehörigen Singulärvektoren den geringsten Einfluß auf A haben.
Auf diese Weise erhalten wir eine Matrix A', für die $\|A - A'\|_F$ (Frobenius Norm) minimiert wird unter der Bedingung, dass A' den gewünschten Rang hat.

A.1.2 Lösen eines Gleichungssystems mit Hilfe der Singulärwertzerlegung

Das Lösen eines Gleichungssystems

$$A\mathbf{x} = \mathbf{0}$$

folgt direkt aus der zweiten Eigenschaft der Singulärwertzerlegung.
Zunächst müssen wir A erweitern, falls A mehr Spalten als Zeilen hat. In diesem Fall können wir einfach fehlende Zeilen mit Nullen auffüllen. Anschließend führen wir eine Singulärwertzerlegung durch. Die Singulärvektoren, deren zugehörige Singulärwerte gleich 0 sind, spannen den Lösungsraum auf.
Hat die Matrix A vollen Rang, ist die einzige Lösung trivialerweise der Vektor $\mathbf{0}$. Meistens ist in diesem Fall jedoch eine Näherung eines 1-dimensionalen Lösungsraums erwünscht. Eine Näherung daran ist der zum kleinsten Singulärwert gehörende Singulärvektor. Dies lässt sich dadurch motivieren, dass genau dieser Vektor den Lösungsraum zu A' aufspannen würde, wenn A' wie in Abschnitt A.1.1 beschrieben durch Verringern des Ranges aus A hervorgeht. Tatsächlich minimiert diese Methode $\|A\mathbf{x}\|$ unter der Bedingung $\|\mathbf{x}\| = 1$

Ein Gleichungssystem der Form $A\mathbf{x} = \mathbf{b}$ kann ebenfalls mit Hilfe der SVD gelöst werden, im überbestimmten Fall in Form einer Abschätzung. Hierzu siehe [HZ04].

A.2 RQ - Zerlegung

Die RQ-Zerlegung ist die Zerlegung einer Matrix A in

$$A = RQ$$

wobei

- R eine obere Dreiecksmatrix ist.
- Q eine orthogonale Matrix ist

Analog existiert eine QR-Zerlegung, in der die Matrix in $A = QR$ zerlegt wird, sowie LQ- und QL-Zerlegungen, in denen die Dreiecksmatrix eine untere Dreiecksmatrix ist. An dieser Stelle werden wir nur die RQ-Zerlegung betrachten und zwar nur für den Fall, dass A eine 3×3 Matrix ist (dies ist der wichtigste Fall und der einzige, der für diese Arbeit von Belang ist).

Um eine Matrix derart zu zerlegen benutzen wir 3-dimensionale *Givens-Rotationen*, das sind Rotationen um eine der Koordinatenachsen. Sie werden durch folgende Matrizen gegeben:

$$Q_x = \begin{pmatrix} 1 & 0 & 0 \\ 0 & c & -s \\ 0 & s & c \end{pmatrix} \; Q_y = \begin{pmatrix} c & 0 & s \\ 0 & 1 & 0 \\ -s & 0 & c \end{pmatrix} \; Q_z = \begin{pmatrix} c & -s & 0 \\ s & c & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

mit $c = \cos(\Theta)$ und $s = \sin(\Theta)$ für den Rotationswinkel Θ.
Wird eine Matrix mit einer Givens-Rotationsmatrix multipliziert, bleibt eine Spalte unverändert (z.B. die erste Spalte bei Multiplikation mit Q_x) und die anderen Spalten werden durch Linearkombinationen der beiden ursprünglichen Spalten ersetzt. Es ist dann möglich, den Winkel Θ so zu wählen, dass ein beliebiger Eintrag der beiden Spalten auf 0 gesetzt wird.
Das Prinzip der RQ-Zerlegung besteht nun darin, die Matrix A so mit drei Givens-Rotationen zu multiplizieren, dass drei Einträge auf 0 gesetzt werden und eine obere Dreiecksmatrix übrig bleibt. Dabei muss darauf geachtet werden, dass eine so gewonnene 0 nicht durch eine weitere Multiplikation verloren geht. Wir setzen zunächst a_{32} durch Multiplikation mit Q_x auf 0. Anschließend setzen wir a_{31} durch Multiplikation mit Q_y auf 0. Da durch diese Multiplikation die zweite Spalte unverändert bleibt, ist a_{32} weiterhin gleich 0. Zuletzt setzen wir a_{21} durch Multiplikation mit Q_z auf 0. Da die ersten beiden Spalten durch ihre Linearkombinationen ersetzt werden, bleiben a_{32} und a_{31} gleich 0. Wir erhalten eine obere Dreiecksmatrix R also durch

$$R = AQ_xQ_yQ_z$$

Daraus folgt

$$A = RQ_z^TQ_y^TQ_x^T = RQ$$

mit

$$Q = Q_z^TQ_y^TQ_x^T$$

Da die Rotationsmatrizen orthogonal sind, ist auch Q eine orthogonale Matrix.
Die gesuchten Rotationswinkel können durch recht einfache Gleichungen gefunden werden. Um beispielsweise a_{32} mit Q_x auf 0 zu setzen, müssen wir c und s finden, welche die Gleichung $sa_{32} + ca_{33} = 0$ erfüllen unter der Bedingung $c^2 + s^2 = 0$ (somit ist sichergestellt, dass ein Θ existiert mit $c = \cos(\Theta)$ und s = $\sin(\Theta)$). Dann ist

$$c = \frac{a_{32}}{\sqrt{a_{33}^2 + a_{32}^2}}$$

und

$$s = \frac{-a_{33}}{\sqrt{a_{33}^2 + a_{32}^2}} \; .$$

Anhang B

Ergebnisse in tabellarischer Übersicht

In den folgenden Tabellen sind die Ergebnisse von 145 Rekonstruktionen aufgelistet. Es wurden in jedem Versuch die gleiche Bildsequenz mit den gleichen Punktkorrespondenzen, in vielen Versuchen jedoch nur ein Teil der Sequenz verwendet. Die Spalte *Nr* gibt die laufende Nummer des Versuchs an. Die Spalten *I*, *L* und *E* stehen für „iterativ“, „Levenberg-Marquardt-Algorithmus“ und „Evolutionärer Algorithmus“. Ist der entsprechende Buchstabe eingetragen, bedeutet das, dass die entprechende Option zur Optimierung verwendet wurde. Die Spalte *FehlerVO* gibt den Rückprojektionsfehler an, der vor der Optimierung bestand (bei der iterativen Optimierung wurde jedoch bereits nach jeden Schritt optimiert). Die Spalte *Fehler* gibt den endgültigen Rückprojektionsfehler an. In der Spalte *Dauer* ist die Laufzeit des Durchgangs in Sekunden angegeben. Die Spalte *Größe* gibt die Größe des erstellten Modells, d.h. die Anzahl der rekonstruierten Raumpunkte an. Die Versuche werden hier getrennt nach Anzahl der verwendeten Bilder aufgelistet. Alle Tabellen sind nach dem am Ende vorhandenen Rückprojektionsfehler sortiert.
Diese Versuchsreihe wurde auf einem Rechner mit einem AMD Duron Prozessor (1,2 GHz) und 320 MB RAM durchgeführt.

B.1 Modelle aus 2 verwendeten Bildern

Nr	Bilder	I	L	E	FehlerVO	Fehler	Dauer	Größe
53	2	I	L	E	115	115	55	28
143	2	I	L	E	124	120	42	28
71	2	-	L	-	211	125	9	29
136	2	I	L	E	138	134	43	28
61	2	I	L	E	162	150	46	28
139	2	I	L	E	178	172	77	28
47	2	I	L	E	434	209	90	28
128	2	-	-	E	2.612	299	32	28

B.2 Modelle aus 3 verwendeten Bildern

Nr	Bilder	I	L	E	FehlerVO	Fehler	Dauer	Größe
108	3	-	L	E	10.645	5.567	141	50
81	3	I	L	E	11.687	9.129	144	49
120	3	I	L	-	11.765	11.746	65	51
80	3	I	L	-	42.346	39.266	108	50
54	3	-	L	E	293.447	118.102	201	49
4	3	-	L	E	403.703	128.982	94	49
97	3	I	L	E	213.809	145.607	179	49
126	3	I	L	E	214.833	208.214	173	50
69	3	-	-	E	244.661	211.798	174	49

B.3 Modelle aus 4 verwendeten Bildern

Nr	Bilder	I	L	E	FehlerVO	Fehler	Dauer	Größe
64	4	-	L	-	8.540	6.301	141	61
3	4	I	L	E	19.402	12.200	250	61
131	4	I	-	E	30.144	28.264	164	61
13	4	I	L	-	28.654	28.636	76	61
95	4	I	L	E	43.147	35.160	380	61
12	4	-	L	E	136.622	49.506	155	63
73	4	-	L	E	$> 10^6$	55.466	266	61
77	4	I	L	E	73.201	60.954	687	64
33	4	I	L	E	74.087	66.675	568	60
151	4	-	L	-	83.771	83.771	116	59
24	4	I	L	E	112.017	86.365	236	66
49	4	-	L	E	$> 10^6$	156.649	211	64
11	4	-	L	E	216.523	161.172	283	59
1	4	I	-	E	542.746	462.856	386	62
40	4	I	L	E	699.798	589.823	409	61
41	4	-	-	E	$> 10^6$	790.674	324	57

B.4 Modelle aus 5 verwendeten Bildern

Nr	Bilder	I	L	E	FehlerVO	Fehler	Dauer	Größe
150	5	I	L	E	4.366	3.221	579	70
137	5	-	-	E	15.484	10.355	220	69
106	5	I	-	E	66.934	17.595	486	65
82	5	-	-	E	39.195	29.842	144	67
15	5	-	L	E	46.666	43.029	207	67
44	5	I	L	E	70.626	54.442	349	72
146	5	-	L	E	$> 10^6$	115.198	459	67
7	5	I	L	-	165.330	165.330	203	67
35	5	-	L	-	472.560	303.313	187	69
125	5	I	L	E	363.341	348.317	532	67
22	5	I	L	E	402.757	360.147	710	68

Nr	Bilder	I	L	E	FehlerVO	Fehler	Dauer	Größe
148	5	-	L	-	$> 10^6$	451.862	242	68
87	5	-	L	E	$> 10^6$	479.421	326	69
93	5	I	-	E	594.559	486.228	408	67
5	5	-	-	E	$> 10^6$	547.403	162	67
39	5	I	L	E	$> 10^6$	$> 10^6$	507	67

B.5 Modelle aus 6 verwendeten Bildern

Nr	Bilder	I	L	E	FehlerVO	Fehler	Dauer	Größe
43	6	-	-	E	32.666	15.954	357	74
76	6	-	L	E	225.625	18.193	352	80
149	6	-	L	-	19.293	19.293	165	79
9	6	-	L	-	58.081	24.518	357	79
110	6	-	L	E	36.615	34.474	416	73
18	6	-	L	E	135.499	34.983	354	79
113	6	-	L	E	188.255	69.286	599	78
50	6	I	L	E	95.493	92.170	693	82
84	6	I	-	E	166.491	155.432	797	78
25	6	-	L	E	462.040	186.021	542	79
38	6	I	L	E	397.117	279.895	947	76
30	6	-	L	E	684.651	280.839	238	81
109	6	I	-	E	317.954	297.180	434	84
99	6	I	L	E	544.892	411.152	419	88
68	6	I	L	E	469.524	421.134	951	79
67	6	-	-	E	$> 10^6$	431.205	300	75
63	6	-	L	E	788.434	486.328	310	77
112	6	-	L	E	$> 10^6$	488.385	402	75
83	6	I	L	E	516.395	513.238	574	81
145	6	I	-	E	584.246	561.721	455	79
98	6	-	-	E	$> 10^6$	565.939	393	76
72	6	-	L	E	$> 10^6$	570.628	307	77
147	6	-	-	E	$> 10^6$	697.227	262	72
59	6	-	L	E	$> 10^6$	704.931	255	76
103	6	-	L	-	$> 10^6$	$> 10^6$	109	80
28	6	-	L	E	$> 10^6$	$> 10^6$	205	80
11	6	-	L	-	$> 10^6$	$> 10^6$	224	77
92	6	-	L	-	$> 10^6$	$> 10^6$	277	76
16	6	I	L	E	$> 10^6$	$> 10^6$	874	78
6	6	I	L	-	$> 10^6$	$> 10^6$	193	77
114	6	-	L	-	$> 10^6$	$> 10^6$	191	79
23	6	-	L	-	$> 10^6$	$> 10^6$	167	78
21	6	I	-	E	$> 10^6$	$> 10^6$	212	85

B.6 Modelle aus 7 verwendeten Bildern

Nr	Bilder	I	L	E	FehlerVO	Fehler	Dauer	Größe
123	7	I	L	E	129.098	126.724	1.118	101
32	7	I	-	E	141.166	140.348	917	101
153	7	I	L	E	283.026	211.141	924	98
127	7	I	L	E	392.104	229.764	733	103
58	7	-	-	E	331.200	262.781	349	109
138	7	I	L	E	328.120	314.155	1.091	98
89	7	I	L	E	321.816	316.369	1.444	99
140	7	I	L	-	326.992	326.988	517	97
104	7	-	L	-	378.554	378.554	249	100
2	7	I	-	E	740.172	435.399	517	97
152	7	I	L	E	590.673	450.629	1.701	101
20	7	-	L	E	871.401	518.499	239	97
34	7	-	L	E	636.638	522.968	168	101
118	7	-	L	-	576.817	576.817	289	97
105	7	-	L	E	$> 10^6$	605.432	688	100
46	7	-	L	E	$> 10^6$	620.368	345	99
102	7	I	L	E	971.612	740.634	1.477	104
19	7	-	-	E	$> 10^6$	747.131	917	98
134	7	-	L	-	$> 10^6$	$> 10^6$	315	99
91	7	I	L	-	$> 10^6$	$> 10^6$	262	99
10	7	-	-	E	$> 10^6$	$> 10^6$	289	99
45	7	-	L	-	$> 10^6$	$> 10^6$	138	97
42	7	I	L	-	$> 10^6$	$> 10^6$	325	97
135	7	I	L	-	$> 10^6$	$> 10^6$	381	99
129	7	I	L	E	$> 10^6$	$> 10^6$	462	106
75	7	-	-	E	$> 10^6$	$> 10^6$	374	99
79	7	I	L	-	$> 10^6$	$> 10^6$	325	107
51	7	-	L	-	$> 10^6$	$> 10^6$	195	102
121	7	I	L	E	$> 10^6$	$> 10^6$	549	106
27	7	I	L	E	$> 10^6$	$> 10^6$	349	110
94	7	-	L	E	$> 10^6$	$> 10^6$	500	95
8	7	-	L	E	$> 10^6$	$> 10^6$	311	98
86	7	I	L	-	$> 10^6$	$> 10^6$	241	101
101	7	-	L	-	$> 10^6$	$> 10^6$	174	101
144	7	-	L	E	$> 10^6$	$> 10^6$	336	100

B.7 Modelle aus 8 verwendeten Bildern

Nr	Bilder	I	L	E	FehlerVO	Fehler	Dauer	Größe
115	8	I	L	E	78.721	71.394	1.153	116
70	8	-	-	E	399.181	282.000	644	114
36	8	I	L	E	346.453	324.220	1.067	117
17	8	I	L	-	343.467	343.467	375	121
48	8	I	L	E	379.697	358.553	1.022	116
85	8	-	L	-	434.004	434.004	245	115
26	8	I	-	E	498.973	494.713	738	118

Nr	Bilder	I	L	E	FehlerVO	Fehler	Dauer	Größe
66	8	-	-	E	$> 10^6$	575.533	492	115
14	8	I	L	E	732.834	690.650	1631	116
88	8	-	-	E	$> 10^6$	710.912	438	116
132	8	I	L	-	729.732	729.732	316	116
37	8	I	L	E	837.754	784.895	744	124
122	8	I	L	E	$> 10^6$	850.285	1.174	117
60	8	-	L	E	$> 10^6$	857.025	368	116
117	8	-	-	E	$> 10^6$	868.028	686	122
130	8	-	L	E	$> 10^6$	903.489	569	116
78	8	-	-	E	$> 10^6$	938.400	540	120
96	8	-	L	-	$> 10^6$	$> 10^6$	243	118
62	8	I	L	E	$> 10^6$	$> 10^6$	1.690	119
56	8	-	L	E	$> 10^6$	$> 10^6$	267	123
100	8	I	L	E	$> 10^6$	$> 10^6$	1.218	125
133	8	I	L	-	$> 10^6$	$> 10^6$	322	118
74	8	I	L	E	$> 10^6$	$> 10^6$	2.012	113
55	8	I	L	E	$> 10^6$	$> 10^6$	658	125
65	8	-	-	E	$> 10^6$	$> 10^6$	602	120
57	8	-	L	E	$> 10^6$	$> 10^6$	918	113
142	8	-	L	-	$> 10^6$	$> 10^6$	227	116
116	8	-	-	E	$> 10^6$	$> 10^6$	220	116

Abbildungsverzeichnis

Literaturverzeichnis

[Bee] Julian Beever. Julian Beever's pavement drawings. http://users.skynet.be/J.Beever/pave.htm.

[BR92] Albrecht Beutelspacher and Ute Rosenbaum. *Projektive Geometrie*. Vieweg, 1992.

[Bre03] Fa. Breuckmann. MPT - miniature projection-technique. http://www.breuckmann.com/flash/messverfahren.pdf, August 2003.

[FB81] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, 1981.

[FCZ98] Andrew W. Fitzgibbon, Geoff Cross, and Andrew Zisserman. Automatic 3d model construction for turn-table sequences. In *SMILE*, pages 155–170, 1998.

[FHD02] G. Finlayson, S. Hordley, and M. Drew. Removing shadows from images using retinex. In *10th Color Imaging Conference: Color Science and Engineering Systems, Technologies, Applications, Scottsdale, Arizona, USA*, pages 73–79, 2002.

[FP03] David A. Forsyth and Jean Ponce. *Computer Vision. A Modern Approach.* Pearson Education International, 2003.

[FZ98a] Andrew Fitzgibbon and Andrew Zisserman. Automatic 3D model acquisition and generation of new images from video sequences. In *Proceedings of European Signal Processing Conference (EUSIPCO '98), Rhodes, Greece*, pages 1261–1269, 1998.

[FZ98b] Andrew W. Fitzgibbon and Andrew Zisserman. Automatic camera recovery for closed or open image sequences. In *ECCV (1)*, pages 311–326, 1998.

[GL89] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, second edition, 1989.

[Gol02] E. Bruce Goldstein. *Sensation and Perception*. Wadsworth, 2002.

[Gro01] Benedikt Großer. *Ein paralleler und hochgenauer Algorithmus für die bidiagonale Singulärwertzerlegung*. PhD thesis, Bergische Universität Wuppertal, 2001.

[HS88] Chris Harris and Mike Stephens. A combined corner and edge detector. In *4th ALVEY Vision Conference*, pages 147–151, 1988.

[HZ04] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004.

[Ill97] IllusionWorks. Ames room. http://psylux.psych.tu-dresden.de/i1/kaw/diverses Material/www.illusionworks.com/ html/ames_ room.html, 1997.

[Jan03] Thomas Jansen. Evolutionäre Algorithmen. http://ls2-www.cs.uni-dortmund.de/lehre/sommer2003/evo-alg/evoAlg-2003.pdf, 2003.

[MvdM96] Thomas Maurer and Christoph von der Malsburg. Tracking and learning graphs on image sequences of faces. In C. v.d. Malsburg, W. v. Seelen, J.C. Vorbrüggen, and B. Sendhoff, editors, *Proceedings of the ICANN 1996*, pages 323–328, Bochum, July 1996.

[Pet02] Gabriele Peters. *A View-Based Approach To Three-Dimensional Object Perception*. Shaker Verlag, 2002.

[PTVF88] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C*. Cambride University Press, 1988.

[RPEvdM99] Michael Rinne, Michael Pötzsch, Christian Eckes, and Christoph von der Malsburg. Designing objects for computer vision: The backbone of the library FLAVOR, December 1999.

[Sch04] Peer Schmidt. *Ein modulares Robotersystem zum Ertasten und Manipulieren von Objekten*. PhD thesis, Universität Bielefeld, 2004.

[SCMS01] G. Slabaugh, B. Culbertson, T. Malzbender, and R. Schafer. A survey of methods for volumetric scene reconstruction from photographs. In *International Workshop on Volume Graphics*, pages 81–100, 2001.

[Ste02] Kenneth M. Steele. Ames room diagram. http://www.acs.appstate.edu/kms/classes/psy3203/Depth/ AmesDiagram.htm, april 2002.

[TMHF00] Bill Triggs, Philip McLauchlan, Richard Hartley, and Andrew Fitzgibbon. Bundle adjustment – a modern synthesis. In W. Triggs, A. Zisserman, and R. Szeliski, editors, *Vision Algorithms: Theory and Practice*, LNCS, pages 298–375. Springer Verlag, 2000.

[Wie01] Jan Wieghardt. *Learning the Topology of Views: From Images to Objects*. PhD thesis, Ruhr-Universität Bochum, 2001.

[WL97] Rolf Würtz and Tino Lourens. Corner detection in color images by multiscale combination of endstopped cortical cells. In Wulfram Gerstner, Alain Germond, Martin Hasler, and Jean-Daniel Nicoud, editors, *Artificial Neural Networks - ICANN '97*, volume 1327 of *Lecture Notes in Computer Science*, pages 901–906. Springer Verlag, 1997.

[ZG96] Philip G. Zimbardo and Richard J. Gerrig. *Psychologie*. Springer, 7th edition, 1996.