Markerlose Selbstlokalisation durch Fusion von Sensordaten

Diplomarbeit im Fach Computervisualistik

vorgelegt
von

Timo Dickscheid

Geboren am 12.11.1975 in Koblenz

Angefertigt am

Institut für Computervisualistik
Arbeitsgruppe Aktives Sehen
Universität Koblenz–Landau

Betreuer: Prof. Dr.-Ing. Dietrich Paulus, Dr.-Ing. Chunrong Yuan

Beginn der Arbeit: 01.04.2006

Abgabe der Arbeit: 30.09.2006

Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Die Richtlinien der Arbeitsgruppe für Studien- und Diplomarbeiten habe ich gelesen und anerkannt, insbesondere die Regelung des Nutzungsrechts.

Koblenz, den 28. September 2006

Timo Dickscheid

Inhaltsverzeichnis

1 Einleitung **5**

2 Techniken zur markerlosen Selbstlokalisation **9**

2.1 Namen und Bezeichnungen 10

2.2 Interessenspunkte 12

2.3 Rekonstruktion der Kameraposition 14

2.4 Initialisierung natürlicher Marker 20

2.5 Fusion von Bild- und Inertialsensordaten 23

3 Entwicklung eines Gesamtkonzepts **25**

3.1 Zielsetzungen 25

3.2 Lösungsansätze zur Rekonstruktion 26

3.3 Konzeption des Systems 31

3.4 Verarbeitung der Sensordaten 33

3.5 Grundschema der Erweiterten Kalman Filter 42

3.6 Erzeugung einer neuen Karte 44

3.7 Schätzung der Kameraposition 57

3.8 Reinitialisierung natürlicher Marker 65

4 Objektorientiertes Design und Realisierung 67

4.1 Klassenmodell 68

4.2 Sensordatenerfassung 68

4.3 Punktdetektoren 71

4.4 Rekonstruktion 75

4.5 Simulatoren 78

5 Experimente und Ergebnisse 81

5.1 Abgleich von Punktmerkmalen 81

5.2 Initialisierung von 3D-Markern 84

5.3 Schätzung der Kamerabewegung 91

6 Zusammenfassung 95

A Symbole und Abkürzungen 99

A.1 Verwendete Symbole 100

A.2 Abkürzungsverzeichnis 102

B Berechnungen 103

B.1 Ableitung der Messwertgleichung aus Abschnitt 3.6.4 104

B.2 Ableitung der Messwertgleichung aus Abschnitt 3.7.1 106

C Bilder und grafische Auswertungen 107

D Inhalt der beiliegenden CD-ROM 127

Kapitel 1

Einleitung

Diese Diplomarbeit ist in den Kontext des *Augmented-Reality*-Projektes *IPCity* eingebettet, das unter anderem am Fraunhofer Institut für angewandte Informationstechnik in St. Augustin entwickelt wird. In der geplanten Anwendung wird sich eine Person frei in einer Stadt bewegen und dabei ein *Head-Worn Display* tragen, an dem eine Kompaktkamera sowie ein Inertialsensor des Typs *Xsens MTx* befestigt ist. Die beiden Sensoren sind mit einem Rechner verbunden, den der Anwender ähnlich einem Rucksack auf dem Rücken trägt. Im Rahmen dieser Diplomarbeit soll ein Konzept erarbeitet werden, um aus den beiden Sensordatenströmen online die jeweils aktuelle Position der Kamera sowie deren Bewegung zu rekonstruieren. Das Wissen über die Kameraposition ist wichtig, um den Bilddatenstrom durch Einblenden von Informationen in der Szene zu augmentieren. Die grundlegende Idee zeigt Bild 1.1.

Der Inertialsensor liefert laut Herstellerangaben zuverlässige Referenzdaten zur eigenen 3D-Orientierung. Die internen Parameter der Kamera sind konstant und werden offline kalibriert. Die Aufgabe besteht also darin, die extrinsischen Kameraparameter bei gegebenen intrinsischen Kameraparametern aus den Sensordatenströmen zu schätzen. Die Rekonstruktion der Kameraposition aus Sensordaten wird oft als 3D Selbstlokalisation (*3D Pose Estimation*) bezeichnet und kann dem etwas breiter gefassten Begriff der Trackingverfahren zugeordnet werden. Der Begriff “Selbstlokalisation” bringt zusätzlich zum Ausdruck, das keine Objekte in der Umgebung detektiert und semantisch bewertet werden. Dennoch

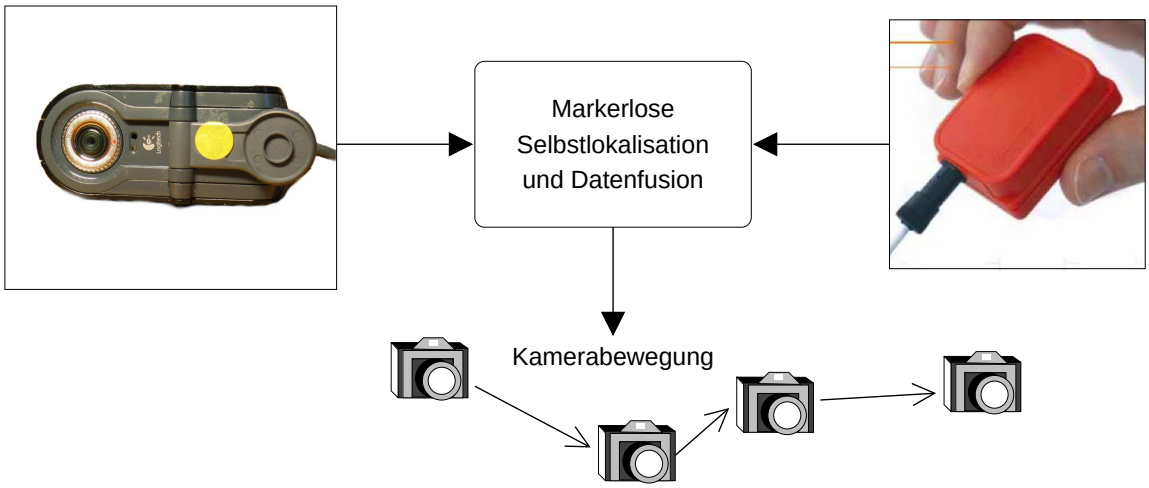


Bild 1.1: Veranschaulichung der grundlegenden Problemstellung: Aus Sensordaten, die von einer Kamera und einem Inertialsensor geliefert werden, wird mittels Datenfusion und ohne Einsatz von künstlichen Markern die Kamerabewegung rekonstruiert (Bildausschnitt oben rechts aus *Xsens MTx* Datenblatt, `http://www.xsens.com`, 17.09.2006).

sind die Techniken direkt miteinander verwandt, da in beiden Fällen die relative Struktur von Kamera und Objekten der Szene im dreidimensionalen Raum rekonstruiert wird.

Bedarf an Trackingverfahren besteht heute neben der *Augmented Reality (AR)* vorwiegend in den Forschungsfeldern Robotik, *Motion Tracking* und *Wearable Computing*. Insbesondere in der Robotik verwendete man zur Selbstlokalisation in der Vergangenheit anstelle von Kameras häufig andere Sensoren wie GPS-Geräte, WLAN-Empfänger, Infrarotkameras oder Radargeräte. Der Einsatz von Kameras zur Positionsbestimmung ist jedoch attraktiv, da in den meisten Anwendungsfällen ohnehin Bilddaten zur Verfügung stehen, und Kameras relativ preisgünstig sind. Darüber hinaus ist der Informationsgehalt im Bilddatenstrom sehr hoch und ermöglicht prinzipiell sehr umfangreiche Auswertungen.

Die ersten bildbasierten Trackingsysteme bedienten sich sogenannter “Marker” zur Detektion von Objekten in der Szene. Bei diesen Markern handelt es sich um leicht segmentierbare Gegenstände wie etwa Aufkleber mit markanten Mustern, die vor dem Einsatz des Systems in der Szene oder an den Akteuren angebracht werden. Als Beispiel seien

das proprietäre System *ViconPeak* der Firma *Vicon*[1] sowie die als Open Source Projekt verfügbare Entwicklungsumgebung *ARToolKit*[2] genannt. Solche Systeme finden bereits breite Anwendung in Industrie und Forschung.

Der Aufwand für die Präparation der Einsatzumgebung mit Markern ist teilweise akzeptabel (man denke etwa an die entsprechende einmalige Präparation einer Produktionsstrasse), in vielen Bereichen jedoch ein Hindernis für den alltäglichen Einsatz der Trackingsysteme. Insbesondere bei AR-Anwendungen im Outdoor-Bereich ist es nicht praktikabel, Objekte in der Umgebung vorher mit Markern auszustatten. Vielmehr sollte das System nach einer kurzen Selbstinitialisierung schnell einsatzbereit sein. Hierzu ist es erforderlich, dass natürliche Merkmale der Umgebung zur Rekonstruktion herangezogen werden. Verfahren, die diesen Ansatz verfolgen, bezeichnet man als *markerlose Trackingverfahren*. In der Robotik wird dabei häufig auch eine Karte des umgebenden Raums erstellt; man spricht dann von *Simultaneous Localisation and Mapping (SLAM)*.

Die Detektion und Zuordnung natürlicher Merkmale in Bildern ist rechenaufwändig und insbesondere unter Echtzeitbedingungen häufig fehlerbehaftet. Es existieren zwar akzeptable Lösungen für rein bildbasiertes markerloses Tracking, jedoch stellen diese in der Regel spezielle Anforderungen an den Einsatzbereich. So liefern die meisten heute bekannten Verfahren nur in Innenräumen brauchbare Ergebnisse. Stabile SLAM-Techniken profitieren wie viele Verfahren aus der Fahrzeugentwicklung von dem Zusatzwissen, das die Motorik über die Kamerabewegung liefern kann. Zur Realisierung echtzeitfähiger, markerloser Trackingsysteme für den mobilen Outdoor-Einsatz an einer Person ist es daher möglicherweise sinnvoll, zusätzliche, performante Sensoren zu verwenden, deren Daten mit wenig Rechenaufwand verwertbar sind. Die Fusion solcher Sensordaten mit der bildbasierten Analyse kann vielleicht die Performanz und Robustheit erhöhen. Aus diesem Grund wird der Einsatz des Inertialsensors mit vorgesehen. Da es sich im Vergleich zur Kamera jedoch um ein teures Gerät handelt, wird dessen Verwendung gleichzeitig kritisch betrachtet.

Im folgenden Kapitel 2 wird zunächst ein Überblick über einige der heute bekannten Techniken zur Erzeugung natürlicher Marker sowie zur Rekonstruktion der Kamerabewegung

[1] `www.vicon.com`
[2] `http://www.hitl.washington.edu/artoolkit/`

aus Sensordatenströmen gegeben. Darauf folgt in Kapitel 3 die Entwicklung eines Gesamtkonzepts für ein solches System unter Berücksichtigung der speziellen Aufgabenstellung. Die dort erarbeiteten Lösungsansätze werden als Softwarebausteine in einer C++-Klassenbibliothek implementiert, deren Design und Anwendung in Kapitel 4 ausführlich besprochen wird. Die wichtigsten Bestandteile des Systems werden in Form von Experimenten und Simulationen evaluiert. Eine Beschreibung und Ergebnisse dieser Experimente enthält Kapitel 5. In Kapitel 6 werden schließlich die Ergebnisse zusammengefasst und Anregungen für künftige Arbeiten gegeben.

Kapitel 2

Techniken zur markerlosen Selbstlokalisation

Bei der Betrachtung markerloser Rekonstruktionsverfahren, deren Kern eine bildbasierte Herangehensweise bildet, lassen sich die folgenden Komponenten unterscheiden:

- Die Art der als natürliche Marker eingesetzten Bildmerkmale
- Der Ansatz zur Lösung des Rekonstruktionsproblems
- Der Umfang des in das Verfahren einfließenden Vorwissens
- Die Verwendung zusätzlicher Sensordaten

Dieses Kapitel stellt für jede dieser Komponenten einige Ansätze der aktuellen Wissenschaft vor. Im folgenden Abschnitt werden jedoch zunächst einige zentrale Begriffe und Sachverhalte erläutert, auf die in der Arbeit vielfach Bezug genommen wird.

2.1 Namen und Bezeichnungen

2.1.1 Extrinsische Kameraparameter

Die *extrinsischen Kameraparameter* bestimmen die Position einer Kamera relativ zu einem beliebigen, aber definierten Weltkoordinatensystem. Sie werden als 4×4-Matrix der Form

$$\begin{pmatrix} \boldsymbol{R} & \boldsymbol{t} \\ 000 & 1 \end{pmatrix}$$

angegeben, mit der ein Punkt im Weltkoordinatensystem von links multipliziert wird, um den entsprechenden Punkt im Koordinatensystem dieser Kamera zu erhalten. Dabei enthält $\boldsymbol{R}$ den Rotations-, und $\boldsymbol{t}$ den Translationsanteil der Transformation des Weltpunktes. In dieser Arbeit werden durchgängig die Begriffe *Rotation* ($\boldsymbol{R}$) und *Translation* ($\boldsymbol{t}$) verwendet. In der Computergrafik ist außerdem der Begriff *Modelview-Matrix* häufig zu finden. [1] Die einzelnen Komponenten in $\boldsymbol{R}$ werden mit r_{ij} abgekürzt, d. h.

$$\boldsymbol{R} = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \quad .$$

In dieser Arbeit wird gleichzeitig von der *Position* und *Orientierung* einer Kamera gesprochen werden, die ebenfalls durch eine 3×3-Rotationsmatrix und einen Verschiebungsvektor darstellbar sind. Die Bedeutung dieser beiden Transformationen ist jedoch nicht mit der Rotation und Translation identisch, denn sie drücken die äußere Orientierung der Kamera selbst im Weltkoordinatensystem aus und werden nicht auf Weltpunkte angewendet. Die Position der Kamera wird im Folgenden mit $\boldsymbol{r} = \begin{pmatrix} r_x & r_y & r_z \end{pmatrix}$ und die Orientierung mit $\boldsymbol{O}$ abgekürzt. Die Transformationen sind direkt ineinander überführbar; es gilt

$$\boldsymbol{O} = \boldsymbol{R}^T \qquad \text{und} \tag{2.1}$$

$$\boldsymbol{r} = -\boldsymbol{R}^T\boldsymbol{t} = -\boldsymbol{O}\boldsymbol{t} \quad . \tag{2.2}$$

[1] Die Modelview-Matrix ist jedoch meist anders formatiert als in der hier gewählten Darstellung.

2.1.2 Koordinatensysteme bei der Projektion

Die Projektion eines 3D-Punktes der Szene in ein digitales Bild wird durch die folgenden Multiplikationen abgebildet:

$$\begin{pmatrix} p_x^P \\ p_y^P \\ p_z^P \end{pmatrix} = \begin{pmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} p_x^C \\ p_y^C \\ p_z^C \\ 1 \end{pmatrix} \tag{2.3}$$

$$= \begin{pmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{R} & \boldsymbol{t} \\ 000 & 1 \end{pmatrix} \begin{pmatrix} p_x^W \\ p_y^W \\ p_z^W \\ 1 \end{pmatrix} \tag{2.4}$$

Es wird keine Radial- oder Tangentialverzerrung der Kamera modelliert und eine korrekte rechtwinklige Form der lichtempfindlichen Sensoren auf dem Kamerachip angenommen. Die tatsächliche Bildposition - also der Index von Zeile und Spalte im Bild - ergibt sich nach Division durch die z-Komponente:

$$\begin{pmatrix} \mu \\ \nu \end{pmatrix} = \begin{pmatrix} p_x^P / p_z^P \\ p_y^P / p_z^P \end{pmatrix} \tag{2.5}$$

In dieser Arbeit wird mehrfach von Punkten und Vektoren die Rede sein, die im Welt-, Kamera- und Bildkoordinatensystem beschrieben werden müssen. Dazu werden die entsprechenden Bezeichner wie in obiger Projektionsgleichung durch einen nachfolgenden hochgestellten Buchstaben versehen, der das entsprechende Koordinatensystem andeutet. W steht dabei für das Welt-, C für das Kamera- und P für das Bildkoordinatensystem.

2.1.3 Markerlose Selbstlokalisation

In der aktuellen Literatur werden Verfahren, die ohne die Installation künstlicher Marker 3D-Struktur aus Bildern berechnen, als “markerlose” Verfahren bezeichnet. Ein Teil dieser Verfahren verwendet bekannte Modelle von Objekten, die schließlich in der Szene detektiert werden, um die Struktur zu bestimmen, ein anderer Teil - so auch das in dieser Arbeit

zum Einsatz kommende Verfahren - sogenannte “natürliche Marker”, die nicht manuell in der Szene installiert werden und oftmals erst zur Laufzeit bestimmt werden. In diesem Fall ist das Attribut “markerlos” möglicherweise irreführend, da es sich im engeren Sinne auch um Marker handelt. Dennoch wird die übliche Bezeichnung auch hier verwendet, um mit der bekannten Literatur konform zu bleiben.

Da kein Modell für die Rekonstruktion zum Einsatz kommt, handelt es sich in dieser Arbeit um ein Verfahren der Selbstlokalisation. Oft wird in der Literatur auch der Begriff “Trackingverfahren” benutzt, da eine Verfolgung natürlicher Marker im Bilddatenstrom den Kern bildet. In dieser Arbeit werden die Begriffe “Selbstlokalisation”, “Trackingverfahren” und “Rekonstruktion” ebenfalls parallel benutzt und sind nicht als Abgrenzung verschiedener Techniken zu verstehen. Wenn von dem “Rekonstruktionsproblem” gesprochen wird, ist der Teil des Verfahrens gemeint, der sich direkt mit der konkreten Berechnung der unbekannten extrinsischen Kameraparameter beschäftigt.

2.2 Interessenspunkte

Erste Verfahren zum markerlosen Tracking wie *RAPiD* [Har92] basierten auf der Ermittlung von Kanten mittels starker Gradienten im Bild. Solche Merkmale lassen sich gut segmentieren, sind aber in manchen Umgebungen nicht in genügender Anzahl zu finden. Darüber hinaus ist die genaue Lokalisation von Kanten schwierig, wenn sich die Beleuchtungseigenschaften in der Szene ändern oder die Kantenenden nicht im Blickfeld der Kamera liegen (“*aperture problem*”, [TV98, Kap. 8.3.2]).

Unter der Annahme, dass die Helligkeit der Bilder im Mittel konstant bleibt, lässt sich die Position eines Punktes über eine Folge von Bildern als scheinbare Bewegung verfolgen (“Optischer Fluss”, [TV98, Kap. 8]). Aus solchen Bewegungen kann man Vektorfelder erzeugen, mittels derer markerlose Trackingverfahren implementiert werden können. Ein Nachteil dieser Verfahren liegt darin, dass sich die Fehler bei der Schätzung über längere Zeiträume stark akkumulieren, die Folge ist *Drift*.[2]

[2] “Drift” beschreibt Fehler in der Schätzung, die sich mit fortschreitendem Auslesen der Messwerte kumulieren.

Lucas und Kanade entwickelten den sogenannten “KLT-Tracker” [KL81], der eine Nachbarschaft von Bildpunkten unter verschiedenen Deformationen in aufeinander folgenden Bildern wiederfindet und lokalisiert. Solche Verfahren, die man allgemein als *Template Matching* kategorisiert, sind sehr stark verbreitet und wurden in vielen Varianten weiterentwickelt.

Heute liegt der Fokus in der Forschung stark auf der Verfolgung sogenannter Interessenspunkte (*Interest Points*). Damit bezeichnet man automatisch ermittelte, markante Punktmerkmale in Bildern. Oft enthalten diese Merkmale keine konkrete Semantik, können aber anhand einer Beschreibung (dem sogenannten “Deskriptor”) wiedergefunden und zugeordnet werden. Das klassische Beispiel für Interessenspunkte sind die *“Harris Corners”*, die auf horizontalen und vertikalen Ableitungen in der Nachbarschaft eines Bildpunktes basieren [HS88]. Sie stellen eine Erweiterung der Moravec Punktmerkmale dar [Mor81]. Diese Merkmale finden Einsatz in vielen Verfahren der Bildverarbeitung, insbesondere in frühen Verfahren zur Punktverfolgung in Bildern. Ein weiteres Verfahren zum Auffinden solcher Punktmerkmale wurde 1991 von Carlo Tomasi und Takeo Kanade entwickelt [TK92] und erkennt insbesondere auch Regionen mit schwächeren eckförmigen Strukturen der Muster im Bild. Der Algorithmus basiert auf dem Verhältnis von charakteristischen Eigenwerten einer Matrix aus Gradienten in der Umgebung des Bildpunktes und ist unter anderem recht stabil gegenüber Rauschartefakten.

David Lowe stellte 2004 ein Verfahren zur Ermittlung rotations- und skalierungsinvarianter Merkmale vor [Low04], die *Scale Invariant Feature Transform (SIFT)*. Die SIFT-Merkmale basieren auf Extremstellen im differentiellen Skalenraum eines Bildes, die subpixelgenau lokalisiert und durch einen Deskriptor beschrieben werden, der aus Gradientenhistogrammen einer lokalen Nachbarschaft der Extremstellen erzeugt wird. Sie lassen sich unter einer Vielzahl verschiedener Transformationen zuordnen und wurden bereits in viele Lösungen zur Objekterkennung integriert. Für die Objektverfolgung und Selbstlokalisation im Offline-Einsatz eignen sich die Merkmale hervorragend. Für die Echtzeitverarbeitung stellt sich das Problem, das die Extraktion der SIFT-Merkmale äußerst rechenintensiv ist. Der Rechenaufwand lässt sich einschränken, indem man die Merkmale beispielsweise nur für die (Re-) Initialisierung des Trackingsystems benutzt oder den Suchraum mittels einfacherer Verfahren stark eingrenzt. Kürzlich wurde eine performante

Approximation des Verfahrens vorgeschlagen [GGB06], die unter anderem keine Subpixellokalisierung vornimmt und den Skalenraum, der im Originalverfahren maßgeblich durch die Faltung mit einem Gaussfilter gebildet wird, mit einem Mittelwertfilter annähert.

Um Unabhängigkeit gegenüber bestimmten Transformationen zu erreichen, bedient man sich auch sogenannter “Invarianten”. Dabei handelt es sich um spezifische Eigenschaften von Bildinhalten, die sich unter einer Klasse von Transformationen nicht ändern. So ist beispielsweise die Länge einer Kante eine Invariante in Bezug auf die 2D-Rotation eines Bildes. In [TV98] wird das Doppelverhältnis von Linien im projektiven Raum als Invariante gegenüber projektiven Transformationen genannt. Der Nachteil von Invarianten ist, dass sie die in Frage kommenden Merkmale sehr stark einschränken und insbesondere die Eindeutigkeit und Unterscheidbarkeit erschweren.

Ein weiterer interessanter Ansatz wurde 1994 von Shi und Tomasi [ST94] beschrieben. Die Idee besteht darin, nicht die Merkmale selbst, sondern ihre Eignung im Hinblick auf den Verfolgungsalgorithmus zu optimieren. Das Verfahren wählt genau die Merkmale aus, welche die genaueste Verfolgung erlauben. Es ist invariant gegenüber affinen Transformationen.

2.3 Rekonstruktion der Kameraposition

2.3.1 Algebraische Verfahren

Unter dem Begriff der algebraischen Verfahren sollen hier solche Methoden der Rekonstruktion zusammengefasst werden, die auf Basis von Punktkorrespondenzen in Bildern versuchen, die zu Grunde liegende 3D-Struktur durch algebraische Lösung eines Gleichungssystems zu ermitteln. Ein solches Verfahren ist die Triangulierung, bei der Parameter der Epipolargeometrie in Form der fundamentalen Matrix oder eines Multifokaltensors aufgrund von Punktkorrespondenzen als optimale Lösung eines Gleichungssystems ermittelt werden. Eine weitere Variante sind Faktorisierungsmethoden, bei denen Punktkorrespondenzen in eine Messwertmatrix einfließen, aus der durch Aufspaltung in Teilmatrizen die Strukturinformation gewonnen wird. Mitunter spricht man hier auch von “*bottom-up*”-

oder “2D-3D”-Methoden, weil die Problemstellung maßgeblich im zweidimensionalen Raum formuliert wird.

Eine grundsätzlich ideale, aber sehr komplexe algebraische Lösung stellt der *globale Bündelausgleich* dar [HZ03, Kap. 18.1]. Dabei betrachtet man eine Menge von Punktkorrespondenzen über die gesamte Bildfolge, die jeweils Projektionen der gleichen 3D-Punkte aus der Szene sind. Daraus wird iterativ eine Maximum Likelihood Schätzung aller Kameramatrizen durch Minimierung des quadratischen Fehlers bei der Rückprojektion der Punkte bestimmt. Für die Rekonstruktion mit globalem Bündelausgleich ist keinerlei Vorwissen nötig. Für den Erfolg des Verfahrens ist aber eine sehr gute Initialisierung und eine hohe Anzahl von Bildern erforderlich. Das macht den Einsatz für die Echtzeit-Schätzung schwierig, da man dort nur eine begrenzte Zahl aufeinander folgender Bilder gleichzeitig betrachten, also nur einen *lokalen* Bündelausgleich vornehmen kann. Darüber hinaus ist das implizierte Optimierungsproblem sehr komplex.

Carlo Tomasi und Takeo Kanade haben 1992 ein Faktorisierungsverfahren vorgeschlagen, um Rotation und Translation einer Kamera aus einem Bildstrom zu rekonstruieren, ohne die Tiefeninformation in einem Zwischenschritt auszurechnen [TK92]. Das Verfahren setzt voraus, dass die Bilder unter orthographischer Projektion erzeugt werden. Basis ist eine Messwertmatrix, deren Spalten jeweils die Trajektorie eines Punktes über die Menge aller Bilder enthalten. Da die Matrix durch eine Rang-3-Zerlegung faktorisiert werden kann, müssen alle Trajektorien in einem dreidimensionalen Unterraum liegen. Tomasi und Kanade zeigen, dass mittels Singulärwertzerlegung eine direkte Faktorisierung der Messwertmatrix möglich ist, die nach einem Optimierungsschritt die Kamerabewegung liefert. Es steht also für den Fall orthographischer Projektion eine *“closed form”*-Lösung für eine Maximum Likelihood Schätzung zur Verfügung, die auch in Echtzeit einsetzbar ist. Neben dem eingeschränkten Projektionsmodell hat dieses Verfahren den Nachteil, dass die verwendeten Punktmerkmale über alle Bilder der Folge hinweg sichtbar sein müssen. Darüber hinaus bringt die Konstruktion der Messwertmatrix mit sich, dass Bewegungen der Kamera in Richtung der optischen Achse nicht rekonstruiert werden können.

Eine Erweiterung des Verfahrens von Tomasi und Kanade auf den Fall perspektivischer Projektion wurde 1996 von Peter Sturm und Bill Triggs vorgestellt [ST96]. Auch hier wird eine Messwertmatrix konstruiert, welche die Punktmerkmale über alle Bilder der Bildfol-

ge hinweg enthält. Nach Neuskalierung der Punktkoordinaten erhält man eine Rang-4-Matrix, aus der sich mit Hilfe der Singulärwertzerlegung die Projektionsdaten sowie die Kamerabewegung abspalten lassen. Für die Neuskalierung ermitteln Sturm und Triggs geeignete Faktoren aus einer Schätzung der fundamentalen Matrix nach dem 8-Punkte-Algorithmus [Har97], die eine Größenordnung für die projektive Tiefe der Punkte angeben. Auch hier besteht die Einschränkung, dass die Punktmerkmale in allen Bildern gleichzeitig sichtbar sein müssen. Verschwinden Punktmerkmale in einzelnen Bildern, so entstehen Lücken in der Messwertmatrix, und das Verfahren lässt sich nicht mehr ohne Weiteres anwenden.

Die Approximation von Messwertmatrizen durch Matrizen geringeren Rangs stellt also ein Problem dar: Während es sich für für vollständige Matrizen leicht durch Singulärwertzerlegung lösen lässt, scheitern viele Verfahren, wenn die Matrizen Lücken aufweisen. Richard Hartley beschäftigte sich mit dieser Problematik und stellte ein Verfahren namens *"PowerFactorization"* vor, mit dem unvollständige und fehlerhafte Messwertmatrizen durch eine Matrix geringeren Rangs approximiert werden können [HS03, KH05]. Damit lässt sich die harte Bedingung lückenloser Trajektorien, die für die oben genannten Faktorisierungsmethoden gilt, auflösen.

Ein Triangulierungsverfahren zur Ermittlung der Eigenbewegung eines Roboters wird von David Nistér in [NNB04] beschrieben. Hierbei wird in einem Initialisierungsschritt die relative Kameraposition zwischen drei Bildern mit dem 5-Punkte-Algorithmus geschätzt. Aus den resultierenden Merkmaltrajektorien bestimmt man die 3D-Position der Punktmerkmale durch Triangulierung. Im Anschluss an die Initialisierung kann jeweils nach einer Reihe von Folgebildern eine Schätzung der Kameraposition durch den 3-Punkte-Algorithmus erfolgen, indem die 3D-Positionen in die Schätzung mit einbezogen werden. Nach jeder neuen Schätzung werden die 3D-Punkte neu trianguliert. Der Initialisierungsschritt wird in regelmäßigen Abschnitten erneut durchgeführt, um Drift zu vermeiden. Echtzeitfähigkeit wird in diesem Fall durch hardwarespezifische Optimierungen erreicht. Bei solchen Triangulierungsverfahren ist es besonders wichtig, Ausreißer in den Punktmerkmalen zu eliminieren (*"Robust Estimation"*). Nistér verwendet dazu in den einzelnen Phasen eine Erweiterung des RANSAC-Verfahrens [FB81, Nis03], das üblicherweise zu diesem Zweck bei der Rekonstruktion verwendet wird.

Tensoren sind verallgemeinerte Darstellungen algebraischer Strukturen wie Vektoren und Matrizen für Räume beliebiger Dimension [HZ03, Kap. 15, 16, 18]. Während die fundamentale Matrix (bzw. die essentielle Matrix für den Fall kalibrierter Kameras) Punktkorrespondenzen zwischen zwei Bildern unter einer Projektion in Beziehung setzt, lässt sich diese Notation mit Hilfe des Trifokalsensors auf drei Bilder und mit Hilfe des Quadrifokaltensors auf vier Bilder ausweiten. Daraus lassen sich Verfahren zur direkten Berechnung der Kamerabewegung herleiten [HZ03, Kap.18.5.2], die jeweils zwischen 3 oder 4 Bildern den Trifokal- oder Quadrifokaltensor berechnen. Diese Berechnungen sind jedoch im Vergleich zu den vorab genannten sehr komplex. Ein Nachteil der Tensoren ist auch, dass durch Hinzufügen weiterer Punktfolgen eine starke Vergrößerung der Matrizen bedingt wird.

2.3.2 Probabilistische Verfahren

Man kann versuchen, die gesuchte 3D-Struktur als Systemmodell zu formulieren, dessen Zustand man zu diskreten Zeitpunkten aufgrund von Messdaten schätzt. Diese Herangehensweise wird üblicherweise mit statistischen Methoden wie Kalman Filtern [May79, JU97] oder Partikelfiltern [IB98] realisiert. Die Formulierung des Problems erfolgt dann - anders als bei den algebraischen Methoden - direkt im dreidimensionalen Raum. Man spricht daher auch von “*Top-Down*”- oder ”3D-3D“-Methoden. Die Übergänge zwischen den Zuständen des Systems müssen den Einfluss der Sensoren und eventuelle Störeinflüsse möglichst sinnvoll modellieren. Solche Verfahren unterscheiden sich hauptsächlich durch die Art der Systemmodelle, insbesondere deren Abstraktionsgrad.

Andrew Davison implementiert unter Verwendung eines Erweiterten Kalman Filters Verfahren für Roboter [Dav03] und Wearable Computing [DMM03]. Der Filter schätzt laufend die Kameraposition, die Translations- und Winkelgeschwindigkeit sowie die 3D-Position einer geringen Menge von Bildmerkmalen. Die initiale Schätzung der Position dieser Merkmale ermittelt Davison separat mit einem Partikelfilter, der eine gleichmäßige Verteilung von Hypothesen auf einer Halbgeraden im 3D-Raum vornimmt. Diese Halbgerade ist durch die initiale Kameraposition und die Merkmalsposition im Bild definiert. Mit jedem neuen Bild verändern sich die Wahrscheinlichkeiten der Hypothesen, bis schließ-

lich die Dichte hinreichend durch eine Gaussfunktion angenähert und die Merkmalposition bestimmt werden kann.

Ein weiteres probabilistisches Verfahren wurde von Jin, Favaro und Soatto [JFS03] vorgestellt. Die Idee dabei ist, Bilddaten als Messwerte der projizierten Szene im engeren Sinne zu begreifen. Auch hier wird ein Erweiterter Kalman Filter verwendet, in dessen Systemmodell Beleuchtungsparameter, Oberflächennormalen, starre Bewegungen und Geschwindigkeiten der Szene einfließen. Oberflächen werden in Anlehnung an Modelle aus der Computergrafik durch planare Patches modelliert. Zur Schätzung der Systemzustände werden allerdings nicht Punkte, sondern Bildregionen verfolgt.

Der Kalman Filter ist ein iteratives, rekursives Verfahren zur Schätzung des Zustandes eines linearen Systemmodells anhand diskreter zeitabhängiger Messungen. Der Filter basiert auf der Grundannahme, dass sowohl die Übergänge zwischen den Zuständen des Modells als auch der Zusammenhang zwischen Systemzuständen und beobachteten Messungen durch lineare Gleichungen darstellbar sind. Er berücksichtigt bei der Schätzung Fehler, die durch die Abweichung des Systemmodells von der Realität und durch das Verhalten der Sensoren bei der Messung entstehen. Diese Fehler fließen als Unsicherheiten in das Verfahren ein. Diese Unsicherheiten werden als voneinander unabhängige Prozesse mit weißem, normalverteiltem Rauschen modelliert. Falls das zugrunde liegende Problem die geforderte Linearität aufweist und die Störeinflüsse einer Normalverteilung unterliegen, ermittelt der Kalman Filter die optimale Lösung im Sinne des minimalen Bayes'schen Fehlers. Wird jedoch eine der Annahmen verletzt, schlägt der Filter fehl.

Der Erweiterte Kalman Filter (EKF) wurde entwickelt, um darüber hinaus auch nichtlineare Prozesse modellieren zu können. Dazu werden die nichtlinearen Funktionen für die Zustandsübergänge des Systems sowie die Entstehung der Messwerte durch partielle Ableitungen linear angenähert. In jeder Iteration wird so durch die Jacobi-Matrix eine Taylornäherung ersten Grades der Systemgleichungen vorgenommen. Der EKF erschließt ein weitaus größeres Feld von Anwendungsmöglichkeiten als der Standard Kalman Filter. Leider wird durch die lineare Annäherung der Prozesse jedoch die statistische Modellierung geschwächt, so dass hier nicht immer die optimale Lösung gefunden wird.

Eine Alternative zum EKF stellt der "*Unscented Kalman Filter*" (UKF) dar, der 1997 von Julier und Uhlmann in [JU97] vorgestellt wurde. Die Grundidee dabei ist, dass ei-

ne Gaussverteilung in der Regel leichter approximiert werden kann als eine beliebige nichtlineare Funktion. Daher wird beim UKF nicht wie im Falle des EKF die System- oder Messwertgleichung linear angenähert. Stattdessen werden Samples der nichtlinearen Funktionen in einer statistisch begründeten Umgebung des aktuellen Funktionswertes gezogen und mit der nichtlinearen Funktion in den Bildraum transformiert. Anhand der resultierenden Werte im Bildraum wird dann versucht, dort eine geeignete Gaussverteilung zu approximieren.

2.3.3 Einbindung von Vorwissen

Oft sind insbesondere in AR-Anwendungen bereits 3D-Modelle von Objekten bekannt, die in der Szene auftauchen können und augmentiert werden sollen. In solchen Fällen ist es sinnvoll, visuelle Merkmale der Objekte offline zu extrahieren und später im Bilddatenstrom zu suchen. Aufgrund der relativen Positionen der gefundenen Merkmale zueinander lässt sich dann anhand des bekannten 3D-Modells die Struktur der Szene ermitteln. Verfahren dieser Form sind stark verbreitet, als Beispiel sei die in [BPS05] vorgestellte AR-Applikation für Industrieanwendungen genannt.

Das Rekonstruktionsproblem für die in Abschnitt 2.3.2 genannten Verfahren lässt sich jedoch auch bereits durch weniger konkretes Vorwissen vereinfachen: Wenn bekannt ist, dass die detektierten Interessenspunkte auf der gleichen Ebene im dreidimensionalen Raum liegen, lässt sich pro Bild eine Homographie bestimmen, die diese Ebene ihrer Projektion im Bild zuordnet [HZ03]. Die Zuordnung der Projektionen in den Bildern untereinander ist wiederum selbst eine Homographie. Die Schätzung von Homographien ist erheblich einfacher als die Schätzung voller Projektionsmatrizen: Bezogen auf die intrinsischen und extrinsischen Kameraparameter bedeutet die Kenntnis um die planare Anordnung der Punkte im Raum, dass ein wesentlicher Teil der Parameter bereits bekannt ist. Letztlich verbleibt als Optimierungsproblem in diesem Fall die Schätzung der Translationsanteile. Ein solches Verfahren wird z. B. von Gilles Simon und Andrew Zisserman in [SFZ00] beschrieben. Die Annahme planarer Flächen ist realistisch für Szenen zwischen Gebäuden und auf offenen Flächen, bei denen tatsächlich überwiegend Merkmale auf Ebenen detektiert werden.

Eine weitere Form der Verwendung von Vorwissen ist der Einsatz sogenannter *Keyframes*. Damit bezeichnet man offline generierte Bildaufnahmen von Objekten, die später im Bilddatenstrom verfolgt werden sollen. Pro Objekt wird eine Vielzahl von Keyframes aus unterschiedlichen Betrachtungsrichtungen erzeugt und daraus markante Merkmale extrahiert. Während der Online-Verarbeitung erfolgt dann eine merkmalbasierte Zuordnung des ähnlichsten Keyframes zu dem aktuellen Bild im Datenstrom. Da zu den Merkmalen des Keyframes bereits 3D-Information vorhanden ist, ist eine robuste Rekonstruktion möglich. Vacchetti, Lepetit und Fua schlagen ein solches Verfahren in [VL04] zur Echtzeitverfolgung rigider Objekte vor. Als Merkmale dienen Harris-Punktmerkmale. Die Offline-Erzeugung der Merkmale zu den Keyframes wird in diesem Verfahren möglichst präzise und aufwendig realisiert. Parallel zur Zuordnung der Keyframes werden online Punktkorrespondenzen zum jeweils vorangehenden Bild im Datenstrom generiert und so eine glatte Kamerabewegung zwischen den beiden Bildern erzwungen.

Falls keinerlei Vorwissen über die Szene verwendet werden soll, steht für die Selbstlokalisation als Information nur das Ergebnis vorangehender Schätzungen und der Startzustand des Systems zur Verfügung. Die während der Echtzeitverarbeitung gefundenen Punktmerkmale müssen bewertet, gespeichert und wieder aufgefunden werden. Dazu ist eine dynamische Verwaltung der Merkmale notwendig, die in dieser Arbeit als *Karte* bezeichnet wird. Durch die Karte ergibt sich ein implizites 3D-Modell ohne Semantik. Der Begriff einer Karte ist bei der Selbstlokalisation in der Robotik weit verbreitet. Dort ist jedoch häufig eine Karte im engeren Sinne gemeint, die menschenlesbare Information über die Umgebung enthält.

2.4 Initialisierung natürlicher Marker

Markerlose Rekonstruktionsverfahren, die im Kontext dieser Arbeit in Betracht kommen, greifen auf eine Karte mit natürlichen Markern zurück, um eine Orientierung im Raum zu erhalten. Diese Karte wird anhand von Bildmerkmalen beim Start des Systems initialisiert. Darüber hinaus müssen im laufenden Betrieb neue Marker erzeugt werden, falls vorhandene Marker wiederholt nicht aufgefunden werden. Diese Fähigkeit wird in der Literatur oft unter dem Begriff *Repeatable Localisation* zusammengefasst.

In [PC05] wird die Initialisierung anhand manuell markierter Punkte auf einer Ebene im Raum vorgenommen, zu der die Kamera parallel positioniert wird. Die Anordnung der 3D-Punkte auf einer Ebene ermöglicht eine vereinfachte Schätzung der 3D-Struktur in Form einer Homographie, wie sie auch im vorangehenden Abschnitt erwähnt wurde.

Die 3D-Position eines Punktes lässt sich durch Triangulierung berechnen, wenn er zusammen mit einer ausreichenden Anzahl weiterer Punkte in zwei Bildern der Szene identifiziert werden kann. Dazu muss der Abstand der Kamerapositionen so groß sein, dass der Weltpunkt auf verschiedene Positionen im Bild projiziert wird - es muss also eine ausreichend große Parallaxe vorhanden sein. Wenn außerdem die intrinsischen und die extrinsischen Kameraparameter bekannt sind, ist die so errechnete Position absolut. Das Ergebnis hängt stark davon ab, wie stabil die Zuordnung der Punktpositionen insgesamt war. Da mit Fehlern unterschiedlicher Art in den Eingabedaten zu rechnen ist, werden unterschiedliche Bildpaare der gleichen Szene zwangsläufig zu abweichenden Berechnungen führen. Die Schätzung der 3D-Position wird somit über den Bilddatenstrom hinweg unregelmäßig sein, wenn keine zusätzliche Ausgleichsberechnung erfolgt.

Im vorigen Abschnitt wurde bereits ein weiterer Ansatz zur Initialisierung natürlicher Marker zitiert, der mit einem Partikelfilter arbeitet [Dav03]. Der Partikelfilter verfolgt eine hohe Anzahl verschiedener Hypothesen über die 3D-Position eines Punktes, die alle auf einer Halbgeraden im Raum liegen. Die Halbgerade wird bei der ersten Beobachtung des Punktes durch das optische Zentrum der Kamera und die Koordinaten der beobachteten Position im Bild definiert. Mit jeder erneuten Beobachtung werden bestehende Hypothesen statistisch bewertet und neue Hypothesen in Regionen hoher Wahrscheinlichkeit erzeugt. Sobald sich die Partikelverteilung auf der Halbgeraden einer Gausskurve annähert, kann der Mittelwert der Gausskurve als wahrscheinlichste 3D-Position akzeptiert werden. Zusätzlich steht durch die Varianz der Kurve ein Maß für die Sicherheit dieser Schätzung zur Verfügung. Leider wird auch für diesen Ansatz eine deutliche Parallaxe benötigt. Genau wie im Fall der Triangulierung repräsentieren Beobachtungen ohne Parallaxe faktisch einen Punkt im Unendlichen, der in diesem Kontext keine verwertbare Information erzeugt.

In [MCD06] wird basierend auf dem eben beschriebenen Ansatz eine neue Parametrisierung für die 3D-Position natürlicher Marker vorgeschlagen, die sich sehr gut für den

Einsatz im Zusammenhang mit einem Kalman Filter eignet. Dabei wird ebenfalls bei der ersten Beobachtung eine Halbgerade im Raum erzeugt, welche eine Einschränkung der tatsächlichen Position des Punktes bildet. Diese wird jedoch nun über die inverse Tiefe auf der Halbgeraden modelliert. Die Parametrisierung einer Schätzung $\boldsymbol{y}_i$ ist dann 6-dimensional:

$$\boldsymbol{y_i} = \begin{pmatrix} r_{x_i} & r_{y_i} & r_{z_i} & \theta_i & \phi_i & \rho_i \end{pmatrix}^T \quad .$$

Hierbei stellt $\begin{pmatrix} r_{x_i} & r_{y_i} & r_{z_i} \end{pmatrix}^T$ das optische Zentrum der Kamera zum Zeitpunkt der ersten Beobachtung des Punktes dar. Eine Skizze der Szene zum Zeitpunkt der Initialisierung zeigt Bild 2.1. Die Punktposition wird durch ihre Tiefe d_i auf dem Strahl $\boldsymbol{m}(\theta_i, \phi_i)$ bestimmt, wobei θ den Drehwinkel (der Winkel auf dem Scheitelkreis bezogen auf das optische Kamerazentrum) und ϕ den Kippwinkel (die Höhe des Strahls relativ zu der horizontalen x-/z-Ebene) angibt. Die Tiefe wird nicht direkt, sondern invers als $\rho_i = \frac{1}{d_i}$ kodiert. Die 3D-Position des Markers in euklidischen Koordinaten erhält man daraus mit

$$\begin{pmatrix} r_{x_i} \\ r_{y_i} \\ r_{z_i} \end{pmatrix} + \frac{1}{\rho_i}\boldsymbol{m}(\theta_i, \phi_i) \quad . \tag{2.6}$$

Hier steht $\boldsymbol{m}$ für die Strahlgleichung, die sich aus Kipp- und Drehwinkel ergibt. Die inverse Tiefe hat gegenüber der direkten Verwendung von d_i den Vorteil, dass der Fehler bei der Linearisierung der entsprechenden Gleichungen, wie sie in einem Erweiterten Kalman Filter notwendig wäre, deutlich geringer ist. In [MCD06, Kap. III] wird dazu ein Beweis geführt. Darüber hinaus ist mit dieser Form der Parametrisierung gleichzeitig eine gültige Repräsentation für Punkte im Unendlichen gegeben. In dem zitierten Beitrag wird außerdem aufgezeigt, wie bereits zum Zeitpunkt der Initialisierung eine Modellierung der Sicherheit über die geschätzte 3D-Position als Normalverteilung möglich ist. Diese Eigenschaft unterstützt die sofortige Verarbeitung des Merkmals durch einen Erweiterten Kalman Filter.

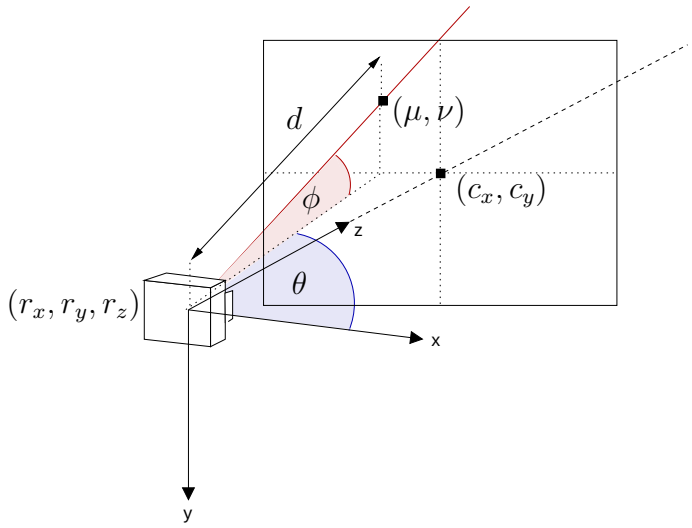


Bild 2.1: Szene zum Zeitpunkt der Initialisierung eines natürlichen Markers durch seine Tiefe auf einer Halbgeraden im Raum.

2.5 Fusion von Bild- und Inertialsensordaten

Suya You und Ulrich Neumann schlugen 2001 ein hybrides Verfahren für Bilder und Gyroskopsensoren vor, das ein Trackingsystem mit sechs Freiheitsgraden mittels eines zweikanaligen Erweiterten Kalman Filters realisiert [YN01]. Zwischen der Zustandsprädiktion und -innovation des Kalman Filters kann jeweils unabhängig voneinander eine Messwertkorrektur durch den Bilddatenstrom oder die Gyroskopdaten erfolgen. Durch die Verteilung auf zwei Verarbeitungskanäle ist es möglich, die Gyroskopdaten in höheren Frameraten zu verarbeiten als die Kamerabilder. Der Zustandsvektor des Systemmodells beinhaltet die Kameraposition sowie die lineare Translationsgeschwindigkeit, Rotationswinkel und Rotationsgeschwindigkeit der Kamera. Dabei wird der Rotationswinkel im Koordinatensystem der Gyroskopsensoren, die Position aber im Koordinatensystem der Kamera modelliert. Das beschriebene System benutzt allerdings künstliche Marker in der Szene.

Strelow und Singh [SS03] setzen einen Erweiterten Kalman Filter ein, der ebenfalls unabhängige Messwertkorrekturen für die beiden Sensordatenströme vorsieht. Der Zustandsvektor beinhaltet zusätzlich zu den oben genannten Parametern die 3D-Koordinaten ver-

folgter Punkte, die sich über eine Anzahl von Bildern als stabil erwiesen haben. Neue Punkte werden nach einer Heuristik in den Zustandsvektor aufgenommen, die auf der Länge der Hauptachse der Punkt-Kovarianzmatrix beruht.

Gabriele Bleser und Didier Stricker [BWBS06] arbeiten mit einem Geräteprototyp, der Inertialsensor und Kamera in einem Gerät vereint und so eine Kalibrierung der beiden Geräte unnötig macht. Das Gerät liefert einen synchronen Datenstrom mit Beschleunigung und Winkelgeschwindigkeit. Die relative Kameraposition kann daraus durch Integration ermittelt werden. Da solche Integrationsverfahren starken Drift in den geschätzten Daten produzieren, wird zusätzlich ein bildbasierter Tracking-Algorithmus verwendet, der 3D-Modelle bekannter Objekte zuordnen und die Schätzung damit regelmäßig reinitialisieren kann. Die Kernidee des Verfahrens ist die klare Unterscheidung zwischen präzisen Reinitialisierungsphasen, in denen die Kameraposition mittels SIFT-Merkmalen von Grund auf neu geschätzt wird, und Online-Trackingphasen, die sich der Inertialdaten bedienen und Objekte mit sehr viel einfacheren, texturbasierten Merkmalen verfolgen. Die Fusion der Sensordaten wird auch hier mit einem Kalman Filter realisiert.

Alenyà, Torras und Martínez setzen Bilder zusammen mit Inertialsensordaten zur Rekonstruktion der Bewegung eines Roboters ein [AMT04]. Das Verfahren basiert, anders als in dieser Arbeit vorgesehen, auf Konturmerkmalen, die als B-Splines repräsentiert werden. Die Projektion wird in diesem System schwach perspektivisch modelliert. Zur Fusion wird auch hier ein Kalman Filter eingesetzt.

Ein weiteres interessantes Verfahren zur Fusion von Bildern und Inertialdaten, das auf Gaborfiltern beruht, wurde von Justin Domke und Yiannis Aloimonos vorgestellt [DA06]. Dieses Verfahren ist allerdings ohne spezielle Hardware nicht echtzeitfähig.

Kapitel 3

Entwicklung eines Gesamtkonzepts

3.1 Zielsetzungen

Das Ziel dieser Diplomarbeit ist die Entwicklung eines Lösungsansatzes zur markerlosen Selbstlokalisation für das in der Einleitung beschriebene Projekt *IPCity*. Bezüglich der Selbstlokalisation stehen im Kontext des Projekts noch keine Vorarbeiten zur Verfügung, so dass die Grundlagen für ein solches Verfahren zu erörtern sind. Im vorangehenden Kapitel wurden einige Veröffentlichungen vorgestellt, die sich bereits mit ähnlichen Fragestellungen beschäftigt haben oder Techniken zeigen, die bei der Umsetzung hilfreich sein könnten.

Zunächst müssen aus den zitierten Verfahren Vorschläge zur Lösung des Rekonstruktionsproblems abgeleitet werden. Dabei soll nach Möglichkeit der am Fraunhofer Institut für angewandte Informationstechnik verfügbare Inertialsensor vom Typ *Xsens MTx* integriert werden. Im folgenden Abschnitt 3.2 werden zwei Verfahren hergeleitet, die sich grundlegend unterscheiden. Das Kapitel schließt mit einem Vergleich und der Entscheidung für eines der Verfahren. Darauf aufbauend wird anschließend ein Konzept für die konkrete Umsetzung erarbeitet. Die Beschreibung beginnt in Abschnitt 3.3 mit einer Darstellung der Struktur des Systems. Hier erfolgt eine Gliederung in Teilprozesse, auf die in den übrigen Abschnitten des Kapitels im Detail eingegangen wird.

Basierend auf dem erarbeiteten Konzept wurde eine Klassenbibliothek in C++ entwickelt, die konkrete Implementierungen der verschiedenen Komponenten bereitstellt. Die Bibliothek wird nach der Beschreibung des Gesamtkonzepts in Kapitel 4 vorgestellt und durch Simulationen und Experimente in Kapitel 5 bewertet.

3.2 Lösungsansätze zur Rekonstruktion

Im Folgenden werden zwei alternative Ansätze zur Lösung des Rekonstruktionsproblems beschrieben. Beide Lösungsansätze benötigen eine Liste natürlicher Marker, die hier als *Karte* bezeichnet wird. Ein natürlicher Marker besteht aus einer Position im dreidimensionalen Raum und einer Beschreibung - dem sogenannten *Deskriptor*. Die Position des i-ten Markers in der Karte ist durch seine euklidischen Koordinaten gegeben und wird im Folgenden in der Form $\boldsymbol{y}_i = (\ x_i \quad y_i \quad z_i\ )^T$ angegeben. Der Deskriptor ist ein Vektor konstanter Länge, der eine Beschreibung der Bildumgebung des Punktes darstellt, anhand derer eine erneute Identifikation in weiteren Bildern möglich ist. Die Identifikation kann beispielsweise durch den minimalen euklidischen Abstand von Deskriptoren erfolgen. Eine genaue Erläuterung der später zum Einsatz kommenden Deskriptoren erfolgt in Abschnitt 3.4.2.

3.2.1 Probabilistischer Lösungsansatz

Der erste Lösungsansatz zählt zu den probabilistischen Verfahren, von denen einige in Abschnitt 2.3.2 vorgestellt wurden. Ähnlich des in [DMM03] beschriebenen Verfahrens wird ein Erweiterter Kalman Filter verwendet, um laufend die Inertial- und Bilddaten zur Schätzung der Kameradaten auszuwerten. Dabei werden die Orientierungsdaten des *MTx* als Referenzdaten behandelt und nicht als unsichere Messwerte im Sinne des Kalman Filters. Dem Datenblatt des *MTx* ist die Eignung als Mess- und Referenzgerät zu entnehmen.

Der Zustandsvektor des Systems enthält die Translation $\boldsymbol{t}$ und Richtungsgeschwindigkeit $\boldsymbol{v}$ der Kamera:

$$\boldsymbol{x} = \begin{pmatrix} \boldsymbol{t} \\ \boldsymbol{v} \end{pmatrix} \tag{3.1}$$

Sowohl $\boldsymbol{t}$ also auch $\boldsymbol{v}$ sind 3D-Vektoren im Weltkoordinatensystem. Die natürlichen Marker $\boldsymbol{y}_i$ sind ebenfalls Teil des Systemzustandes. Im Gegensatz zu [DMM03] entfällt die Modellierung der Winkelgeschwindigkeit ω als Parameter für die Zustandsübergänge, da die Orientierung nicht in den Systemzustand einbezogen wird.

Für das Systemmodell wird eine konstante Geschwindigkeit der Kamera angenommen. In jeder Iteration verschiebt sich die Kameraposition aufgrund der aktuellen Geschwindigkeit und einer unbekannten Beschleunigung, die im Rauschen des Systemmodells enthalten ist. Als unsichere Messdaten des Systems dienen die Bildpositionen der verfolgten Punktmerkmale. Für die Punktmerkmale lässt sich aufgrund des letzten aktuellen Systemzustandes eine Prädiktion der 3D-Position im Kamerakoordinatensystem der nächsten Iteration vornehmen, die sich wiederum in eine Vorhersage der Projektion im neuen Bild umrechnen lässt. Diese Berechnung ist ohne Umwege möglich, da die intrinsischen Kameraparameter bekannt sind. Gleichzeitig kann die Unsicherheit der Prädiktion durch zusätzliche Berücksichtigung der aktuellen geschätzten Kamera- und Merkmalsposition berechnet werden und in die Systemkovarianz einfließen.

Im vorangehenden Kapitel wurde bereits erwähnt, dass ein robustes Verfahren zur Selbstlokalisation die Fähigkeit aufweisen muss, laufend einzelne natürliche Marker neu zu erzeugen (*"Repeatable Localisation"*). Dieser Prozess wird hier als Reinitialisierung natürlicher Marker bezeichnet. Sowohl für die initiale Erstellung der Karte als auch für die Reinitialisierung würde sich bei diesem Lösungsansatz das Verfahren aus [MCD06] sehr gut eignen, das die bereits beschriebene Parametrisierung der Schätzungen durch inverse Tiefe vorsieht.

Das beschriebene Systemmodell setzt voraus, dass die Bewegung der Kamera relativ glatt verläuft. Sehr schnelle Kopfbewegungen des Anwenders werden also zwangsläufig zu Problemen führen. Es ist denkbar, zur Vermeidung dieses Problems zusätzlich die Winkelbeschleunigungsdaten des Inertialsensors auszulesen. Liegt eine abrupt hohe Winkelbeschleunigung vor, könnte man den Vorgang kurz unterbrechen und anschließend wieder aufnehmen. Falls die letzten bekannten Punktmerkmale dann nicht mehr sicher detektiert werden können, muss eine Reinitialisierung stattfinden. Generell ist es im Vorfeld schwer, die Robustheit und Präzision des Kalibrierverfahrens einzuschätzen, wovon die Qualität der Ergebnisse aber deutlich abhängen wird. Ein Vorteil dieses Lösungsansatzes ist aller-

dings, dass durch die iterative Vorgehensweise unruhige Bewegungen in der Schätzung vermieden werden. Außerdem kann die probabilistische Modellierung durch die Kovarianzen ohne zusätzlichen Aufwand ein Qualitätskriterium für die verfolgten Punkte liefern.

3.2.2 Algebraischer Lösungsansatz

In Kapitel 2.3.1 wurde eine vereinfachte Lösung des Rekonstruktionsproblems mittels Faktorisierung genannt, die Punktemengen auf planaren Ebenen im 3D-Raum verarbeitet. In diesem Fall sind Teile der zu schätzenden Kameraparameter bereits implizit bekannt. Mathematisch ergibt sich der gleiche Fall, wenn die intrinsischen Parameter sowie die Orientierung der Kamera gegeben sind [HZ03, Kap. 18.5]. Dann ist die gleiche vereinfachte Lösung des Rekonstruktionsproblems auch möglich, wenn die Punktpositionen keiner Einschränkung unterliegen. Dieser Fall ist in der vorliegenden Aufgabenstellung gegeben: Die intrinsischen Parameter sind bekannt, da die verwendete Kamera vorkalibriert ist und mit fester Brennweite arbeitet. Die Orientierung kann von dem Inertialsensor geliefert werden. So lässt sich eine algebraische Lösung des Rekonstruktionsproblems ableiten, die nun skizziert wird. Der Ablauf orientiert sich an den in [NNB04] beschriebenen Verarbeitungsschritten zur robusten Schätzung.

Über eine im Experiment zu bestimmende, größere Anzahl von Bildern werden Punktmerkmale verfolgt und Trajektorien dazu gebildet. Durch Singulärwertzerlegung wird die relative Kamerabewegung zwischen den Bildern aus den Punktmesswerten ermittelt. Dazu wird das in [HZ03, Kap. 18.5.1] beschriebene Verfahren entsprechend angepasst. In diesen Schritt muss zur Verbesserung der Robustheit außerdem ein RANSAC-Verfahren zur Erkennung von Ausreißern integriert werden. Aufgrund der ermittelten Kamerabewegung lassen sich dann durch Triangulierung gezielt die 3D-Positionen der verfolgten Punktmerkmale bestimmen. Dazu werden das erste und das letzte Bild der Initialisierungsfolge herangezogen.

Die Punkte werden dann über eine deutlich geringere Anzahl von Bildern hinweg verfolgt als im ersten Schritt. Die Bewegung der Kamera wird nach dem gleichen Prinzip, aber mit entsprechend kompakteren Gleichungssystemen fortlaufend rekonstruiert. Gegebenenfalls wird auch hier wieder ein Verarbeitungsschritt zur Eliminierung von Ausreißern durchge-

führt. Für die Rekonstruktion wird jeweils der Rückprojektionsfehler aufgrund der initial geschätzten 3D-Positionen der Punkte minimiert. Außerdem wird als Kriterium für die Glattheit der Bewegung die jeweils vorangehende Schätzung der Bewegung in Form eines einfachen lokalen Bündelausgleichs herangezogen.

Für jede wiederholte Kameraschätzung erfolgt zur Neuberechnung der 3D-Positionen eine erneute Triangulierung basierend auf dem ersten und letzten Bild der entsprechenden Teilbildfolge. Die beschriebenen Verarbeitungsschritte mit kurzen Bildfolgen werden mehrmals wiederholt. In größeren Zeitabständen wird das gesamte Verfahren wiederholt.

3.2.3 Vergleich der beiden Alternativen

Der zuletzt beschriebene algebraische Lösungsansatz berechnet wiederholt Rekonstruktionen für eine geringe Anzahl von Bildern, die in einem beschränkten Umfang gegeneinander abgeglichen werden. Diese Teillösungen sind optimal im Sinne des geringsten quadratischen Fehlers. Es wird jeweils nur ein sehr begrenzter Bereich des Bilddatenstroms betrachtet, für den eine weitgehend eindeutige Lösung existiert. Die Folge ist, dass sich Rekonstruktionsfehler nicht so sehr über die Zeit akkumulieren, wie das bei dem probabilistischen Ansatz der Fall ist. Punktkorrespondenzen müssen nur zwischen wenigen Bildern abgeglichen werden, da darauf folgende Berechnungen unabhängig erfolgen. Das kann eine erhebliche Vereinfachung der Bildverarbeitung darstellen. Damit geht jedoch auch einher, dass die rekonstruierte Kamerabewegung sehr unruhig sein kann, weil sich die Fehler zwischen aufeinander folgenden Berechnungen nicht gegenseitig bedingen (Jitter). Zur Glättung der rekonstruierten Bewegung dienen die Ausgleichsverfahren (lokaler und globaler Bündelausgleich), die sowohl die Glattheit der Gesamtbewegung als auch die Korrektheit der einzelnen Berechnungen gleichzeitig mit einbeziehen. Die Qualität des Gesamtergebnisses wird also entscheidend davon abhängen, wie viele Ausgleichsberechnungen durchgeführt und wie viele Einzelberechnungen einbezogen werden.

Sehr gute Ergebnisse lassen sich mit algebraischen Verfahren insbesondere in Anwendungen erzielen, bei denen der gesamte Datenstrom zum Zeitpunkt der Berechnung vorliegt und für den Ausgleich herangezogen werden kann. Hier ist das nicht der Fall: Der Datenstrom muss synchron verarbeitet werden, und es können zu einem Zeitpunkt t lediglich

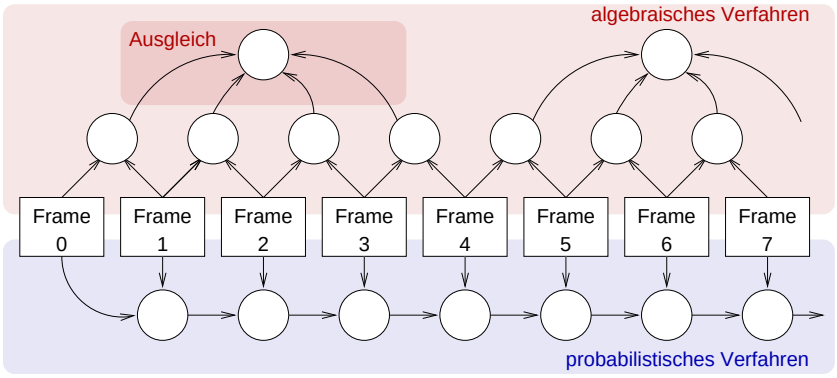


Bild 3.1: Schematische Darstellung des Informationsflusses von Sensordaten und Zwischenergebnissen beim algebraischen und probabilistischen Lösungsansatz. Die weißen Kreise stellen Zwischenergebnisse dar, die Rechtecke jeweils Sensordateninformation (Frames).

die aktuellen sowie die vorangehenden Sensordaten betrachtet werden. Hinzu kommt, dass die Sensordaten in definierten, konstanten Zeitabständen (Frames) anfallen und innerhalb dieser verarbeitet werden müssen. Es wäre sinnvoll, pro Frame nur Berechnungen mit möglichst gleich bleibendem Aufwand durchzuführen. Aus diesem Grund unterliegt die Qualität des algebraischen Verfahrens hier einer natürlichen Einschränkung.

Genau da liegt der Vorteil des probabilistischen Ansatzes: Die Lösung des Problems, also die rekonstruierte Kamerabewegung, ist Zustand eines dynamischen Modells und wird in jeder Iteration aktualisiert. Diese Anpassung erfolgt unter Berücksichtigung des bereits berechneten Zustands sowie der neu hinzugekommenen Sensordaten und nimmt fortlaufend eine statistische Bewertung dieser verschiedenen Informationen vor. Die Berücksichtigung der Zwischenergebnisse und der verfügbaren Information aus den einzelnen Frames folgt also einem anderen Schema als bei dem algebraischen Ansatz. In Bild 3.1 sind diese Schemen des Informationsflusses für die beiden Ansätze zur Verdeutlichung grafisch angedeutet. Die Menge an verwerteter Information pro Frame bleibt bei dem probabilistischen Ansatz weitestgehend gleich. Der Systemzustand lässt sich als Akkumulation aller bisherigen Zwischenberechnungen auffassen, da alle bereits vorhandene Information darin vorliegt. So erfolgt implizit eine Betrachtung der gesamten bereits gelesenen Information,

was Jitter vermeidet und die Rekonstruktion einer glatten Kamerabewegung ermöglicht. Die Akkumulation vorangehender Ergebnisse bringt jedoch gleichzeitig die bereits weiter oben angesprochenen Drift-Probleme mit sich.

In erster Linie scheint die Entscheidung für einen der beiden Ansätze also bestimmt zu sein von der Frage, ob man sich eher dem Problem Jitter oder dem Problem Drift stellen möchte. Da das Trackingsystem für eine AR-Anwendung eingesetzt werden soll, fließen die Daten direkt in eine Benutzerschnittstelle ein. Dort spielen ästhetische und ergonomische Aspekte eine große Rolle. Eine möglichst glatte Rekonstruktion der Kamerabewegung hat aus diesem Betrachtungswinkel eine hohe Bedeutung und würde die Entscheidung für den probabilistischen Ansatz nahe legen. Darüber hinaus ist davon auszugehen, dass in den Eingabedaten sehr hohe Rausch- und Fehleranteile enthalten sein werden. Es ist wichtig, damit kontrolliert umgehen zu können. Hier liefert der probabilistische Ansatz bessere Mittel, da eine fundierte statistische Fehlermodellierung Grundlage des Kalman Filters ist.

Ein weiterer Pluspunkt für die probabilistische Herangehensweise ist die hervorragende Eignung der Kalman Filter zur Einbindung verschiedener Sensordatenquellen. In Abschnitt 2.5 wurden mehrere Verfahren der Datenfusion vorgestellt, die nahezu ausnahmslos auf Erweiterten Kalman Filtern basieren. Sollte sich später herausstellen, das weitere Sensoren zur Verfügung stehen, die sinnvoll für die Rekonstruktion eingesetzt werden könnten, bietet eine solche Implementierung die besseren Voraussetzungen, diese Sensordaten einzubinden.

Aus den genannten Gründen basiert das nun folgende Systemkonzept auf dem probabilistischen Lösungsansatz. Die Ideen des algebraischen Ansatzes können trotzdem von Interesse sein, um bei künftigen Arbeiten eine weitere Stabilisierung des Systems zu erreichen oder alternative Methoden der Systeminitialisierung zu finden.

3.3 Konzeption des Systems

Bei einem Trackingverfahren handelt es sich im Prinzip um ein *reaktives System*. Diesen Begriff verwendet man für Systeme, die auf innere oder äußere Ereignisse zustandsabhängige Reaktionen zeigen und dabei ihren eigenen Zustand verändern können. Als äußere

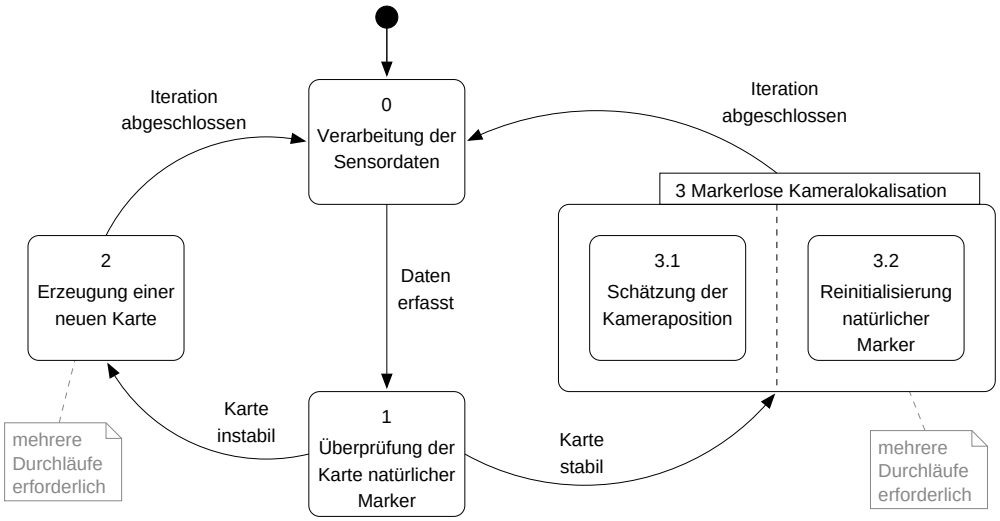


Bild 3.2: Zustandsdiagramm des Gesamtsystems

Ereignisse lassen sich die Sensordaten verstehen, auf die das Trackingsystem geeignet reagieren muss. Zur Spezifikation reaktiver Systeme empfehlen sich Zustands-Übergangs-Diagramme, die in in den folgenden Abschnitten als Basis für die Beschreibung der verschiedenen Prozesse dienen.

Ein Diagramm des Systems auf oberster Ebene ist in Bild 3.2 dargestellt. Die vier darin enthaltenen Zustände stellen Prozesse dar, die in verfeinerter Darstellung in den folgenden Abschnitten beschrieben werden.

Für die Schätzung der Kamerabewegung ist es ständig erforderlich, dass die Karte (siehe Abschnitt 3.2) eine minimale Anzahl stabiler natürlicher Marker enthält. Beim Start des Systems ist die Karte jedoch leer - es sind keine natürlichen Marker vorhanden. Das System muss also eine Initialisierungsphase bereitstellen, während der eine ausreichende Anzahl natürlicher Marker erzeugt wird. Diese Phase wird in Bild 3.2 durch den linken Zyklus dargestellt, der den Prozess 2 “Erzeugung einer neuen Karte” beinhaltet. Jeder Durchlauf dieses Zyklus basiert auf den Sensordaten zu einem Zeitpunkt t und beinhaltet eine Iteration des dort implementierten Kalman Filters. Es werden mehrere Durchläufe erforderlich sein, um eine ausreichende Anzahl von Markern stabil zu erzeugen. Der Prozess

1 “Überprüfung der Karte natürlicher Marker” entscheidet anhand der jeweils aktuellen Karte, ob die Initialisierung abgeschlossen ist und der Zyklus verlassen werden kann. Das Verfahren zur Erzeugung einer neuen Karte wird im Detail in Abschnitt 3.6 beschrieben.

Sobald die Karte einen stabilen Zustand mit ausreichender Anzahl natürlicher Marker erreicht hat, kann die Kamerabewegung rekonstruiert werden. Die Rekonstruktion wird durch den Prozess 3 “Markerlose Kameralokalisation” implementiert und bildet den Kern des in Bild 3.2 rechts dargestellten Zyklus. Auch hier beinhaltet jeder Durchlauf eines Zyklus eine Kalman Filter Iteration und basiert auf den Sensordaten zu einem Zeitpunkt t. Die geschätzte Kamerabewegung stabilisiert sich erst nach mehreren Iterationen. Eine genaue Beschreibung dieses Teils des Verfahrens erfolgt in Abschnitt 3.7.

Parallel zur Rekonstruktion kann eine Reinitialisierung natürlicher Marker erfolgen, solange in der Karte weiterhin stabile Marker in ausreichender Anzahl vorhanden sind. Der Prozess 3 ist daher in zwei voneinander unabhängige Teilprozesse aufgeteilt, von denen der Teilprozess 3.2 die Reinitialisierung einzelner Marker und somit die “Pflege” der Karte übernimmt. Der Teilprozess wird in Abschnitt 3.8 beschrieben. Sollten während der laufenden Schätzung der Kameraposition zu viele Marker verloren gehen und das Verfahren divergieren, muss das System vollständig neu initialisiert werden. Dieser Fall wird ebenfalls durch den Prozess 1 “Überprüfung der Karte natürlicher Marker” festgestellt, der den rechten Zyklus dann unterbricht.

3.4 Verarbeitung der Sensordaten

Den Ausgangspunkt der beiden im vorangehenden Abschnitt beschriebenen Zyklen bildet jeweils die Erfassung und Vorverarbeitung von Sensordaten zu aufeinander folgenden Zeitpunkten t. Das Diagramm der Verfeinerung dieses Prozesses zeigt Bild 3.3. Als Sensordaten sind in der Hauptsache die Bilder zu verstehen, welche die Kamera liefert. Es wird zunächst davon ausgegangen, dass die Bilder aktiv von einem *Framegrabber* angefordert werden, sobald eine neue Iteration begonnen wird (Teilprozess 0.0). Ein Framegrabber greift zu diskreten Zeitpunkten Daten von einem Sensor ab. Bei der späteren Anwendung in einem AR-System wird es so sein, dass die Bilder in festen Zeitabständen geliefert

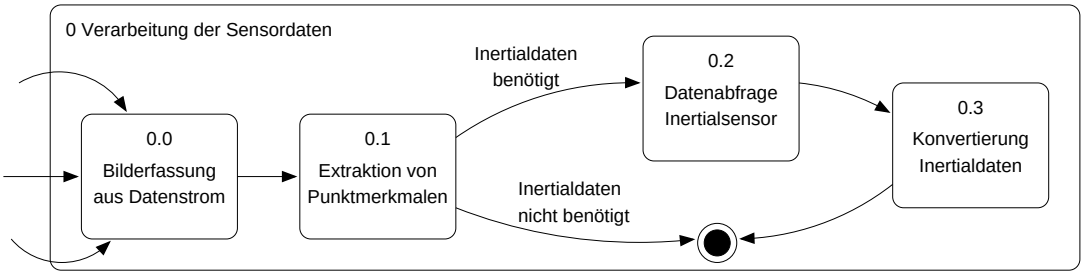


Bild 3.3: Verfeinerung des Prozesses 0 “Verarbeitung der Sensordaten”

werden. Als weitere Sensordaten liegen Orientierungsdaten des Inertialsensors vor, deren Einsatz für das System vorgesehen ist. In diesem Fall können die Daten vom Gerät abgefragt werden (Teilprozess 0.2) und müssen anschließend in eine brauchbare Formatierung gebracht werden. Die dazu notwendigen Konvertierungen sind in Teilprozess 0.3 enthalten. Neben der Umrechnung der Orientierungsdaten in die Repräsentation als Rotationsmatrix muss hier insbesondere eine Fusion der Referenzkoordinatensysteme vorgenommen werden. Eine genauere Betrachtung der Ausrichtung des Koordinatensystems des *Xsens MTx* bezüglich eines gegebenen Welt- oder Kamerakoordinatensystems erfolgt in Abschnitt 4.2.2 im Kontext der Implementierung.

Im Gegensatz zu den Orientierungsdaten des Inertialsensors fließen die Bilder nicht als solche in die Rekonstruktion und Initialisierung ein. Vielmehr benötigen die Verfahren pro Iteration eine Menge von Punkten im Bild, die Projektionen der natürlichen Marker aus der Karte sind. Die Verarbeitung des Bilddatenstroms muss also zunächst eine performante Extraktion von Punktmerkmalen im Bild gewährleisten. Später muss damit eine Zuordnung der extrahierten Punktmerkmale zu natürlichen Markern möglich sein.

In Kapitel 2.2 wurden Verfahren zur Ermittlung von Interessenspunkten in Bildern genannt. Die *Scale Invariant Feature Transform* [Low04] ist ein Verfahren, mit dem Punkte unter einer Vielzahl verschiedener Transformationen wiedergefunden und zugeordnet werden können. Leider ist das Verfahren sehr rechenintensiv und wurde bisher eher für offline-Anwendungen eingesetzt. Es bietet jedoch einige Ansatzpunkte für Vereinfachungen und Beschleunigungen. Grabner und Bischof stellten kürzlich einige Ideen vor, die eine enorme Beschleunigung des Verfahrens versprechen [GGB06]. Dazu gehört die Ver-

wendung von Mittelwertfiltern zur Glättung und der Einsatz von Integralbildern bei der Punktdetektion und bei der Erzeugung des Deskriptors.

In den folgenden Abschnitten wird ein Verfahren beschrieben, das diese Ideen aufgreift und weitere Anpassungen vornimmt, um mehrere Bilder pro Sekunde verarbeiten zu können. Mit diesem Verfahren steht eine performante und stabile Implementierung zur Ermittlung von Interessenspunkten zur Verfügung.

3.4.1 Performante Detektion von SIFT-Merkmalen

Die Detektion der Punkte wurde weitestgehend dem Verfahren nach Grabner entnommen. Aus diesem Grund wird hier mit Verweis auf die entsprechende Veröffentlichung [GGB06] nur kurz die Vorgehensweise skizziert. Das Originalverfahren nach David Lowe wurde in [Low04] veröffentlicht und wird unter anderem in [Dic05] im Detail besprochen.

Die Punktmerkmale für die *Scale Invariant Feature Transform* basieren auf Extremstellen im differentiellen Skalenraum des Bildes, der durch eine Auflösungshierarchie angenähert wird. Zur Erzeugung des Skalenraumes erfolgt eine inkrementelle Faltung des Ausgangsbildes mit Gaussfiltern in der Weise, dass ein konstanter Abstand in Richtung des Skalenparameters σ entsteht. Mit jeder Verdopplung des Skalenparameters erfolgt eine Halbierung der physischen Bildauflösung. Die Ebenen innerhalb der gleichen physischen Auflösung bezeichnet Lowe als *Oktaven*. Der differentielle Skalenraum wird schließlich durch einfache Differenzbildung benachbarter Ebenen einer Oktav gebildet. Die erste Oktav wird im Originalverfahren mit der doppelten Originalbildgrösse erzeugt, um eine höhere Abtastung des unteren Skalenraumbereiches zu erhalten. Als Punktmerkmale kommen schließlich solche Positionen im Bild in Frage, an denen der differentielle Skalenraum einen Extremwert aufweist, so dass der entsprechende Funktionswert des dreidimensionalen Raumes entweder kleiner als alle seine direkten Nachbarwerte oder größer als diese ist.

In [Dic05] wurde aufgezeigt, dass der Aufwand dieses Verfahrens im Bereich der Punktdetektion maßgeblich durch die vergrößerte erste Oktav sowie durch die Gauss-Faltungen bestimmt wird. Genau hier setzen die Vereinfachungen von Grabner und Bischof an: Der dort zum Einsatz kommende Skalenraum wird nur beginnend mit der Originalbildauflö-

sung gebildet und daher im unteren Skalenbereich weniger intensiv abgetastet. So reduziert sich die auszuwertende Bildinformation deutlich. Darüber hinaus erfolgt keine Halbierung der Bildgröße mit zunehmenden Skalenparameterwerten. Stattdessen werden sehr hohe Skalen vernachlässigt, da man annimmt, dass sie in vielen Anwendungen keinen zusätzlichen Nutzen bringen.

Die Faltung der aufeinander folgenden Ebenen wird nicht mehr durch den theoretisch korrekten Gaussfilter vorgenommen, sondern durch einen einfachen Mittelwertfilter. Dieser besitzt als Tiefpassfilter ebenfalls die Eigenschaft der Vereinfachung, erfüllt aber nicht alle Eigenschaften der Wärmeleitungsgleichung (siehe dazu [Dic05], Kapitel 3.2.2). Daher handelt es sich bei dem verwendeten Raum nach Koenderink [Koe84] nicht um einen Skalenraum im theoretischen Sinne. Grabner und Bischof berichten in ihrer Veröffentlichung jedoch von stabilen Ergebnissen.

Für den Mittelwertfilter lässt sich ein Verfahren verwenden, das auf Integralbildern beruht. Diese in [VJ01] vorgeschlagene Darstellungsform liefert ein Bild gleicher physischer Auflösung ii, das in jedem Bildpunkt die Summe der links oberhalb gelegenen Bildpunkte des Originalbildes i enthält, also an der Stelle (x, y) die Summe der Bildpunkte der Spalten 0 bis x und der Zeilen 0 bis y:

$$ii\,(x, y) = \sum_{x' \leq x, y' \leq y} i\,(x', y')$$

Aus dem Integralbild lässt sich nun für jede Bildposition die Summe einer rechteckigen Umgebung durch drei Subtraktionen berechnen: Von dem Wert des Integralbildes in der rechten unteren Ecke der gewünschten Maske sind die Werte an den drei anderen Ecken der Maske abzuziehen. Der Aufwand dieser Berechnung ist unabhängig von der Größe der Maske. Zur Berechnung des Mittelwerts ist zusätzlich eine Multiplikation pro Faltung notwendig. Das Integralbild lässt sich in einem Durchlauf mit zwei Additionen pro Pixel erstellen und kann anschließend unverändert verwendet werden, um Faltungen mit Mittelwertfiltern beliebiger Größe vorzunehmen. Insbesondere bei mehrfacher Wiederverwendung des Integralbildes ist dieses Verfahren also effizienter als ein einfach implementierter Mittelwertfilter, und deutlich effizienter als die Berechnung eines Gaussfilters.

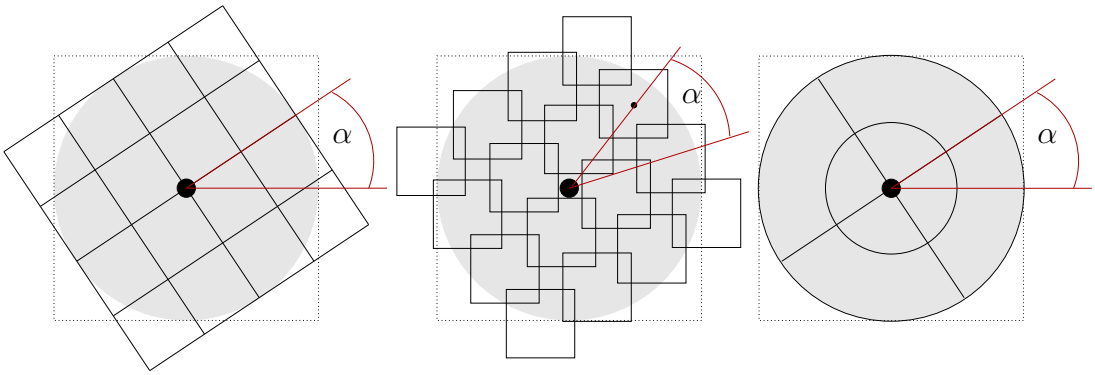


Bild 3.4: Anordnung der Teilfenster für Gradientenhistogramme bei Lowe (links), Grabner und Bischof (Mitte) und dem hier vorgestellten Verfahren (rechts). Die tatsächliche Anzahl der Teilfenster weicht von der Darstellung ab. α bezeichnet die Orientierung des Punktmerkmals, mit der die Deskriptorfenster zu rotieren sind.

3.4.2 Dynamische Deskriptoren

Wie bereits erläutert, dienen die Deskriptoren der Zuordnung korrespondierender Punktmerkmale. Dieser Abgleich ist nicht Teil des Prozesses 0 “Verarbeitung der Sensordaten”. Die Deskriptoren werden aber direkt im Anschluss an die Detektion der Punktmerkmale erzeugt und gespeichert, so dass das Konzept hier beschrieben wird.

Die von David Lowe vorgeschlagenen Deskriptoren sind 128-stellig und werden aus den Gradientenhistogrammen von 16 quadratischen Fenstern in der Umgebung des Merkmals zusammengesetzt. Die Fenster werden relativ zu einer charakteristischen Orientierung, die dem Punktmerkmal zugeordnet ist, rotiert, um Rotationsinvarianz zu erhalten. Um Unregelmäßigkeiten durch die Rotation der rechteckigen Regionen zu umgehen, werden die Werte vor dem Einfügen in die Histogramme zusätzlich mit einer kreisförmigen Gauss-Maske gewichtet. Die Vorgehensweise wird detailliert in [Dic05] besprochen. Die Deskriptoren sind äußerst distinktiv, erfordern jedoch einen hohen Berechnungsaufwand.

Grabner und Bischof verfolgen in [GGB06] zur Erstellung des Deskriptors einen ähnlichen Ansatz. Auch hier werden Gradientenhistogramme in 16 quadratischen Teilfenstern erstellt. Allerdings wird hier keine Gewichtung der Werte mehr vorgenommen, um das

Verfahren zu beschleunigen. Außerdem werden nicht alle Werte einzeln relativ zur Orientierung des Merkmals rotiert, sondern zunächst nur die Mittelpunkte der quadratischen Teilfenster. Anschließend werden die Urnen der entstandenen Histogramme, die bereits deutlich stärker diskretisiert sind, relativ zur Orientierung verschoben. Dieses Verfahren liefert also Unregelmäßigkeiten bei der Rotation der Teilfenster, da die Werte in den äußeren Ecken der Fenster ebenso verarbeitet werden wie die Werte im Innern. Außerdem führt die verspätete Rotation der Histogrammurnen zwangsläufig zu deutlich höheren Diskretisierungsverlusten.

Bei der Verwendung von SIFT-Merkmalen stehen zwei Anforderungen oft in unterschiedlicher Gewichtung orthogonal zueinander: Zum Einen sollen die Merkmale so eindeutig wie möglich einander zugeordnet werden können, zum Anderen soll das Verfahren möglichst performant arbeiten. Beide Anforderungen hängen eng mit der Größe der Deskriptoren und der Anordnung der Teilfenster für die Histogramme zusammen. Es ist daher wünschenswert, das Verfahren zur Erstellung der Deskriptoren zu parametrisieren und je nach Anwendung anzupassen. In dieser Arbeit soll daher ein schlüssiges Konzept zur Parametrisierung der Größe, Zahl und Anordnung von Teilfenstern entwickelt werden, das eine einfache Anpassung erlaubt. Das Konzept soll eine vereinfachte Berechnung wie das von Grabner und Bischof erlauben, bei dem keine erneute Gewichtung mit einer Gaussmaske notwendig ist. Außerdem soll die Rotation der Teilfenster gemäß der Orientierung des Punktmerkmals vereinfacht werden. Eine verspätete Normalisierung der Orientierung in den stark diskretisierten Histogrammen soll ebenso wenig zum Einsatz kommen wie eine aufwändige Berechnung der Rotation für jeden einzelnen Punkt im Fenster. Darüber hinaus fällt bei der Betrachtung der beiden vorgestellten Verfahren auf, dass bei der Verwendung quadratischer Fenster im Zuge der Rotation die Reichweite des Fensters ungleichmäßig verteilt ist: Die äußeren Ecken der Teilfenster ragen weiter in die Bildumgebung hinein als die flachen Seiten. Möchte man auf die Gaussgewichtung verzichten, führt das zu einer starken Variation der ausgewerteten Gradienten bei Merkmalen mit unterschiedlicher Orientierung. Dieser Effekt wird bei der schematischen Gegenüberstellung der Teilfensteranordnung für die genannten Verfahren in Bild 3.4 deutlich.

Für die Deskriptoren wurde daher ein neues Konzept eingeführt, das die Einstellung verschiedener Eigenschaften der Deskriptoren im konkreten Anwendungsfall erlaubt und

trotzdem performant ist. Das Prinzip wird in Bild 3.4 auf der rechten Seite skizziert. Anstelle einer quadratischen Struktur wird eine Kreisstruktur zugrunde gelegt, so dass auch ohne nachträgliche Gewichtung mit einer Gaussmaske Unregelmäßigkeiten bei der Rotation der Gradientenwerte vermieden werden können. Durch den Verzicht auf diese Gewichtung können viele Multiplikationen bei der Erstellung jedes einzelnen Deskriptors eingespart werden. Die Teilfenster, für die jeweils ein Gradientenhistogramm erstellt wird, werden nun als Kreisausschnitte gleicher Größe definiert. Primäre Parameter des Verfahrens sind der Radius des Kreises, die Anzahl der Kreisausschnitte sowie die Anzahl der Urnen pro Gradientenhistogramm.

Die Gauss-Gewichtung des Deskriptorfensters in [Low04] hat jedoch noch einen weiteren Effekt: Sie erhöht den Einfluss von Gradientenwerten in direkter Nähe der Merkmalsposition gegenüber solchen in den Randbereichen der Masken. Eine vergleichbare Wirkung lässt sich auch mit dem hier vorgestellten Verfahren erzielen: Das Kreismodell lässt sich in Ringe unterteilen, so dass jeder Kreisausschnitt erneut in mehrere Bereiche aufgeteilt wird. Diese Ringe sind in gleichmäßige Abstände aufgeteilt, so dass die Ausschnitte in den äußeren Bereichen deutlich mehr Werte enthalten als solche im inneren Ring. Da die Histogramme pro Teil eines Kreisauschnittes erstellt werden und später zu gleichen Teilen in den Deskriptor einfließen, ist bei der Verwendung von Ringen die Wirkung von innen liegenden Werten auf den Deskriptor erheblich höher. Dieser Effekt lässt sich durch den Parameter “Anzahl der Ringe” verstärken oder - im Fall eines einzigen Rings - aufheben. In Bild 3.4 ist das Prinzip für zwei Ringe illustriert: Die im inneren Kreisausschnitt enthaltenen Werte erhalten im Deskriptor eine stärkere Gewichtung als die in den benachbarten Ausschnitten des äußeren Rings. Das Verfahren ließe sich mit geringem Aufwand so umstellen, dass die Breite der Ringe von innen nach außen abnimmt, falls sich die Aufteilung in gleich breite Ringe doch als nachteilig erweist. Dazu wären minimale Anpassungen in der Klasse `DynamicCircleDescriptorBuilder` notwendig, die in Abschnitt 4.3.1 beschrieben wird.

Um herauszufinden, zu welchem Histogramm ein Gradientenwert zählt, muss seine relative Position zum zentralen Pixel entgegen der Orientierung des Punktmerkmals rotiert werden. Diese Rotation erfordert die Berechnung des Arcustangens für jeden Wert unter dem Fenster. Um diese Berechnung während der Online-Verarbeitung zu umgehen,

liegt der Gedanke nahe, die Berechnungen für verschiedene Orientierungen offline vorzunehmen. Die Idee ist, zur Laufzeit lediglich auf eine Tabelle zuzugreifen, die den Index des jeweiligen Teilfensters (Histogramms) für jedes Pixel der Maske bereits enthält. Dazu müsste für jede mögliche Orientierung des Punktmerkmals eine separate Tabelle existieren. Da die Anzahl von Tabellen offline festgelegt werden muss, erfordert das eine Diskretisierung der Orientierungen, die zu Problemen führen kann. Eine Diskretisierung ist jedoch ohnehin durch die Abtastung des Bildsignals in Form von Pixeln gegeben, so dass die Offline-Berechnung nur dann nachteilig wirkt, wenn die Orientierungen grober als die Bildpunkte diskretisiert würden. Hierzu ein Beispiel: Bei einem Radius des Deskriptorkreises von 8 Pixeln hat der Kreis einen Umfang von $17 \cdot \pi \simeq 53$ Pixeln. Die Orientierung ist also selbst bei der höchsten Abtastung auf der Außenlinie des Kreises in Abständen von $360/53 \simeq 6.7$ Grad diskretisiert. Würde man Zugriffstabellen in Abständen von weniger als 6 Grad generieren, so würde durch die Offline-Berechnung der Histogramm-Indizes kein kritischer Fehler entstehen. Die Auflösung der Tabellen wird im Rahmen dieser Arbeit mit 5 Grad gewählt, ist jedoch als Parameter justierbar. Es wird also eine Liste von 72 Tabellen vorberechnet, die jeweils die Größe des gewünschten Deskriptorfensters besitzen. Zwei Beispiele solcher Tabellen zeigt Bild 3.5. Für die Tabelleneinträge gilt: Alle Werte, die kleiner als die Gesamtzahl der Teilfenster sind, werden direkt als Index für das jeweilige Histogramm (also Teilfenster) verwendet. Der Wert, der genau der Gesamtzahl entspricht, verschlüsselt eine Zugehörigkeit zu allen Teilfenstern. Dieser Wert wird immer dem Mittelpunkt selbst zugewiesen. Alle Werte, die größer sind als die Gesamtzahl der Teilfenster, sind ungültig und fließen nicht in die Deskriptoren ein.

Durch die Offline-Berechnung dieser Deskriptormasken vereinfacht sich die Erstellung eines Deskriptors enorm; es ergibt sich der in Codefragment 3.1 dargestellte Pseudocode. Hierauf folgt noch die Normalisierung des Deskriptors auf einen festgelegten Wertebereich zur Verbesserung der Invarianz gegenüber Beleuchtungsschwankungen, die in [Dic05, Kap. 3.4.4] besprochen wird. Weitere Details zur Implementierung des Verfahrens werden in Abschnitt 4.3.1 besprochen.

Die Berechnung des Deskriptorabstands stellt den größten Aufwand bei der Ermittlung korrespondierender Merkmale dar. Es ist sinnvoll, den Kreis der zu prüfenden Merkmale einzuschränken, sofern das mit weniger aufwändigen Berechnungen realisierbar ist. Eine

7 7 7 7 7 7 1 1 1 1 1 7 7 7 7 7 7	7 7 7 7 7 7 1 1 1 0 0 7 7 7 7 7 7
7 7 7 7 1 1 1 1 1 1 1 1 1 7 7 7 7	7 7 7 7 1 1 1 1 1 1 0 0 0 7 7 7 7
7 7 2 2 2 1 1 1 1 1 1 1 0 0 0 7 7	7 7 1 1 1 1 1 1 1 1 0 0 0 0 0 0 7 7
7 7 2 2 2 2 1 1 1 1 1 1 0 0 0 0 7 7	7 7 1 1 1 1 1 1 1 1 0 0 0 0 0 0 7 7
7 2 2 2 2 2 1 1 1 1 1 1 0 0 0 0 0 7	7 2 2 1 1 1 1 1 1 1 0 0 0 0 0 0 0 7
7 2 2 2 2 2 2 1 1 1 1 0 0 0 0 0 0 7	7 2 2 2 1 1 1 1 1 1 0 0 0 0 0 0 5 7
2 2 2 2 2 2 2 1 1 1 1 0 0 0 0 0 0 0	2 2 2 2 2 2 1 1 1 1 0 0 0 0 5 5 5 5
2 2 2 2 2 2 2 2 1 1 0 0 0 0 0 0 0 0	2 2 2 2 2 2 2 2 1 1 1 0 0 5 5 5 5 5 5
3 3 3 3 3 3 3 3 6 0 0 0 0 0 0 0 0 0	2 2 2 2 2 2 2 2 2 6 5 5 5 5 5 5 5 5
3 3 3 3 3 3 3 3 4 5 5 5 5 5 5 5 5 5	2 2 2 2 2 2 2 3 3 4 4 5 5 5 5 5 5 5
3 3 3 3 3 3 3 4 4 4 5 5 5 5 5 5 5 5	2 2 2 2 3 3 3 3 4 4 4 5 5 5 5 5 5 5
7 3 3 3 3 3 3 4 4 4 5 5 5 5 5 5 5 7	7 2 3 3 3 3 3 3 4 4 4 4 4 5 5 5 7
7 3 3 3 3 3 4 4 4 4 4 5 5 5 5 5 7 7	7 3 3 3 3 3 3 3 4 4 4 4 4 4 5 5 7
7 7 3 3 3 3 4 4 4 4 4 4 5 5 5 5 7 7	7 7 3 3 3 3 3 3 4 4 4 4 4 4 4 4 7 7
7 7 3 3 3 4 4 4 4 4 4 4 4 5 5 5 7 7	7 7 3 3 3 3 3 3 4 4 4 4 4 4 4 4 7 7
7 7 7 7 4 4 4 4 4 4 4 4 4 4 7 7 7 7	7 7 7 7 3 3 3 3 4 4 4 4 4 7 7 7 7
7 7 7 7 7 7 4 4 4 4 4 7 7 7 7 7 7	7 7 7 7 7 7 3 3 4 4 4 7 7 7 7 7 7

Bild 3.5: Zwei Tabellen mit Indizes zur Erstellung von Deskriptoren. Die Zahlen geben den Index des Gradientenhistogramms an, zu dem die Werte an den entsprechenden Positionen beitragen. In diesem Beispiel wurde ein Kreisdeskriptor mit Radius 8, 6 Kreisausschnitten und nur einem Ring gewählt. Links ist die Tabelle für Punktmerkmale mit einer Orientierung zwischen 0° und 5 ° abgebildet, rechts die Tabelle für solche mit einer Orientierung zwischen 25° und 30°.

einfache und effektive Methode für Verfahren, bei denen nur geringe Änderungen zwischen aufeinander folgenden Bildern zu erwarten sind, ist die Festlegung eines Suchfensters. Überschreitet der Abstand zwischen den Bildpositionen einen Schwellwert, wird das Paar nicht weiter untersucht. Für die Berechnung des Ortsabstandes sind lediglich zwei Multiplikationen und eine Quadratwurzel notwendig. Alternativ kann man sogar als Annäherung die Differenzen der Spalten- und Reihenpositionen ohne Berechnung der Wurzel direkt gegen einen entsprechenden Schwellwert prüfen.

Die Berechnung des Deskriptorabstands erfordert die laufende Summierung quadrierter Einzeldifferenzen. Das Ergebnis ist die Wurzel dieser Summe. Da die Wurzelfunktion stetig ist, lassen sich auch die Zwischenwerte bei der Aufsummierung sofort gegen das Quadrat des Schwellwerts vergleichen. Sobald ein Zwischenwert das Quadrat des Schwellwerts übersteigt, kann auch der Gesamtabstand nicht mehr kleiner als der Schwellwert

```
Berechne den Index der korrekten Tabelle
Für alle Zeilen des Fensters i:
    Für alle Spalten des Fensters j:
        Lese den Index h an der Tabellenposition (i,j)
        Berechne die Urne u aus dem Gradienten an der Stelle (i,j)
        Addiere die Steigung an der Stelle (i,j) in die Urne u \
             des Histogramms mit dem Index h
```

Codefragment 3.1: Pseudocode zur Erstellung eines Deskriptors. Die Normalisierung des Deskriptors ist nicht aufgeführt.

sein. Dann kann die Berechnung je nach Anwendungskontext abgebrochen worden. Diese alternative Berechnung wurde unter dem Namen `fastDescriptorDistance` als Methode in die entsprechenden Klassen aufgenommen, die in Kapitel 4 besprochen werden.

3.5 Grundschema der Erweiterten Kalman Filter

In den Kapiteln 3.6 und 3.7 werden die System- und Messwertmodelle der Kalman Filter für die Initialisierung von Markern sowie die Schätzung der Kameraposition beschrieben. Da das Basismodell des Erweiterten Kalman Filters für beide Instanzen gilt, werden die zentralen Gleichungen hier vorab aufgeführt. Die konkrete Belegung der Funktionen und Matrizen unterscheidet sich natürlich bei den Modellen und wird in den darauf folgenden Kapiteln separat behandelt.

Die Gleichungen werden hier nur kurz aufgeführt. Eine gut verständliche Einführung in das Thema bietet [WB04], als detailliertes Standardwerk empfiehlt sich dagegen [May79]. Sehr gut sind auch die Ausführungen in dem Buch von Thrun [TBF05, Kap. 3]. Die verwendeten Symbole unterscheiden sich leider in der Literatur; für diese Arbeit wurden die Bezeichner wie in [WB04] gewählt.

Beiden Erweiterten Kalman Filtern ist gemeinsam, dass die Systemübergangsgleichung linear ist, die Messwertgleichung jedoch nicht. Daher können die Vorhersagen wie beim klassischen Kalman Filter direkt mit der Systemübergangsmatrix $\boldsymbol{A}$ berechnet werden,

während die Aktualisierung von Systemzustand und -varianz mit einer linearen Näherung der Messwertgleichung $\boldsymbol{h}$ durch ihre Jacobi-Matrix $\boldsymbol{H}$ erfolgen muss. $\boldsymbol{H}$ wird jeweils im Punkt der besten aktuellen Schätzung des Systemzustandes gebildet und in jeder Iteration neu berechnet. Deshalb erfolgt die Notation hier als Funktion des Systemzustandes $\boldsymbol{H}(\boldsymbol{x})$.

In den Gleichungen steht ein Strich über dem Bezeichner (z. B. $\bar{\boldsymbol{x}}$) für einen *a priori* Wert, also einem Ergebnis der Prädiktion. Die Systemübergangsmatrix sowie die Kovarianzen $\boldsymbol{R}$ und $\boldsymbol{Q}$ werden analog zum Systemzustand mit einem Index versehen, der den Zeitschritt andeutet ($\boldsymbol{A}_t$). Diese Indizierung wurde gewählt, weil die Matrizen teilweise dynamisch in jeder Iteration angepasst werden.

Jede Iteration der Kalman Filter lässt sich in fünf Schritte unterteilen, die nun einzeln aufgeführt werden.

1. Zu Beginn jeder Iteration steht die Prädiktion des wahrscheinlichsten nächsten Systemzustandes auf Basis der besten aktuellen Schätzung mithilfe der Systemübergangsgleichung:

$$\bar{\boldsymbol{x}}_{t+1} = \boldsymbol{A}_{t+1}\boldsymbol{x}_t \tag{3.2}$$

   Manchmal wird an dieser Stelle zusätzlich ein Kontrollvektor $\boldsymbol{B}_t\boldsymbol{u}_t$ für Eingabeparameter modelliert, der hier aber entfällt.

2. Anschließend erfolgt die Prädiktion der Unsicherheit dieser Vorhersage:

$$\bar{\boldsymbol{P}}_{t+1} = \boldsymbol{A}_{t+1}\boldsymbol{P}_t\boldsymbol{A}_{t+1}^T + \boldsymbol{Q}_{t+1} \tag{3.3}$$

   $\boldsymbol{Q}$ ist die Kovarianz des normalverteilten Rauschens im Systemmodell, die meist konstant bleibt, aber auch pro Iteration dynamisch angepasst werden kann. $\boldsymbol{P}$ ist die Systemkovarianz.

3. Basierend auf der Unsicherheit der Vorhersage lässt sich die Gain-Matrix berechnen, welche die Gewichtung der Messung bei der nächsten Aktualisierung herstellt:

$$\boldsymbol{K}_{t+1} = \bar{\boldsymbol{P}}_{t+1}\boldsymbol{H}(\bar{\boldsymbol{x}}_{t+1})^T\left(\boldsymbol{H}(\bar{\boldsymbol{x}}_{t+1})\bar{\boldsymbol{P}}_{t+1}\boldsymbol{H}(\bar{\boldsymbol{x}}_{t+1})^T + \boldsymbol{R}_{t+1}\right)^{-1} \tag{3.4}$$

   Die Gewichtung wird bestimmt durch das Verhältnis zwischen der Unsicherheit der Vorhersage und der Kovarianz des Messwertrauschens ($\boldsymbol{R}$). Die Matrix $\boldsymbol{R}$ kann,

so wie $\boldsymbol{Q}$ im vorangehenden Schritt, ebenfalls pro Iteration unterschiedliche Werte zugewiesen bekommen, obwohl sie in vielen Implementierungen konstant ist.

4. Mit Hilfe von $\boldsymbol{K}$ erfolgt die Aktualisierung des Systemzustandes mit der neuen Messung $\boldsymbol{z}_{t+1}$:

$$\boldsymbol{x}_{t+1} = \bar{\boldsymbol{x}}_{t+1} + \boldsymbol{K}_{t+1} \underbrace{(\boldsymbol{z}_{t+1} - \boldsymbol{h}(\bar{\boldsymbol{x}}_{t+1}))}_{\text{Innovation}} \tag{3.5}$$

   Die Innovation ist die im Darstellungsraum der Messwerte gebildete Differenz zwischen vorhergesagtem Systemzustand und den tatsächlichen neuen Messungen.

5. Analog dazu wird auch die Systemkovarianz aktualisiert, welche nun die Unsicherheit über den neuen Systemzustand darstellt:

$$\boldsymbol{P}_{t+1} = (\boldsymbol{I} - \boldsymbol{K}_{t+1}\boldsymbol{H}(\bar{\boldsymbol{x}}_{t+1}))\bar{\boldsymbol{P}}_{t+1} \tag{3.6}$$

3.6 Erzeugung einer neuen Karte

In Abschnitt 2.4 wurde ein Verfahren zur Initialisierung von 3D-Merkmalen [MCD06] vorgestellt, das deren Position im Raum durch inverse Tiefe auf einem Strahl parametrisiert. Verschiedene Vorteile dieser Technik für den probabilistischen Ansatz, der hier entwickelt wird, wurden bereits in Abschnitt 3.2.1 genannt.

Der wichtigste Vorteil ist, dass auch Punkte im Unendlichen mit dieser Parametrisierung eine klare Repräsentation haben, die sich formell nicht von der “normaler” Punkte unterscheidet. Es ist bekannt, dass ein Punkt im Unendlichen zwar keine Information zur Rekonstruktion der Tiefe, wohl aber Information bezüglich der Kameraorientierung liefern kann.[1] Die Parametrisierung mit inverser Tiefe erlaubt es, genau diesen Sachverhalt abzubilden, indem die Tiefe mit hoher und die Richtung mit geringer Varianz belegt werden kann. Wenn sich die Kamera über mehrere Bilder hinweg dem Punkt weiter nähert, ist es möglich, dass später eine Parallaxe entsteht - der Punkt würde also aus dem Unendlichen

[1] Diese Tatsache wird in [MD06] genutzt, um einen bildbasierten Kompass zu entwerfen.

in eine reelle Tiefe wandern. Dieser Prozess lässt sich fließend abbilden und ermöglicht die dauerhafte Beobachtung weit entfernter Punkte.

Die sinnvolle Verarbeitung weit entfernter Punkte ist ein Vorteil, der genau der Problemstellung dieser Arbeit zu Gute kommt: Das beschriebene System soll langfristig für den Außeneinsatz tauglich sein. Hier werden vermehrt größere Distanzen in den Bildern enthalten sein als bei den besser erforschten Innenraum-Szenarios. Die Merkmale mit größeren Distanzen werden sogar eine besonders wichtige Rolle spielen, da sie mit hoher Wahrscheinlichkeit über längere Bildsequenzen hinweg sichtbar bleiben als nahe gelegene Merkmale. Sie bilden also die Substanz natürlicher Marker, die selten reinitialisiert werden müssen und daher zur Stabilität der Karte beitragen.

Die Implementierung der nun folgenden Herleitungen in einer entsprechenden Klasse wird in Abschnitt 4.4 beschrieben.

3.6.1 Initiale Kameraposition durch künstliche Marker

Für die initiale Erzeugung natürlicher Marker beim Start des Systems muss eine Schätzung der Kameraposition bekannt sein. Wenn das gleiche Verfahren später zur Reinitialisierung einzelner Marker eingesetzt wird (siehe Abschnitt 3.8), kann diese Schätzung als Systemzustand der parallel laufenden Kalman Filter Instanz abgegriffen werden. Die Neuerstellung einer Karte findet aber statt, wenn das System startet oder die Schätzung der Kameraposition instabil geworden ist. In diesem Zustand sind als einzige verlässliche Daten die intrinsischen Parameter der Kamera vorhanden. Ohne weiteres Zusatzwissen kann damit eine Rekonstruktion der 3D-Struktur nur bis auf eine unbekannte Skalierung erfolgen ([TV98, Abschnitt 7.4]). Für den Einsatz in einem Augmented Reality System ist es aber erforderlich, dass die Rekonstruktion Werte mit festen metrischen Größen liefert.

Da für das Verfahren keine bekannten Objektmodelle herangezogen werden sollen, die eine metrische Festlegung ermöglichen, wird das System deshalb zu Beginn die Kameraposition mittels künstlicher Marker bestimmen. Dazu wurden Schnittstellen zu den Bibliotheken *ARToolkit* und *ARToolkitPlus* entwickelt, die in Abschnitt 4 erläutert werden. Experimente haben ergeben, dass dabei *ARToolkitPlus* stabilere Schätzungen der Kameraposition liefert und kompaktere Marker erlaubt. Im Gegensatz zu der Positionsschätzung

Bild 3.6: Kameraaufnahme einer Innenszene mit *ARToolkitPlus* Marker.

in *ARToolkit* wurde bei *ARToolkitPlus* das *Robust Planar Pose* Verfahren [SP05] implementiert. Die Marker sind in beiden Fällen quadratische Schwarz-/Weiß-Muster, die auf ein Blatt Papier gedruckt und auf einer ebenen Fläche in der Szene angebracht werden. Ein Bild mit *ARToolkitPlus* Marker, das von einer Webcam aufgezeichnet wurde, zeigt Abbildung 3.6.

ARToolkitPlus liefert bei Auffinden eines Markers eine *Modelview*-Matrix in einem direkt für *OpenGL* verwendbaren, nach Spalten geordneten Format der Form

$$M_{\textit{ARToolkitPlus}} = \begin{pmatrix} r_{11} & r_{21} & r_{31} & 0 \\ r_{12} & r_{22} & r_{32} & 0 \\ r_{13} & r_{23} & r_{33} & 0 \\ t_x & t_y & t_z & 1 \end{pmatrix}$$

Diese Matrix wird transponiert, um eine nach Zeilen geordnete Matrix mit Rotation und Translation der Kamera relativ zu dem detektierten Marker zu erhalten. Mit der transponierten Matrix lassen sich sofort Punkte ins Bild projizieren, wenn man von links mit einer Matrix multipliziert, welche die intrinsischen Parameter der Kamera enthält. Die Umrechnung in Orientierung und Position der Kamera im Weltkoordinatensystem erfolgt wie in Abschnitt 2.1 beschrieben.

Die Daten aus *ARToolkitPlus* definieren ein Rechte-Hand-Weltkoordinatensystem, bei dem die z-Achse senkrecht auf dem Marker steht und nach vorne zeigt, während x- und y-Achse genau in der 2D-Ebene des Markers liegen (Bild 3.7). Die Kamera ist ebenfalls in

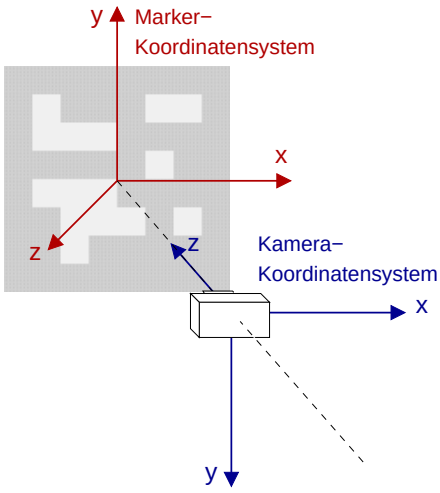


Bild 3.7: Implizite Koordinatensysteme der Modelview-Matrix von *ARToolkitPlus* .

einem Rechtssystem gegeben, bei dem die z-Achse in der optischen Achse liegt und ins Bild zeigt, so dass positive z-Werte vor der Kamera liegen. Die y-Achse zeigt senkrecht zur Blickachse nach unten und die x-Achse folglich nach rechts. Dies hat den Vorteil, dass die y-Werte nicht gekippt werden müssen, wenn die übliche Zeilensortierung digitaler Bilder vorliegt, welche zum Index 0 die obere Zeile liefert. Einige Programme liefern eine umgekehrte Zeilensortierung nach dem Einlesen von Bilddaten. Dazu gehören zum Beispiel einige Funktionen der *OpenCV* Bibliothek, wenn diese unter Windows gelinkt wurde. In solchen Fällen muss man sehr vorsichtig sein, da dann Linke-Hand-Systeme entstehen. Die in Abschnitt 4.2.1 beschriebenen Klassen zur Kapselung dieser Funktionen nehmen entsprechende Konvertierungen automatisch vor.

Um ein beliebiges Weltkoordinatensystem zu verwenden, muss die Transformation aus dem Welt- in das Markerkoordinatensystem von *ARToolkitPlus* berechnet werden. Diese ist entweder in Form der Rotation und Translation von links oder in Form der Orientierung und Position von rechts mit den weiteren Transformationen zu multiplizieren.

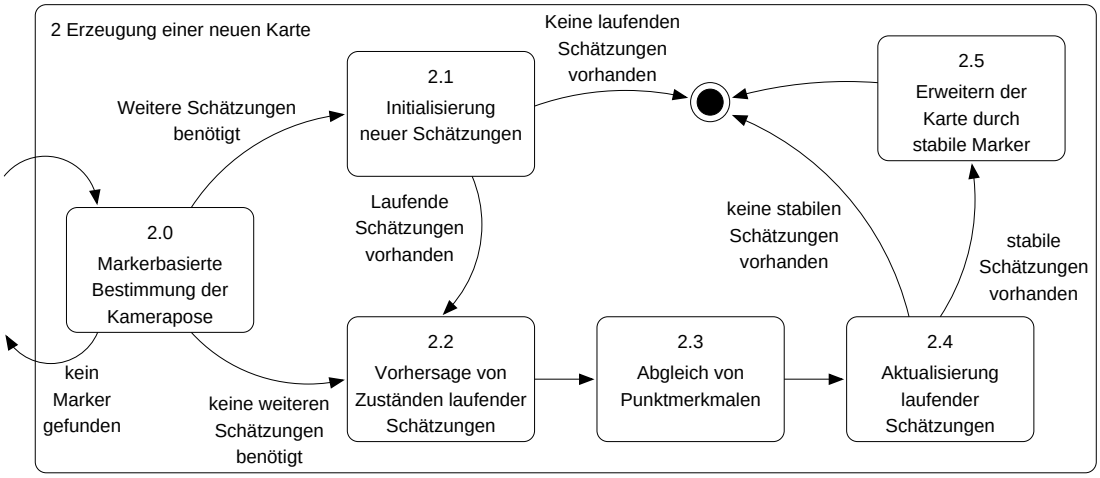


Bild 3.8: Verfeinerung des Prozesses 2 “Erzeugung einer neuen Karte”.

3.6.2 Ablauf der Neuerstellung

Die Verfeinerung des Prozesses zur Erzeugung einer neuen Karte zeigt das Zustands-Übergangs-Diagramm in Bild 3.8. Aus dem Prozess 0 “Verarbeitung der Sensordaten” (siehe Bild 3.2) steht bereits ein aktuelles Bild aus dem Datenstrom der Kamera sowie eine Menge von SIFT-Merkmalen, die aus diesem Bild extrahiert wurden, zur Verfügung. Zunächst wird versucht, mit der Bibliothek *ARToolkitPlus* einen gültigen Marker im Bild zu detektieren und damit eine direkte Schätzung der Kameraposition zu erhalten. Falls kein Marker gefunden werden kann, wird der Prozess sofort beendet und die nächste Iteration eingeleitet. Falls noch keine oder nicht genügend laufende Schätzungen von 3D-Markern vorhanden sind, müssen neue Schätzungen initialisiert werden (Teilprozess 2.1). Eine neue Schätzung für einen potentiellen 3D-Marker wird anhand der aktuellen Kameraposition im Weltkoordinatensystem und je einer 2D-Bildposition eines SIFT-Merkmals erzeugt, indem eine sinnvolle initiale Belegung der Parameter für die inverse Tiefe auf dem Sichtstrahl berechnet wird. Diese Initialisierung wird im nächsten Abschnitt im Detail beschrieben.

Mit den implementierten Softwarebausteinen ist es sowohl möglich, alle Schätzungen in einer Kalman Filter Instanz zu verwalten, als auch für jede Schätzung eine separate Instanz anzulegen. Im Folgenden werden diese unterschiedlichen Methoden als *Einzel-Instanz-*

und *Multi-Instanz-Methode* bezeichnet. Die Verwaltung in einer einzigen Filterinstanz führt insbesondere dazu, dass für n Marker eine gemeinsame Systemkovarianzmatrix der Größe $6n \times 6n$ geführt wird, welche sämtliche Korrelationen zwischen den Komponenten der einzelnen Marker abbilden kann. Damit lässt sich etwa erkennen, wenn zwei Merkmale in einer Ebene der Szene liegen und deshalb korrelieren. Allerdings erhöht sich der Berechnungsaufwand pro Iteration deutlich: Die Gesamtzahl an Koeffizienten bei der Einzel-Instanz-Methode beträgt das Quadrat der Anzahl an Koeffizienten bei der Multi-Instanz-Methode. Für eine Matrixmultiplikation sind bei der Variante mit getrennten Instanzen $6^2 \cdot n$ Einzelmultiplikationen nötig, während bei der Einzel-Instanz-Methode $(6 \cdot n)^2$ berechnet werden. Es sind pro Kalman Iteration natürlich mehrere Matrixmultiplikationen notwendig. Die Modellierung der vollen Kovarianz durch die Einzel-Instanz-Methode erfordert also einen Wechsel von linearem zu quadratischem Aufwand bezüglich der Anzahl von Markern. Darüber hinaus ist die dynamische Skalierung der Kalman Filter Matrizen bei häufigem Hinzufügen und Entfernen von Markern sehr aufwändig, wie im Kontext der Implementierung in Abschnitt 4.4 noch erläutert wird. Entscheidend ist also der mögliche Vorteil durch die zusätzlich verfügbaren Korrelationen, der im Experiment festgestellt werden kann.

Ab der zweiten Iteration sind laufende Schätzungen vorhanden, die aufgrund von SIFT-Merkmalen aus dem nächsten Bild aktualisiert werden können. In diesem Fall wird eine Iteration des Erweiterten Kalman Filters durchlaufen, mit dem die Schätzung realisiert wird. Nachdem die Schätzung aufgrund der neu eingelesenen Kameraposition vorhergesagt wurde (Teilprozess 2.2), werden in der Liste neuer SIFT-Merkmale Korrespondenzen zu den laufenden Schätzungen gesucht. Hierzu ist es erforderlich, die Deskriptoren der Merkmale bei der Initialisierung zusammen mit den Schätzungen abzuspeichern. Sofern eine Korrespondenz zu einer laufenden Schätzung ermittelt wurde, kann mit der Bildposition dieser neuen Beobachtung ein Aktualisierungsschritt des Kalman Filters durchgeführt werden (Teilprozess 2.4). Hier kann die Schätzung drastisch verbessert werden, wenn es gelingt, die Qualität der Korrespondenz als Verlässlichkeit der Messung mit einfließen zu lassen.

Nach Durchlaufen mehrerer Zyklen sollten sich erste Schätzungen als stabil erweisen. In diesem Fall können die 6-dimensionalen Repräsentationen mit inverser Tiefe in einfache

3D-euklidische Repräsentationen umgerechnet und als natürliche Marker in die Karte eingefügt werden (Teilprozess 2.5). Die Stabilität einer Schätzung lässt sich auf verschiedene Weisen feststellen: Durch das Alter der Schätzung, durch die Höhe der zugehörigen Kovarianzen, und durch die Kontinuität aufeinander folgender Schätzungen. Hier sind umfangreiche Experimente notwendig, da entsprechende Schwellwerte schwer festzulegen sind. Erste Ergebnisse werden in Abschnitt 5.2 beschrieben.

3.6.3 Initialisierung nach erster Beobachtung

Im Gegensatz zu den Ausführungen in [MCD06] wird hier die Orientierung der Kamera nicht als unsicherer Parameter des Systemzustandes, sondern als Konstante behandelt. Die Kameraposition hingegen bleibt Teil des Systemzustandes, da sie den Ursprung des Strahls darstellt. So hat der Erweiterte Kalman Filter die Möglichkeit, auch die Strahlrichtung zu justieren.

Die Beobachtung eines Punktmerkmals definiert eine Einschränkung zwischen der zugehörigen Position im Weltkoordinatensystem und der Kameraposition, die aus dem vorangehenden Teilprozess als Schätzung vorliegt. Die Initialisierung der Schätzung nach einer solchen Beobachtung lässt sich als Funktion $\hat{\boldsymbol{y}} : \mathbb{R}^6 \rightarrow \mathbb{R}^6$ formulieren:

$$\hat{\boldsymbol{y}}(\hat{\boldsymbol{r}}, \mu, \nu, \rho_{min}) = (\hat{r}_x, \hat{r}_y, \hat{r}_z, \hat{\theta}, \hat{\phi}, \hat{\rho})$$

Hierbei bezeichnet $\hat{\boldsymbol{r}}$ die geschätzte Position der Kamera im Weltkoordinatensystem, $\hat{\theta}$ und $\hat{\phi}$ den Dreh- und Kippwinkel der Halbgeraden und $\hat{\rho}$ die inverse Tiefe der geschätzten 3D-Position auf dem Strahl. Das Dachsymbol deutet an, dass es sich um Schätzwerte handelt. Die Herleitung der Funktion $\hat{\boldsymbol{y}}$ lehnt sich eng an [MCD06] an, wird im Folgenden jedoch stärker konkretisiert.

Zunächst lässt sich ein Vektor $\boldsymbol{h}^C$ bestimmen, der im Kamerakoordinatensystem vom optischen Zentrum in Richtung des beobachteten Punktes zeigt. Mit der Bildposition (μ, ν) erhält man diesen Vektor, indem man die Tiefe 1 annimmt. Dazu setzt man den auf das Kamerakoordinatensystem bezogenen Teil der Projektionsgleichung (2.3) in (2.5) ein:

$$\mu = \frac{p_x^C \cdot f_x + p_z^C \cdot c_x}{p_z^C} \qquad \nu = \frac{p_y^C \cdot f_y + p_z^C \cdot c_y}{p_z^C}$$

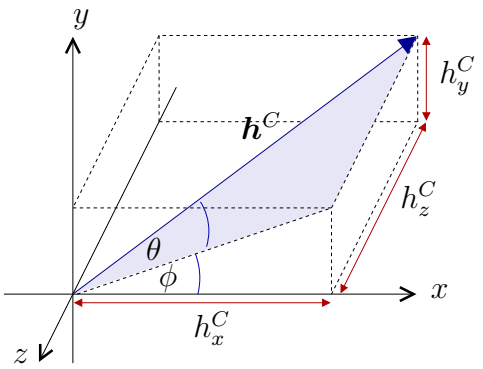


Bild 3.9: Ermittlung von Drehwinkel θ und Kippwinkel ϕ zu einem Vektor $\boldsymbol{h}$.

Mit $p_z^C = 1$ erhält man den Vektor in Kamerakoordinaten

$$\boldsymbol{h}^C = \begin{pmatrix} \frac{\mu - c_x}{f_x} \\ \frac{\nu - c_y}{f_y} \\ 1 \end{pmatrix} \quad .$$

Dieser Vektor muss in das Weltkoordinatensystem umgerechnet werden. Da es sich um einen Richtungsvektor handelt, ist die Translation unerheblich und es genügt

$$\boldsymbol{h}^W = \begin{pmatrix} h_x^W \\ h_y^W \\ h_z^W \end{pmatrix} = \boldsymbol{R}^T \cdot \boldsymbol{h}^C = \begin{pmatrix} \frac{r_{11}(\mu - c_x)}{f_x} + \frac{r_{21}(\nu - c_y)}{f_y} + r_{31} \\ \frac{r_{12}(\mu - c_x)}{f_x} + \frac{r_{22}(\nu - c_y)}{f_y} + r_{32} \\ \frac{r_{13}(\mu - c_x)}{f_x} + \frac{r_{23}(\nu - c_y)}{f_y} + r_{33} \end{pmatrix} \quad .$$

Im Gegensatz zu dem Verfahren in [MCD06] wird in dieser Arbeit die Radialverzerrung der Kameralinse nicht modelliert, wodurch sich das Verfahren vereinfacht. Am Fraunhofer Institut für angewandte Informationstechnik liegen Erfahrungen vor, dass die Radialverzerrung der verwendeten Kamera nicht sehr hoch ist und zunächst vernachlässigt werden kann.

Zur Parametrisierung von $\boldsymbol{h}^W$ in Kippwinkel ϕ und Drehwinkel θ muss eine Umrechnung mit dem Arcustangens erfolgen. Zur Veranschaulichung der Berechnung sind die entspre-

chenden rechtwinkligen Dreiecke in Bild 3.9 eingezeichnet. [2]

$$\begin{pmatrix} \theta_i \\ \phi_i \end{pmatrix} = \begin{pmatrix} \arctan\left(h_z^W, \sqrt{h_x^{W2} + h_y^{W2}}\right) \\ \arctan\left(h_y^W, h_x^W\right) \end{pmatrix} \tag{3.7}$$

Der Wert für den fehlenden sechsten Parameter der Schätzung, der die Tiefe auf dem Strahl codiert, wird schließlich auf die Hälfte des gültigen Bereichs festgelegt. Der gültige Bereich reicht vom Punkt im Unendlichen bis zur minimalen Distanz d_{min}. Als inverse Tiefe ausgedrückt, erstreckt sich der Bereich also von $\rho_\infty = 0$ bis $\rho_{min} = 1/d_{min}$. Die Hälfte des gültigen Bereiches liegt damit bei $\hat{\rho}_0 = \rho_{min}/2$. Damit steht die zu Beginn dieses Abschnitts genannte multivariate Funktion $\hat{\boldsymbol{y}}$ fest:

$$\hat{\boldsymbol{y}} \begin{pmatrix} \hat{r}_x \\ \hat{r}_y \\ \hat{r}_z \\ \mu \\ \nu \\ \rho_{min} \end{pmatrix} = \begin{pmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \hat{y}_3 \\ \hat{y}_4 \\ \hat{y}_5 \\ \hat{y}_6 \end{pmatrix} \begin{pmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \\ \mu \\ \nu \\ \rho_{min} \end{pmatrix} = \begin{pmatrix} \hat{r}_x \\ \hat{r}_y \\ \hat{r}_z \\ \arctan\left(h_z^W, \sqrt{h_x^{W2} + h_y^{W2}}\right) \\ \arctan\left(h_y^W, h_x^W\right) \\ \hat{\rho}_0 \end{pmatrix} \tag{3.8}$$

Bei der Initialisierung eines Markers ist es wichtig, die Werte in der Systemkovarianzmatrix des Filters richtig zu setzen. Die unkorrelierten Grundvarianzen für die sechs Dimensionen werden der Kovarianz des Messwertrauschens, $\boldsymbol{R}$, entnommen. Lediglich der sechste Parameter, die inverse Tiefe ρ, erhält einen abweichenden Wert: Die Unsicherheit nach der initialen Schätzung in Bezug auf die Tiefe erstreckt sich nämlich über den gesamten gültigen Bereich der Halbgeraden. Da die Unsicherheiten im Kontext des Kalman Filters als Gaussverteilungen modelliert sind, kann man sich diese als Gausskurve vorstellen, deren Mittelwert in der Mitte des gültigen Bereichs liegt, und deren Varianz auf der einen Seite bis zur minimalen Tiefe, auf der anderen Seite bis zum Punkt im Unendlichen reicht. Daher wird die Grundvarianz der sechsten Komponente mit der Hälfte der geschätzten inversen Tiefe initialisiert, d. h. mit $\rho_{min}/4$.

[2] In [MCD06] ist für die Berechnung von θ die Formel $\arctan(-h_z, \sqrt{h_x^2 + h_z^2})$ angegeben. In dem Dreieck, das durch h_x und h_z aufgespannt würde, liegt der rechte Winkel jedoch so, dass der Arcustangens nicht in dieser Form angewendet werden kann.

Auf diese Weise entsteht eine Diagonalmatrix, welche die Korrelationen zwischen den einzelnen Parametern nicht berücksichtigt. Insbesondere der Kipp- und Drehwinkel sind jedoch stark voneinander abhängig. Die neue Kovarianzmatrix wird daher von links und rechts mit der Jacobi-Matrix der Funktion $\hat{\boldsymbol{y}}$ multipliziert, d.h mit einer weiteren 6×6-Matrix der Form

$$\boldsymbol{J} = \begin{pmatrix} \frac{\partial \hat{\boldsymbol{y}}}{\partial \hat{r}_x} & \frac{\partial \hat{\boldsymbol{y}}}{\partial \hat{r}_y} & \frac{\partial \hat{\boldsymbol{y}}}{\partial \hat{r}_z} & \frac{\partial \hat{\boldsymbol{y}}}{\partial \mu} & \frac{\partial \hat{\boldsymbol{y}}}{\partial \nu} & \frac{\partial \hat{\boldsymbol{y}}}{\partial \rho} \end{pmatrix} \quad .$$

Nach Betrachtung von (3.8) sieht man leicht, dass $\boldsymbol{J}$ nur vier komplexere Ableitungen enthält:

$$\boldsymbol{J} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{\partial \hat{y}_3}{\partial \hat{\mu}} & \frac{\partial \hat{y}_3}{\partial \hat{\nu}} & 0 \\ 0 & 0 & 0 & \frac{\partial \hat{y}_4}{\partial \hat{\mu}} & \frac{\partial \hat{y}_4}{\partial \hat{\nu}} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{2} \end{pmatrix}$$

Dabei gilt

$$\begin{aligned} \frac{\partial \hat{y}_3}{\partial \hat{\mu}} &= \frac{\frac{r_{31}}{f_x}\sqrt{h_x^{W^2} + h_y^{W^2}} \; - \; h_z^W \frac{h_x^W \frac{r_{11}}{f_x} + h_y^W \cdot \frac{r_{21}}{f_x}}{\sqrt{h_x^{W^2} + h_y^{W^2}}}}{h_x^{W^2} + h_y^{W^2} + h_z^{W^2}} \\ \frac{\partial \hat{y}_3}{\partial \hat{\nu}} &= \frac{\frac{r_{32}}{f_y}\sqrt{h_x^{W^2} + h_y^{W^2}} \; - \; h_z^W \frac{h_x^W \frac{r_{12}}{f_y} + h_y^W \frac{r_{22}}{f_y}}{\sqrt{h_x^{W^2} + h_y^{W^2}}}}{h_x^{W^2} + h_y^{W^2} + h_z^{W^2}} \\ \frac{\partial \hat{y}_4}{\partial \hat{\mu}} &= \frac{h_x^W \frac{r_{21}}{f_x} + h_y^W \frac{r_{11}}{f_x}}{h_x^{W^2} + h_y^{W^2}} \\ \frac{\partial \hat{y}_4}{\partial \hat{\nu}} &= \frac{h_x^W \frac{r_{22}}{f_y} + h_y^W \frac{r_{12}}{f_y}}{h_x^{W^2} + h_y^{W^2}} \quad . \end{aligned}$$

Bei Verwendung der Einzel-Instanz-Methode (Abschnitt 3.6.2) stellt sich zusätzlich die Frage, wie mit der Systemkovarianzmatrix beim Hinzufügen und Löschen neuer Schätzungen zu verfahren ist.

Zunächst wird der Fall auftreten, dass eine neu initialisierte Schätzung zu einem bestehenden Filter hinzugenommen wird. In diesem Fall sind die initialen Kovarianzen in den entsprechenden 6×6-Teil der Systemkovarianzmatrix einzutragen. Der restliche Teil der

Matrix sollte davon unberührt bleiben. Daraus ergibt sich ein praktisches Problem: Wendet man die Aktualisierung auf die a posteriori Varianz P an, so werden die Einträge überschrieben, falls sich der Filter gerade zwischen Vorhersage und Aktualisierung befindet. Wendet man sie auf die a priori Varianz $\bar{P}$ an, erhält man umgekehrt Seiteneffekte, falls der Filter zwischen Aktualisierung und Vorhersage steht. In dieser Arbeit wurde das Problem gelöst, indem das Hinzufügen natürlicher Marker nur nach dem vorangehenden Aktualisierungsschritt zu Beginn einer Iteration, also vor der Vorhersage, stattfinden darf. Dazu enthalten die implementierten Klassen ein Attribut für den Zustand, in dem sie sich befinden. Die Methode zum Hinzufügen eines Markers wird nur durchgeführt, wenn sich der Filter im korrekten Zustand befindet.

Außerdem muss eine Anpassung der Systemmatrizen erfolgen, wenn eine Schätzung aus dem Systemzustand zu löschen ist. Das würde gemäß Bild 3.8 in Teilprozess 2.4 erfolgen, falls beim Abgleich in Teilprozess 2.3 mehrfach keine Korrespondenzen gefunden wurden oder die Unsicherheit des Markers zu hoch wird. Die Einträge in der System-Kovarianzmatrix $\boldsymbol{P}$ sind in dem Fall nur noch teilweise gültig. Für einen Marker i, dessen Parameter sich im Systemzustand an der Stelle $6i$ bis $6i + 5$ befunden haben, enthalten alle Einträge in den Spalten $6i$ bis $6i + 5$ und den Reihen $6i$ bis $6i + 5$ korrelierte und unkorrelierte Information, die sich auf den Marker bezieht. Diese Einträge müssen streng genommen zurückgesetzt werden. Ebenso beziehen sich Einträge in der Systemrauschen-Kovarianzmatrix $\boldsymbol{Q}$ und der Messwertrauschen-Kovarianzmatrix $\boldsymbol{R}$ an entsprechenden Positionen auf die ungültige Schätzung. Wenn $\boldsymbol{R}$ und $\boldsymbol{Q}$ nicht dynamisch angepasst werden, ist hier jedoch keine Korrektur notwendig.

Eine entstandene “Lücke” in den Systemmatrizen bleibt jedoch ein Problem: Sie wird weiterhin durch die Kalman Iterationen verarbeitet und nimmt - wenn auch in geringem Maße - eine statistische Funktion ein, da sie Ressourcen zum Ausgleich von Korrelationen bereitstellt. Es sollte also vermieden werden, Schätzungen ersatzlos aus einer Filterinstanz zu löschen. Das lässt sich erreichen, indem bei Verwendung der Einzel-Instanz-Methode jede gelöschte Schätzung simultan durch eine Neu-Initialisierung ersetzt wird.

3.6.4 Linearisierung der Messwertgleichung

Das Systemmodell zur Schätzung der Position natürlicher Marker mittels inverser Tiefe beinhaltet zwei unterschiedliche Räume für die Repräsentation von Systemzuständen und Messwerten. Damit für den Kalman Filter die Innovation (siehe (3.5)), also die Differenz zwischen der Vorhersage des Systemzustands und der darauf folgenden tatsächlichen Messung, berechnet werden kann, ist eine Umrechnung der Zustände zwischen diesen beiden Räumen nötig. Die Umrechnung eines Systemzustands in einen Messwert ist durch die Messwertgleichung (3.8) gegeben. Zur Berechnung der *Gain*-Matrix ist jedoch die umgekehrte Berechnung in linearer Form nötig, obwohl die Messwertgleichung zwei nichtlineare Komponenten enthält. Aus diesem Grund ist für die Schätzung eine Linearisierung notwendig, die im Erweiterten Kalman Filter durch ihre erste Taylor-Näherung realisiert wird. Dazu ist die Jacobi-Matrix der Messwertgleichung zu bilden.

Zur Berechnung des Punktes in Weltkoordinaten ist zunächst die Umrechnung in eine 3D-Euler-Repräsentation nötig. In Gleichung (2.6) wurde diese Umrechnung unter Verwendung der Funktion $\boldsymbol{m}(\theta, \phi)$ angegeben, die nun ausnotiert werden muss. $\boldsymbol{m}$ stellt den Richtungsvektor dar, der durch Dreh- und Kippwinkel definiert ist. Für den Vektor ergibt sich bei Annahme der Länge 1

$$\boldsymbol{m}(\theta, \phi) = \begin{pmatrix} \cos\theta \cdot \cos\phi \\ \cos\theta \cdot \sin\phi \\ \sin\theta \end{pmatrix} \quad .$$

Somit erhält man aus der Strahlrepräsentation den Punkt in Weltkoordinaten

$$\boldsymbol{p}^W = \begin{pmatrix} p_x^W \\ p_y^W \\ p_z^W \end{pmatrix} = \begin{pmatrix} \hat{r}_x \\ \hat{r}_y \\ \hat{r}_z \end{pmatrix} + \frac{1}{\hat{\rho}} \boldsymbol{m}(\hat{\theta}, \hat{\phi}) = \begin{pmatrix} \hat{r}_x + \frac{1}{\hat{\rho}} \cdot \cos\hat{\theta} \cdot \cos\hat{\phi} \\ \hat{r}_y + \frac{1}{\hat{\rho}} \cdot \cos\hat{\theta} \cdot \sin\hat{\phi} \\ \hat{r}_z + \frac{1}{\hat{\rho}} \cdot \sin\hat{\theta} \end{pmatrix} \quad .$$

Die Messwertgleichung ist für n Schätzungen natürlicher Marker eine multivariate Funktion der Form $\boldsymbol{h} : \mathbb{R}^{6n} \rightarrow \mathbb{R}^{2n}$. Sie lässt sich notieren als Vektor von $2n$ nichtlinearen

Funktionen der Form $h_i : \mathbb{R}^{6n} \rightarrow \mathbb{R}$:

$$\boldsymbol{h}(\boldsymbol{x}) = \begin{pmatrix} h_{\mu_1}\begin{pmatrix} \hat{r}_{x_1} & \hat{r}_{y_1} & \hat{r}_{z_1} & \hat{\theta}_1 & \hat{\phi}_1 & \hat{\rho}_1 \end{pmatrix}^T \\ h_{\nu_1}\begin{pmatrix} \hat{r}_{x_1} & \hat{r}_{y_1} & \hat{r}_{z_1} & \hat{\theta}_1 & \hat{\phi}_1 & \hat{\rho}_1 \end{pmatrix}^T \\ \vdots \\ h_{\mu_n}\begin{pmatrix} \hat{r}_{x_n} & \hat{r}_{y_n} & \hat{r}_{z_n} & \hat{\theta}_n & \hat{\phi}_n & \hat{\rho}_n \end{pmatrix}^T \\ h_{\nu_n}\begin{pmatrix} \hat{r}_{x_n} & \hat{r}_{y_n} & \hat{r}_{z_n} & \hat{\theta}_n & \hat{\phi}_n & \hat{\rho}_n \end{pmatrix}^T \end{pmatrix}$$

Der Vektor enthält n Paare äquivalenter Gleichungen h_{μ_i} und h_{ν_i}, die jeweils die Projektion einer solchen Schätzung ins Bild darstellen. Jede Gleichung leitet sich aus der Projektion (2.5) ab, erfüllt also

$$\begin{aligned} h_{\mu_i} &= \frac{p_x^P}{p_z^P} = \frac{f_x p_x^C + c_x p_z^C}{p_z^C} \\ &= \frac{f_x(r_{11}p_x^W + r_{12}p_y^W + r_{13}p_z^W + t_x) + c_x(r_{31}p_x^W + r_{32}p_y^W + r_{33}p_z^W + t_z)}{r_{31}p_x^W + r_{32}p_y^W + r_{33}p_z^W + t_z} \\ h_{\nu_i} &= \frac{p_y^P}{p_z^P} = \frac{f_y p_y^C + c_y p_z^C}{p_z^C} \\ &= \frac{f_y(r_{21}p_x^W + r_{22}p_y^W + r_{23}p_z^W + t_y) + c_y(r_{31}p_x^W + r_{32}p_y^W + r_{33}p_z^W + t_z)}{r_{31}p_x^W + r_{32}p_y^W + r_{33}p_z^W + t_z} \end{aligned} .$$

Die Jacobi-Matrix hat die Form

$$\begin{pmatrix} \frac{\partial h_{\mu_1}}{\partial \boldsymbol{y}_1} & \frac{\partial h_{\mu_1}}{\partial \boldsymbol{y}_2} & \dots & \frac{\partial h_{\mu_1}}{\partial \boldsymbol{y}_n} \\ \frac{\partial h_{\nu_1}}{\partial \boldsymbol{y}_1} & \frac{\partial h_{\nu_1}}{\partial \boldsymbol{y}_2} & \dots & \frac{\partial h_{\nu_1}}{\partial \boldsymbol{y}_n} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial h_{\mu_n}}{\partial \boldsymbol{y}_1} & \frac{\partial h_{\mu_n}}{\partial \boldsymbol{y}_2} & \dots & \frac{\partial h_{\mu_n}}{\partial \boldsymbol{y}_n} \\ \frac{\partial h_{\nu_n}}{\partial \boldsymbol{y}_1} & \frac{\partial h_{\nu_n}}{\partial \boldsymbol{y}_2} & \dots & \frac{\partial h_{\nu_n}}{\partial \boldsymbol{y}_n} \end{pmatrix} = \begin{pmatrix} \frac{\partial h_{\mu_1}}{\partial \boldsymbol{y}_1} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \frac{\partial h_{\nu_1}}{\partial \boldsymbol{y}_1} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \frac{\partial h_{\mu_2}}{\partial \boldsymbol{y}_2} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \frac{\partial h_{\nu_2}}{\partial \boldsymbol{y}_2} & \mathbf{0} & \dots & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \frac{\partial h_{\mu_n}}{\partial \boldsymbol{y}_n} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \frac{\partial h_{\nu_n}}{\partial \boldsymbol{y}_n} \end{pmatrix} .$$

Es handelt sich letztlich um zwei mal sechs Ableitungen, die jeweils für μ und ν sowie pro Marker leicht voneinander abweichen. Die Ableitungen sind im Anhang in Abschnitt B.1 im Einzelnen aufgeführt.

Bei der Berechnung der Jacobi-Matrix kann ein Problem entstehen, wenn die geschätzte Position auf dem Strahl in das optische Zentrum der Kamera fällt, denn dann ist die Projektion nicht definiert. Konkret wird dieses Problem, wenn durch p_Z^C geteilt wird - dieser Wert beträgt dann Null. Dieser Effekt kann verschiedene Ursachen haben. Angenommen, die geschätzte Position des Markers im Weltkoordinatensystem sowie die geschätzte Position der Kamera wären annähernd korrekt. Dann kann mit der aktuellen Kameraposition keine Beobachtung des Punktes erfolgt sein, da er nicht sichtbar ist. In diesem Fall wäre also die Beobachtung, genauer gesagt die Zuordnung des SIFT-Merkmals, falsch.

Es könnte aber auch sein, dass die geschätzte Position des Markers im Weltkoordinatensystem falsch ist. In diesem Fall wäre es sinnvoll, vor dem Aktualisierungsschritt des Kalman Filters die Varianz des Systemrauschens $\boldsymbol{Q}$ sehr hoch, die Varianz des Messwertrauschens $\boldsymbol{R}$ dagegen sehr niedrig zu setzen. Auf diese Weise würde die Messung sehr viel stärker bei der Aktualisierung berücksichtigt als das Systemmodell.

Zuletzt könnte aber auch die geschätzte Position der Kamera stark von der realen Position abweichen. Dann müssten auch die Schätzungen weiterer Markerpositionen das gleiche Problem zeigen. Dieser Fall ist jedoch sehr unwahrscheinlich, da die Schätzung aus dem markerbasierten Tracking in *ARToolkitPlus* stammt.

Es ist schwierig, die genannten Fälle automatisch zu unterscheiden. Die Implementierungen, die für diese Arbeit vorgenommen wurden, reagieren mit einem kontrollierten Abbruch des Systems zur Laufzeit. Während der Experimente mit der Software ist dieser Fall bisher nicht aufgetreten.

3.7 Schätzung der Kameraposition

Wenn eine Karte mit einer ausreichenden Anzahl stabiler natürlicher Marker zur Verfügung steht, kann das System die Kamerabewegung ohne Verwendung künstlicher Marker rekonstruieren. In dem Fall wird der in Bild 3.2 rechts eingezeichnete Zyklus durchlaufen. Die Rekonstruktion wird durch den Prozess 3.1 “Schätzung der Kameraposition” realisiert, der in diesem Kapitel detailliert beschrieben wird. Die parallel laufende Reinitialisierung natürlicher Marker wird im folgenden Abschnitt 3.8 behandelt.

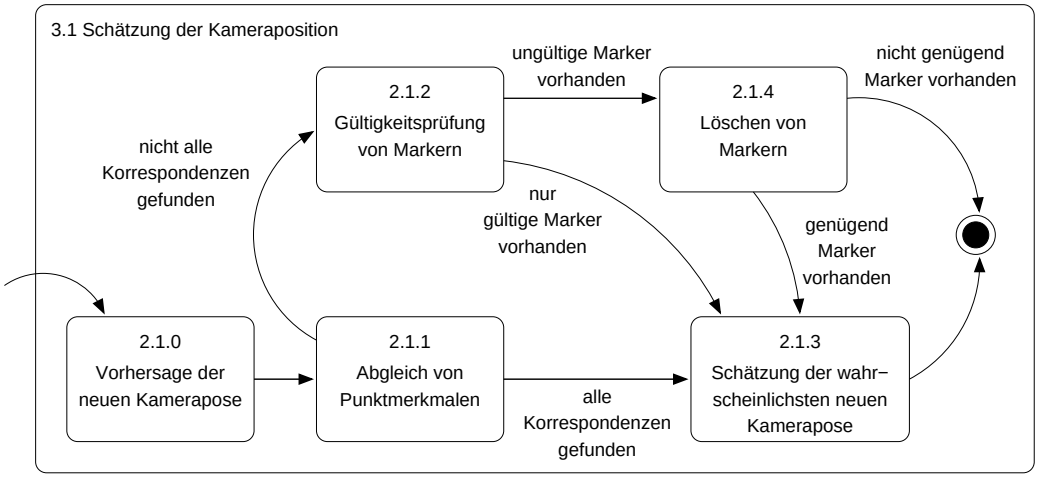


Bild 3.10: Verfeinerung des Prozesses “Schätzung der Kameraposition”.

Dem gesamten Prozess liegen zwei zentrale Annahmen zu Grunde. Zunächst wird vorausgesetzt, dass die natürlichen Marker in der Karte zu starren Objekten in der Welt gehören und demnach selbst keiner Eigenbewegung unterliegen. Es wird angenommen, dass jede Veränderung ihrer Projektion im Bild durch die Bewegung der Kamera verursacht ist. Marker, die diese Voraussetzung verletzen, weil sie beispielsweise auf einem beweglichen Gegenstand detektiert wurden, verursachen Fehler in der Rekonstruktion.

Als zweite Grundannahme wird vorausgesetzt, dass die Orientierung der Kamera zu jedem Zeitpunkt bekannt und korrekt ist. Diese Annahme wird erfüllt, wenn der verfügbare Inertialsensor, so wie im Datenblatt angegeben, stabile Daten liefert und diese durch Montage und präzise Kalibrierung fehlerfrei in das Kamerakoordinatensystem transformiert werden können. Erste Versuche und Erfahrungen mit dem *Xsens MTx* sind in Abschnitt 4.2.2 im Kontext der Implementierung aufgeführt.

Bild 3.10 liefert die verfeinerte Darstellung einer Iteration zur Schätzung der Kameraposition als Zustands-Übergangs-Diagramm. Wie auch in Prozess 2 “Erzeugung einer neuen Karte” wird hier implizit eine Iteration des zu Grunde liegenden Erweiterten Kalman Filters durchlaufen. Dabei erfolgt die Prädiktion in Teilprozess 2.1.0 und die Korrektur in Teilprozess 2.1.3. Diese beiden Verarbeitungsschritte ergeben sich direkt aus der Mo-

dellierung des Erweiterten Kalman Filters, der im folgenden Abschnitt 3.7.1 beschrieben wird.

Die übrigen Prozesse dienen dazu, die aus dem aktuellen Bild extrahierten Punktmerkmale abzugleichen und auf fehlende oder ungültige Korrespondenzen zu reagieren. Der Abgleich von Punktmerkmalen erfolgt grundsätzlich über den euklidischen Abstand der in Abschnitt 3.4.2 beschriebenen Deskriptoren. Den natürlichen Markern wird deshalb der Deskriptor des Punktmerkmals zugewiesen, aus dem sie initialisiert wurden. Der Abgleich wurde durch die Klasse `PointFeatureMatcher` implementiert, die in Abschnitt 4.3 vorgestellt wird. Es lassen sich zwei Strategien der Zuordnung wählen:

- Bei der *Zuordnung der besten Korrespondenz* wird zu einem Marker in der Karte das Punktmerkmal mit dem kürzesten euklidischen Deskriptorabstand zum Marker ausgewählt.
- Bei der *Zuordnung eindeutiger Korrespondenzen* wird ebenfalls das Merkmal mit dem kürzesten euklidischen Deskriptorabstand selektiert; jedoch nur dann, wenn der Abstand der potentiellen zweitbesten Korrespondenz deutlich schlechter war. In Experimenten hat es sich als brauchbare Grenze erwiesen, wenn der Deskriptorabstand der besten Zuordnung 90% oder weniger des Abstandes der zweitbesten Zuordnung beträgt.

3.7.1 Modellierung des Erweiterten Kalman Filters

Das Modell des Erweiterten Kalman Filters, das hier zur Schätzung der Kamerabewegung entwickelt wird, orientiert sich in vielen Punkten an dem in [DMM03] vorgestellten SLAM-Verfahren. Der Systemzustand hat für n Marker in der Karte die Dimension $6+3n$:

$$\boldsymbol{x} = \begin{pmatrix} \boldsymbol{x}_v & \boldsymbol{y}_1 & \dots & \boldsymbol{y}_n \end{pmatrix}^T = \begin{pmatrix} \boldsymbol{t} & \boldsymbol{v} & \boldsymbol{y}_1 & \dots & \boldsymbol{y}_n \end{pmatrix}^T$$

Hier bezeichnet $\boldsymbol{t} = \begin{pmatrix} t_x & t_y & t_z \end{pmatrix}$ die Translation der Kamera, $\boldsymbol{v} = \begin{pmatrix} v_x & v_y & v_z \end{pmatrix}$ die Richtungsgeschwindigkeit der Kamera und $\boldsymbol{y}_i = \begin{pmatrix} y_{i_x} & y_{i_y} & y_{i_z} \end{pmatrix}$ die Marker in 3D-euklidischer Repräsentation. Es wird bewusst die Translation, nicht die Position, modelliert, da auch für die Messwertgleichung die Translation herangezogen werden muss. Bei

Verwendung der Position wäre ansonsten in jeder Iteration eine Umrechnung gemäß 2.2 erforderlich.

Die Orientierung der Kamera ist nicht Teil des unsicheren Systemzustandes, da sie gemäß der oben getroffenen Annahmen bekannt und korrekt ist. Sie wird stattdessen in jeder Iteration direkt als Koeffizient der Messwertgleichung gesetzt. So kann gleichzeitig die Modellierung der Winkelgeschwindigkeit entfallen, die zur Abbildung von Veränderungen der Orientierung in der Systemübergangsgleichung notwendig wäre. Die Systemübergangsgleichung bleibt so linear und verbessert die Eigenschaften des Filters gegenüber einer nichtlinearen Variante.

Da die Markerpositionen laut Annahme konstant sind, enthält die Systemübergangsmatrix $\boldsymbol{A}$ nur in dem linken oberen 6×6-Ausschnitt relevante Einträge. Man kann daher eine Partitionierung vornehmen:

$$\boldsymbol{x}_{t+1} = \left(\begin{array}{c|ccc} \boldsymbol{A}_{\boldsymbol{x}_v\boldsymbol{x}_v} & \boldsymbol{A}_{\boldsymbol{x}_v\boldsymbol{y}_1} & \dots & \boldsymbol{A}_{\boldsymbol{x}_v\boldsymbol{y}_n} \\ \hline \boldsymbol{A}_{\boldsymbol{y}_1\boldsymbol{x}_v} & \boldsymbol{A}_{\boldsymbol{y}_1\boldsymbol{y}_1} & \dots & \boldsymbol{A}_{\boldsymbol{y}_1\boldsymbol{y}_n} \\ \vdots & \vdots & & \vdots \\ \boldsymbol{A}_{\boldsymbol{y}_n\boldsymbol{x}_v} & \boldsymbol{A}_{\boldsymbol{y}_n\boldsymbol{y}_1} & \dots & \boldsymbol{A}_{\boldsymbol{y}_n\boldsymbol{y}_n} \end{array}\right) \cdot \boldsymbol{x}_t \tag{3.9}$$

Hierbei gilt für die rechte untere Partition, dass alle Teilmatrizen $\boldsymbol{A}_{\boldsymbol{y}_i\boldsymbol{y}_i}$ Einheitsmatrizen der Größe 3×3 sind. Die übrigen Einträge in dieser Partition sowie alle $\boldsymbol{A}_{\boldsymbol{y}_i\boldsymbol{x}_v}$ und $\boldsymbol{A}_{\boldsymbol{x}_v\boldsymbol{y}_i}$ sind Nullmatrizen. Für die Partition $\boldsymbol{A}_{\boldsymbol{x}_v\boldsymbol{x}_v}$ gilt

$$\begin{aligned} \boldsymbol{x}_{v_{t+1}} &= \boldsymbol{A}_{\boldsymbol{x}_v\boldsymbol{x}_v} \cdot \boldsymbol{x}_{v_t} \\ \begin{pmatrix} t_{x_{t+1}} \\ t_{y_{t+1}} \\ t_{z_{t+1}} \\ v_{x_{t+1}} \\ v_{y_{t+1}} \\ v_{z_{t+1}} \end{pmatrix} &= \begin{pmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} t_{x_t} \\ t_{y_t} \\ t_{z_t} \\ v_{x_t} \\ v_{y_t} \\ v_{z_t} \end{pmatrix} \quad . \end{aligned}$$

Δt ist das Zeitintervall zwischen der aktuellen und der vorangehenden Iteration. Ein Zustandsübergang besteht also darin, die Kameratranslation mit der Translationsgeschwindigkeit gemäß der verstrichenen Zeit anzupassen.

Die Translationsgeschwindigkeit selbst bleibt im Modell zunächst konstant. Da das jedoch nicht der Realität entspricht, enthält das Rauschen des Systemmodells ($\boldsymbol{w}$) eine unbekannte Beschleunigung $\boldsymbol{a}$. Diese Beschleunigung führt zu einer Veränderung der Systemgeschwindigkeit und der Kameratranslation mit einem additiven Faktor $\boldsymbol{V} = \boldsymbol{a} \cdot \Delta t$. Der zugrunde liegende Prozess lautet also[3]

$$\boldsymbol{x}_{v_{t+1}} = \boldsymbol{f}_v \left(\begin{pmatrix} \boldsymbol{x}_{v_t} & \boldsymbol{a}_t \end{pmatrix}^T \right) \tag{3.10}$$

$$= \boldsymbol{A}_{\boldsymbol{x}_v \boldsymbol{x}_v} \cdot \boldsymbol{x}_{v_t} + \begin{pmatrix} \Delta t^2 & 0 & 0 \\ 0 & \Delta t^2 & 0 \\ 0 & 0 & \Delta t^2 \\ \Delta t & 0 & 0 \\ 0 & \Delta t & 0 \\ 0 & 0 & \Delta t \end{pmatrix} \cdot \begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix} \quad . \tag{3.11}$$

Das Rauschen des Systemmodells $\boldsymbol{w}$ wird nicht explizit in den Kalman-Gleichungen berücksichtigt. Stattdessen beeinflusst die Kovarianz $\boldsymbol{Q}$ dieses Rauschens in jeder Iteration die Systemkovarianz, welche sich schließlich auf die Aktualisierung des Systemzustandes auswirkt.

$\boldsymbol{Q}$ ist in Kalman Filtern als normalverteiltes Gaussrauschen vorgesehen und wird meist konstant gesetzt. Um eine präzisere Modellierung der unbekannten Beschleunigung zu erreichen, wird in [DMM03] eine dynamische Anpassung von $\boldsymbol{Q}$ in jeder Iteration vorgeschlagen. Dazu wird eine konstante Grundkovarianz des Systemrauschens $\boldsymbol{P}_n$ als Parameter des Verfahrens gewählt, die in jedem Schritt von links und rechts mit einer Matrix multipliziert wird, welche sich aus der Ableitung der Gleichung (3.10) nach der unbekannten Beschleunigung $\boldsymbol{a}$ ergibt. Für das hier gewählte Modell hieße das

$$\boldsymbol{Q} = \frac{\partial \boldsymbol{f}_v}{\partial \boldsymbol{a}} \boldsymbol{P_n} \frac{\partial \boldsymbol{f}_v}{\partial \boldsymbol{a}}^T \quad .$$

Damit erreicht man eine dynamische Veränderung der Systemrauschen-Kovarianz, die vom Einfluss der unbekannten Beschleunigung auf das System abhängt. Die Belegung der

[3] Hier wird das Quadrat der Zeitintervalle verwendet, da zum einen die Beschleunigung selbst pro Zeit berechnet wird, um die Geschwindigkeitsveränderung zu erhalten. Diese Veränderung wirkt zusätzlich pro Zeit auf die Kameraposition ein.

Grund-Kovarianz $\boldsymbol{P}_n$ ist dann kritischer Parameter für die Veränderungen in der Kamerabewegung. Hohe Werte in $\boldsymbol{P}_n$ ermöglichen die Abbildung starker Bewegungsschwankungen, führen jedoch gleichzeitig zu einer starken Zunahme der Unsicherheit im System. In einem solchen Fall sind sehr gute Messwerte mit entsprechend geringen Varianzen nötig, um das System stabil zu halten. Auf der anderen Seite erzwingen kleine Werte in $\boldsymbol{P_n}$ sehr glatte Bewegungen, ermöglichen aber ein stabileres System auch bei relativ unsicheren Messwerten.

Zur Umrechnung von Systemzuständen in eine Repräsentation als Messwert ist die Projektion der 3D-Marker anhand der aktuellen intrinsischen und extrinsischen Kameraparameter notwendig. Multipliziert man die Projektionsgleichung (2.3) aus und übernimmt die Terme in die perspektivische Division (2.5), so erhält man [4]

$$\begin{pmatrix} p_x^P \\ p_y^P \\ p_z^P \end{pmatrix} = \begin{pmatrix} f_x r_{11} + c_x r_{31} & f_x r_{12} + c_x r_{32} & f_x r_{13} + c_x r_{33} & f_x t_x + c_x t_z \\ f_y r_{21} + c_y r_{31} & f_y r_{22} + c_y r_{32} & f_y r_{23} + c_y r_{33} & f_y t_y + c_y t_z \\ r_{31} & r_{32} & r_{33} & t_z \end{pmatrix} \begin{pmatrix} p_x^W \\ p_y^W \\ p_z^W \\ 1 \end{pmatrix}$$

$$= \begin{pmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ r_{31} & r_{32} & r_{33} & t_z \end{pmatrix} \begin{pmatrix} p_x^W \\ p_y^W \\ p_z^W \\ 1 \end{pmatrix} \quad .$$

Daraus entstehen zwei Gleichungen zur Berechnung der Projektion (μ, ν) der Marker im Bild. Gleichzeitig lässt sich die Translation $\begin{pmatrix} t_x & t_y & t_z \end{pmatrix}$ auflösen durch die Summe aus der Translation in der vorherigen Iteration und der Translationsgeschwindigkeit $\boldsymbol{v}$ entsprechend der Zeit Δt.[5] Man erhält dann

$$h_{i_\mu}(\boldsymbol{x}) = \mu_i = \frac{f_x t_x + f_x \Delta_t v_x + c_x t_z + c_x \Delta_t v_z + y_{i_x} p_{11} + y_{i_y} p_{12} + y_{i_z} p_{13}}{t_z + \Delta_t v_z + r_{31} y_{i_x} + r_{32} y_{i_y} + r_{33} y_{i_z}} \quad \text{und}$$

$$h_{i_\nu}(\boldsymbol{x}) = \nu_i = \frac{f_y t_y + f_y \Delta_t v_y + c_y t_z + c_y \Delta_t v_z + y_{i_x} p_{21} + y_{i_y} p_{22} + y_{i_z} p_{23}}{t_z + \Delta_t v_z + r_{31} y_{i_x} + r_{32} y_{i_y} + r_{33} y_{i_z}} \quad .$$

[4] Die zweite Zeile dient nur der Vereinbarung der Abkürzungen p_{ij} für die folgenden Ausführungen.

[5] Da die Gleichung hier noch nicht in Bezug zu einer Iteration des Kalman Filters gesetzt wird, entfallen der Übersichtlichkeit halber die Indizierungen für die Iteration. Durch die Auflösung der Translation wären ansonsten alle Komponenten von $\boldsymbol{t}$ und $\boldsymbol{v}$ mit $t-1$ statt t zu indizieren.

Die Messwertgleichung ist die nichtlineare multivariate Funktion

$$\boldsymbol{h}(\boldsymbol{x}) = \begin{pmatrix} h_{\mu_1} \\ h_{\nu_1} \\ h_{\mu_2} \\ h_{\nu_2} \\ \vdots \\ h_{\mu_n} \\ h_{\nu_n} \end{pmatrix} (\boldsymbol{x}) \quad .$$

$\boldsymbol{h}(\boldsymbol{x})$ muss auch hier für die Verwendung im EKF linearisiert werden. Die Jacobi-Matrix hat die Form

$$\begin{pmatrix} \frac{\partial \mu_1}{\partial \boldsymbol{t}} & \frac{\partial \mu_1}{\partial \boldsymbol{v}} & \frac{\partial \mu_1}{\partial \boldsymbol{y}_1} & \cdots & \frac{\partial \mu_1}{\partial \boldsymbol{y}_n} \\ \frac{\partial \nu_1}{\partial \boldsymbol{t}} & \frac{\partial \nu_1}{\partial \boldsymbol{v}} & \frac{\partial \nu_1}{\partial \boldsymbol{y}_1} & \cdots & \frac{\partial \nu_1}{\partial \boldsymbol{y}_n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial \mu_n}{\partial \boldsymbol{t}} & \frac{\partial \mu_n}{\partial \boldsymbol{v}} & \frac{\partial \mu_n}{\partial \boldsymbol{y}_1} & \cdots & \frac{\partial \mu_n}{\partial \boldsymbol{y}_n} \\ \frac{\partial \nu_n}{\partial \boldsymbol{t}} & \frac{\partial \nu_n}{\partial \boldsymbol{v}} & \frac{\partial \nu_n}{\partial \boldsymbol{y}_1} & \cdots & \frac{\partial \nu_n}{\partial \boldsymbol{y}_n} \end{pmatrix} \quad .$$

Die partiellen Ableitungen werden immer in einem fixen Punkt gebildet. Dieser fixe Punkt ist die aktuellste verfügbare Schätzung des Systemzustandes. Es muss allerdings beachtet werden, dass zur Berechnung der Matrix die Translation der vorangehenden Iteration verwendet wird, damit die darin enthaltene Translationsgeschwindigkeit explizit wird.

Führt man die Ableitungen aus, so ergibt sich unter der Verwendung der zuvor eingeführten Abkürzungen für die Projektionsgleichung die in Abschnitt B.2 angegebene Matrix. Zur vereinfachten Darstellung werden dort die Zähler von h_{μ_i} mit n_{μ_i} (n wie *numerator*) und die Nenner mit d_{μ_i} (d wie *denominator*) abgekürzt; analog dazu n_{ν_i} und d_{ν_i}.

3.7.2 Initiale Kovarianz

Das Verhalten des Systems wird stark davon abhängen, wie die initiale Systemkovarianz $\boldsymbol{P}$ gesetzt wird. Bei der Initialisierung sind die Werte auf der Hauptdiagonalen der Matrix interessant, d. h. die unkorrelierten reinen Varianzen der Parameter. Initiale Werte für Translation und Translationsgeschwindigkeit lassen sich aus dem initialen Systemzustand

ableiten. Dieser wird durch den Prozess 2 “Erstellung einer neuen Karte” festgelegt, wo beide Werte durch markerbasiertes Tracking berechnet werden. Entsprechend sollten die ersten sechs Werte auf der Hauptdiagonalen mit einer Varianz belegt werden, die dem Fehler in der Rekonstruktion von *ARToolkitPlus* entspricht. Als sinnvoller Wert bestätigte sich in ersten Versuchen eine Abweichung in der Größenordnung 0.0001.

Die initialen Kovarianzen für die natürlichen Marker sollten direkt aus den Kovarianzen des Filters berechnet werden, mit dem sie initialisiert wurden (siehe Abschnitt 3.6.3). Da dort jedoch eine 6-dimensionale Parametrisierung benutzt wird, ist eine Umrechnung der Kovarianzen in die dreidimensionale Parametrisierung notwendig. Eine sinnvolle Konvertierung erhält man über die Jacobi-Matrix der Funktion (2.6), wenn man diese von links und rechts mit der Kovarianz des Markers in 6-dimensionaler Parametrisierung multipliziert. So entsteht aus der 6×6-Matrix wieder eine gültige Kovarianzmatrix der Größe 3×3 unter Berücksichtigung der Eigenschaften der ursprünglichen Darstellung.

3.7.3 Vereinfachte Variante

Das Verhalten des soeben beschriebenen Erweiterten Kalman Filter Modells ist komplex. Um stabile Ergebnisse zu erzielen, ist eine möglichst exakte Einstellung der Systemparameter notwendig, insbesondere der initialen Systemkovarianz $\boldsymbol{P}$, der Systemrauschen-Kovarianzmatrix $\boldsymbol{Q}$ und der Messwertrauschen-Kovarianzmatrix $\boldsymbol{R}$. In einer ersten Implementierung divergierte das System je nach Einstellung sehr schnell. So muss zum Beispiel abgebildet werden, dass die Positionen der Marker als weitestgehend konstant anzunehmen sind, während sich die Kamera stark bewegt. In Experimenten kam es aber vor, dass die reale Kamerabewegung im Systemzustand auf eine Bewegung der Marker abgebildet wurde. Die Reaktion des Systems auf verschiedene Parametereinstellungen sowie verschiedene Rauschanteile in den Sensordaten sollte daher in möglichst eingeschränkten Szenarien simuliert werden, um die Zusammenhänge besser zu verstehen.

Daher wurde zunächst eine vereinfachte Implementierung des Modells vorgenommen, in der die Markerpositionen nicht Teil des Systemzustandes sind. Damit reduziert sich der Systemzustand auf den 6-dimensionalen Vektor $\boldsymbol{x}_v$, der nur die Kameraposition und die Translationsgeschwindigkeit beinhaltet. Für die Messwertgleichung bedeutet diese Ände-

rung, dass die 3D-Koordinaten der Marker nun als konstante Koeffizienten einfließen und nicht mehr im Ursprungsraum der Funktion liegen. Als linearisierte Messwertgleichung $\boldsymbol{H}$ kann die linke obere 6×6-Matrix aus (B.2) verwendet werden. Die Zustandsübergangsgleichung $\boldsymbol{A}$ reduziert sich auf die linke obere 6×6-Matrix von (3.9).

Dieser vereinfachte Filter wurde in der Klasse `CameraPoseEkf` implementiert, die in Abschnitt 4.4 beschrieben wird. Einige Experimente dazu sind in Abschnitt 5.3 aufgeführt.

3.8 Reinitialisierung natürlicher Marker

In den vorangehenden Kapiteln wurden beide in Bild 3.2 dargestellten Zyklen beschrieben: Der linke Zyklus dient der Erzeugung einer neuen Karte natürlicher Marker unter Zuhilfenahme eines markerbasierten Trackingverfahrens, der rechte Zyklus zur Schätzung der Kameraposition mittels dieser Karte. Beide Prozesse verarbeiten in einer Iteration die gelieferten Sensordaten und stellen Instanzen eines Erweiterten Kalman Filters dar.

Noch nicht beschrieben wurde der Prozess 3.2 “Reinitialisierung natürlicher Marker” in Bild 3.2. Dieser Prozess dient dazu, parallel zur Schätzung der Kameraposition einzelne natürliche Marker neu zu initialisieren. Die Reinitialisierung erfolgt unabhängig von der Schätzung der Kameraposition, obwohl sie die gleichen Sensordaten zur Verfügung gestellt bekommt. Eine Iteration des gesamten in Bild 3.2 rechts dargestellten Zyklus ist daher erst dann beendet, wenn beide Prozesse - 3.1 und 3.2 - abgeschlossen sind. Das Zustands-Übergangs-Diagramm in Bild 3.11 zeigt die verfeinerte Darstellung von Prozess 3.2. Es ist kein Zufall, dass das Schema kongruent zu dem Diagramm in Bild 3.8 ist, welches die Erzeugung einer neuen Karte beschreibt. In der Tat sind die beiden Prozesse nahezu identisch, da das gleiche Modell eines Erweiterten Kalman Filters zum Einsatz kommt, das in Abschnitt 3.6 hergeleitet wurde. Der einzige Unterschied besteht zunächst darin, dass in dem hier vorliegenden Fall auf die Schätzung der Kameraposition aus dem parallel laufenden Prozess 3.1 zugegriffen wird, anstatt eine markerbasierte Rekonstruktion zu verwenden.

Hier entsteht allerdings ein Problem, wenn basierend auf einem Punktmerkmal eine neue Schätzung erzeugt werden soll: Dazu sollte keinesfalls ein Merkmal verwendet werden,

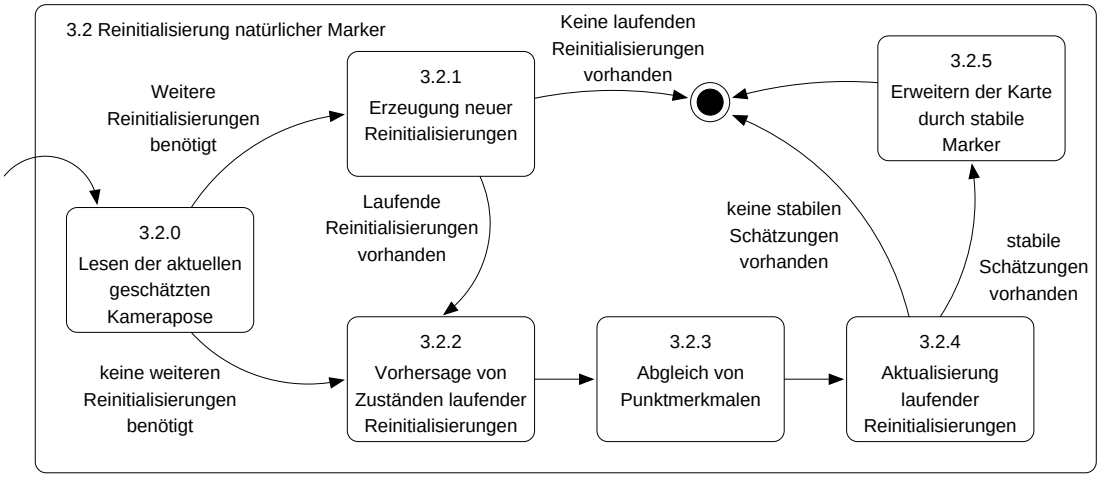


Bild 3.11: Verfeinerung des Prozesses “Reinitialisierung natürlicher Marker”.

das einem bereits bestehenden Marker zugeordnet ist. Daher muss in Teilprozess 3.2.1 ein Negativabgleich der detektierten SIFT-Merkmale mit den Markern in der Karte erfolgen, um die bereits “belegten” Merkmale herauszufiltern. Dieser Negativabgleich wurde in der Klasse `PointFeatureMatcher` implementiert, die in Abschnitt 4.3 vorgestellt wird. Der Negativabgleich könnte entfallen, wenn die Korrespondenzen aus dem Abgleich im parallel ablaufenden Prozess 3.1 zwischengespeichert und aus der Menge der hier verwendeten Punktmerkmale entfernt werden. Dadurch würde aber eine zeitliche Abhängigkeit zwischen den beiden Prozessen entstehen, die eine spätere Aufteilung des Verfahrens in separate *Threads* erschwert.

In Abschnitt 3.6 wurde unterschieden zwischen der Einzel- und der Multi-Instanz-Methode. Die Frage, welche der beiden Methoden vorzuziehen ist, stellt sich nun erneut, weil die gleiche Implementierung des Filters verwendet wird. Anders als bei der Erzeugung einer neuen Karte werden hier allerdings neue Schätzungen unabhängig voneinander in verschiedenen Iterationen angelegt und stabilisieren sich über stärker abweichende Zeiträume. Aus diesem Grund bietet es sich an, für die laufende Reinitialisierung von Markern separate Filter Instanzen zu verwenden.

Kapitel 4

Objektorientiertes Design und Realisierung

In diesem Kapitel erfolgt die Beschreibung einer C++-Bibliothek, die Klassen zu den im vorangehenden Kapitel behandelten Komponenten bereitstellt. Die zugehörigen Quellen sind auf der beiliegenden CD enthalten. Der Schwerpunkt dieses Kapitels wird darin bestehen, ein grundlegendes Verständnis für Klassenmodell und Anwendung zu vermitteln. Detaillierte und vollständige Informationen zu den einzelnen Methoden, Attributen und Parametern sind in englischer Sprache direkt im Quellcode enthalten und lassen sich mithilfe des kostenlosen Programms *Doxygen*[1] extrahieren und als umfangreiches HTML-Dokument ausgeben. Eine Fassung dieser HTML-Dokumentation ist bereits auf der beiliegenden CD enthalten.

Die entwickelten Softwarebausteine sollen eine solide Basis für künftige Arbeiten am Fraunhofer Institut für angewandte Informationstechnik bilden. Dazu ist neben einer sorgfältigen Beschreibung des Konzepts ein gutes Verständnis der Prozesse unter verschiedenen Parametereinstellungen erforderlich. Aus diesem Grund wurde die Bibliothek um Klassen zur Simulation von Sensordaten ergänzt. Simulatoren sind ein wichtiges Werkzeug, um das Verhalten der Komponenten in praktischen Situationen kontrolliert nachvollziehbar zu machen. Sie werden in Abschnitt 4.5 separat erläutert.

[1] `http://www.stack.nl/~dimitri/doxygen/download.html`

4.1 Klassenmodell

Das Klassenmodell mit den wichtigsten Komponenten der Bibliothek zeigt Bild 4.1. Die Klassen sind gemäß der prozessorientierten Sicht auf das im vorangehenden Kapitel beschriebene Systemmodell gruppiert: Grün hinterlegt sind die Komponenten zum Einlesen von Sensordaten, rot hinterlegt die Klassen zur Bildverarbeitung und blau hinterlegt die Klassen für die Rekonstruktion, zu der auch die Initialisierung natürlicher Marker zählt. Das Modell enthält nicht alle Klassen und Beziehungen, sondern nur einen für das Verständnis relevanten Teil. Die grün hinterlegten Klassen werden im nächsten Abschnitt 4.2.1 erläutert. Anschließend erfolgt in Abschnitt 4.3 eine Betrachtung der Klassen zur Punktdetektion und zum Vergleich von Punktmerkmalen. Die blau hinterlegten Implementierungen der Erweiterten Kalman Filter werden in 4.4 beschrieben. Am Ende des Kapitels wird das Konzept der Simulatorenklassen vorgestellt.

4.2 Sensordatenerfassung

4.2.1 Bilddaten

Die beiden Sensordatenströme für Inertial- und Bilddaten werden jeweils durch eine abstrakte Basisklasse gekapselt, für die mehrere Implementierungen zur Verfügung stehen. Zum Einlesen von Bilddatenströmen existiert zunächst die Klasse `CvAviGrabber`, die Funktionen der *OpenCV* Bibliothek kapselt um komprimierte Videos lesen zu können. Auch die Klasse `ImgSeqGrabber` dient dazu, Bilder vom Datenspeicher einzulesen, indem sie Sequenzen nummerierter Einzelbilder mit gleich formatierten Dateinamen erkennt. Um Bilder direkt von einer Kamera zu lesen, steht die Klasse `CvFrameGrabber` zur Verfügung, die eine Anbindung an die Bibliothek `videograbber` schafft. Diese Bibliothek ist nur unter *Microsoft Windows* verfügbar, so dass `CvFrameGrabber` unter Unix-basierten Systemen nicht kompiliert wird.

Alle Grabber-Klassen schreiben über die Methode `grab()` die Daten in ein bestehendes Bildobjekt. Stehen keine weiteren Bilddaten zur Verfügung, dann gibt diese Methode den Wert `false` zurück. In Codefragment 4.2 ist der Auszug eines Beispielprogramms ent-

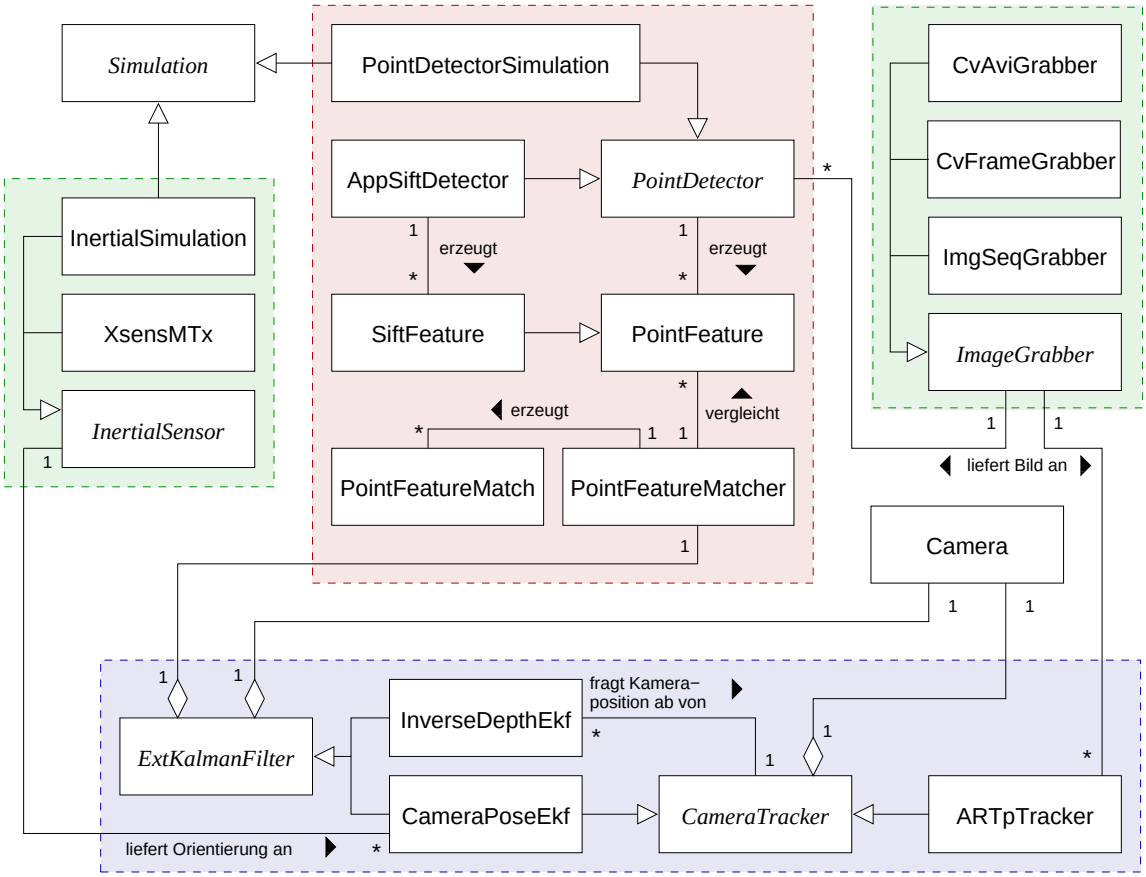


Bild 4.1: Klassendiagramm der wichtigsten Komponenten der Bibliothek. Die Namen abstrakter Klassen sind kursiv geschrieben.

halten, das eine Schleife über die Einzelbilder eines Films bildet, die mit `CvAviGrabber` realisiert wurde.

4.2.2 Inertialsensor

Als Inertialsensor stehen zwei Klassen zur Verfügung: Eine Simulation sowie eine Klasse für den Zugriff auf den *Xsens MTx* Inertialsensor. Das Konzept der Simulatoren wird in Abschnitt 4.5 beschrieben. Die Klasse `XsensMtx` spricht die COM-Schnittstelle des

Xsens MTx an, und liest auf Anfrage die Orientierung des Gerätes als 3×3-Matrix aus. Damit bei der Anfrage aktuelle Daten zurückgeliefert werden, wird vor jedem Aufruf ein `flush`-Befehl an das Gerät gesendet, welcher alte Werte in den Datenfeldern löscht und mit neuen Sensordaten füllt.

Die Orientierungsdaten des Gerätes sind zunächst relativ zum Erdmagnetfeld gegeben. Da das Erdmagnetfeld als solches in der Regel nicht bekannt ist, lassen sich die Daten so jedoch schlecht mit dem System synchronisieren. Der *Xsens MTx* stellt eine Reihe von Funktionen zur Neuinitialisierung des internen Weltkoordinatensystems bereit. Löst man eine Initialisierung des Typs "globale Neuinitialisierung" aus, so speichert der *Xsens MTx* seine eigene Orientierung zu diesem Zeitpunkt als "neues" Weltkoordinatensystem.

Die Orientierungsdaten liegen laut Handbuch als Feld von Fließkommawerten der Länge neun vor, das nach Spalten der zugrunde liegenden Matrix sortiert ist. Eine einfache Interpretation als 3×3-Matrix würde also die Rotation liefern. Daher muss die resultierende Matrix transponiert werden, um die Orientierung zu erhalten. Orientierungen werden dem "Alter" nach von rechts multipliziert:

$$\boldsymbol{O}_{gesamt} = \boldsymbol{O}_{t=0} \cdot \boldsymbol{O}_{t=1} \cdots$$

Die Multiplikationen lassen sich als inkrementelle Rotation des Weltkoordinatensystems auffassen, bis es schließlich die Orientierung des aktuellsten Kamerakoordinatensystems aufweist.

Die Methode `resetOrientation` der Klasse `XsensMtx` implementiert eine globale Neuinitialisierung des Sensors nach diesem Schema. Sie erlaubt die optionale Angabe einer Basis-Orientierung zum Zeitpunkt der Initialisierung. Mit dieser wird die aktuelle Orientierung des *Xsens MTx* von rechts multipliziert. Auf diese Weise lässt sich die Orientierung des Inertialsensors relativ zum gewünschten Weltkoordinatensystem bei der Initialisierung festhalten, und künftige Orientierungen werden relativ dazu ausgegeben.

Experimente mit der Klasse `XsensMtx` führten bisher jedoch leider nicht zu zufriedenstellenden Ergebnissen. Nach Befestigung des Sensors an der Kamera konnte die Orientierung zwar zurückgesetzt und mit dem durch die Klasse `ARTpTracker` bestimmten Weltkoordinatensystem abgeglichen werden, jedoch zeigten sich bereits nach kurzer Zeit deutliche Abweichungen zwischen der konvertierten Orientierung aus dem *Xsens MTx* und der

Referenz aus der markerbasierten Kamerarekonstruktion. Der Fehler kumuliert sich wie durch starken Drift und wird zusätzlich umso stärker, je weiter sich die Kamera von ihrer anfänglichen Position fort bewegt. Zum Zeitpunkt der Initialisierung sowie nach schwachen Bewegungen sind die Werte dahingegen korrekt. Die Probleme könnten verschiedene Ursachen haben: Zunächst ist es wichtig, dass die Achsen der Koordinatensysteme der Kamera und des Sensors gleich ausgerichtet sind, damit die Rotationen auf die beschriebene Weise miteinander verrechnet werden können. Dazu ist eine stabile und möglichst genau kalibrierte Montage erforderlich. Im Rahmen dieser Arbeit war nur eine provisorische Montage möglich, da das zur Verfügung stehende Gerät zeitgleich in anderen Projekten eingesetzt wurde. Als weiterer Grund für die Fehler kommen magnetische Störeinflüsse in Betracht, welche die Bestimmung des Erdmagnetfelds durch den *Xsens MTx* negativ beeinflussen.

4.3 Punktdetektoren

Für Punktdetektoren steht die abstrakte Basisklasse `PointDetector` zur Verfügung. Diese Basisklasse wird durch die in Abschnitt 3.4.1 beschriebene performante Approximation eines SIFT-Detektors in Form der Klasse `AppSiftDetector` sowie in Form eines Simulators[2] implementiert. Die Basisklasse ermöglicht eine einfache Erweiterung der Bibliothek mit weiteren Punktdetektoren. Um beispielsweise einen Harris-Detektor einzubinden, könnte man eine neue Klasse `HarrisDetector` ableiten, welche die Funktion `cvCornerHarris` aus *OpenCV* aufruft. Deskriptoren zu den neuen Merkmalen lassen sich leicht erzeugen: In Abschnitt 4.3.1 wird eine eigene Klassenhierarchie beschrieben, die diese Aufgabe übernimmt.

`PointDetector` legt die Schnittstelle der Punktdetektoren fest. Die Detektorenklassen speichern die erzeugten Merkmale nicht selbst, sondern schreiben sie in einen STL Vektor, den die aufrufende Einheit anlegen muss. Beim Aufruf der Methode `detectFeatures` wird dem Detektor neben dem Eingabebild eine Referenz auf diesen Vektor zur Speicherung der neuen Merkmalobjekte übergeben. Die Objekte sind von der aufrufenden Einheit

[2] Auf die Simulation wird in Abschnitt 4.5 näher eingegangen.

selbst zu löschen, wenn sie nicht mehr benötigt werden. Dieses Verhalten wurde gewählt, weil Punktmerkmale auf verschiedene Weise und letztlich unabhängig von der Lebensdauer des Detektors verwendet werden.

Sehr häufig ist es erforderlich, Korrespondenzen zwischen zwei Listen von Punktmerkmalen zu finden. Diese Aufgabe übernimmt die Klasse `PointFeatureMatcher`. Sie stellt Methoden bereit, um zwei Listen von Punktmerkmalen abzugleichen oder in einer Liste die beste Korrespondenz zu einem bestehenden Merkmal zu finden. Das Codefragment 4.1 zeigt den Ausschnitt eines Programms, das SIFT-Merkmale in zwei Bildern detektiert und anschließend eine Liste mit Korrespondenzen erzeugt. Ein korrespondierendes Merkmal wird durch die Klasse `PointFeatureMatch` repräsentiert, die einen Zeiger auf jedes der beiden Merkmalobjekte sowie die berechneten Abstandsmaße enthält. `PointFeatureMatcher` verhält sich bei der Rückgabe der Ergebnisse ähnlich wie `PointDetector`, indem die aufrufende Einheit einen STL Vektor zum Speichern der Korrespondenzen bereitstellen muss.

Damit Detektion und Abgleich von Punktmerkmalen möglichst performant ablaufen, werden Merkmale und Korrespondenzen in STL Vektoren als Objektzeiger gespeichert. Um möglichst wenige Objekte zu kopieren, erzeugt die Klasse `PointFeatureMatcher` zunächst keine neuen Merkmale, wenn sie Objekte vom Typ `PointFeatureMatch` bildet. Sie speichert dort lediglich Zeiger auf die bestehenden Merkmalobjekte aus den Eingabelisten. Dieses Verhalten minimiert den zusätzlichen Speicher- und Verwaltungsaufwand beim Abgleich von Punktmerkmalen, erfordert aber eine erhöhte Aufmerksamkeit bei der Verwaltung der Objekte: Werden die Eingabelisten mit den Punktmerkmalen gelöscht, so enthalten die Objekte des Typs `PointFeatureMatch` ungültige Zeiger. Es ist also wichtig, die Korrespondenzen gleichzeitig mit den ursprünglichen Punktmerkmalen zu löschen. Damit alternativ ein sicheres Verhalten zur Verfügung steht, stellen alle Methoden zum Abgleich von Punktmerkmalen optional den Parameter `copy` bereit. Wenn dieser auf `true` gesetzt wird, werden echte Kopien der Punktmerkmalobjekte angefertigt. Der Abgleich verliert dann aber an Performanz und verursacht einen höheren Speicherbedarf.

Für den Abgleich von Punktmerkmalen wurden in Abschnitt 3.7 zwei Strategien vorgestellt. Der optionale Parameter `strategy` ermöglicht es, bei jedem Abgleich eine

```
// read images
IplImage* img1 = cvLoadImage( "img1.png", -1 );
IplImage* img2 = cvLoadImage( "img2.png", -1 );

// create detector and detect features in both images
AppSiftDetector* sift = new AppSiftDetector( 0.1f, 5.0f );
std::vector<PointFeature*> list1, list2;
sift->detectFeatures( img1, list1 );
sift->detectFeatures( img2, list2 );

// match the detected features
PointFeatureMatcher* matcher = new PointFeatureMatcher( 500 );
std::vector<PointFeatureMatch> matches;
matcher->match( list1, list2, matches,
        PointFeatureMatcher::STRATEGY_UNIQ );

// release memory
for ( std::vector<PointFeature*>::iterator
        it = list1.begin(); it != list1.end(); it++ )
    delete *it;
for ( std::vector<PointFeature*>::iterator
        it = list2.begin(); it != list2.end(); it++ )
    delete *it;
```

Codefragment 4.1: Abgleich von SIFT-Merkmalen eines Bildpaars

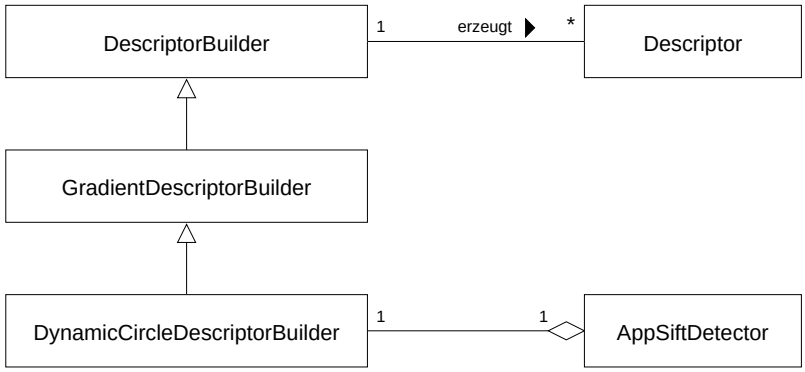


Bild 4.2: Klassen zur Erzeugung und Verwaltung von Deskriptoren. Die Klasse `AppSiftDetector` verwendet eine solche Klasse als statische Membervariable.

dieser Strategien explizit auszuwählen. Dazu steht ein Aufzählungstyp mit den Werten `STRATEGY_BEST` und `STRATEGY_UNIQ` zur Verfügung. Wird der Parameter nicht angegeben, gilt `STRATEGY_UNIQ`.

4.3.1 Deskriptoren

Für jede Punktmerkmalklasse, die von `PointFeature` abgeleitet ist, muss ein Deskriptor zur Verfügung stehen. Das ist notwendig, damit ein Abgleich der Merkmale erfolgen kann. Zur Erzeugung und Speicherung der Deskriptoren existieren eigene Klassen, die in Bild 4.1 zugunsten der Übersichtlichkeit ausgelassen wurden und stattdessen in Bild 4.2 dargestellt sind. Die abstrakte Basisklasse `DescriptorBuilder` schreibt das grundsätzliche Konzept der Offline-Berechnung von Histogramm-Indizes vor, das in Abschnitt 3.4.2 ausführlich beschrieben wurde. Als weitere Abstraktionsebene wurde die Klasse `GradientDescriptorBuilder` eingeführt, die als Basis für die Deskriptorenerzeugung je ein Bild mit Gradienten und Steigungen benötigt. Als konkrete Implementierung des in Abschnitt 3.4.2 vorgestellten Konzepts wurde davon schließlich die Klasse `DynamicCircleDescriptorBuilder` abgeleitet. Ein Objekt dieser Klasse ist als statische Membervariable in `AppSiftDetector` enthalten. So werden die Histogrammindizes im Regelfall nur einmal pro Laufzeit berechnet. Neue Instanzen der Klasse

verwenden dann das bereits bestehende Objekt. Lediglich für den Fall, dass die Parameter zur Deskriptorerzeugung geändert werden, muss eine Neuberechnung erfolgen.

4.4 Rekonstruktion

Kern der in Kapitel 3 beschriebenen Rekonstruktionsansätze ist jeweils ein Erweiterter Kalman Filter. Es wurden zwei Modelle mit verschiedenen konkreten System- und Messwertmodellen entwickelt. Zunächst stellt sich also die Frage, ob und inwiefern man Gemeinsamkeiten der beiden Modelle in eine gemeinsame Basisklasse zusammenfassen kann. Ein grundsätzlicher Bestandteil aller Kalman Filter ist die Aufteilung des Prozesses in einen Prädiktions- und einen Aktualisierungsschritt. In Abschnitt 3.5 wurde festgestellt, dass in beiden Fällen als Linearisierung eine Jacobi-Matrix für die Messwertgleichung bereitgestellt werden muss, während die Systemübergänge auf direktem Weg als Matrix modelliert werden können. Sieht man von der Belegung der Systemübergangsmatrix $\boldsymbol{A}$ und der Jacobi-Matrix der Messwertgleichung, $\boldsymbol{H}$, ab, so lassen sich die Methoden zur Vorhersage und zur Aktualisierung in eine Implementierung zusammenfassen. $\boldsymbol{H}$ und $\boldsymbol{A}$ müssen dann für jedes Modell separat gebildet werden. Anschließend erfolgen ebenfalls getrennt die Beobachtungen der Sensordaten. Während dem gemeinsamen Aktualisierungsschritt der Filter muss eine Vorhersage der erwarteten Messung mit der nichtlinearen Messwertgleichung $\boldsymbol{h}$ analog (3.5) erfolgen, um die Innovation zu ermitteln. Diese Vorhersage unterscheidet sich ebenfalls pro Modell.

Man kann also eine gemeinsame Basisklasse `ExtKalmanFilter` zu den konkreten Implementierungen `InverseDepthEkf` und `CameraPoseEkf` abgrenzen. Die Basisklasse enthält die Methoden `predict` und `update`, die der Kalman Vorhersage und der Kalman Aktualisierung entsprechen. `predict` führt dabei die Gleichungen (3.2) und (3.3) aus. Für den Filter zur Schätzung der Kameraposition muss zusätzlich $\boldsymbol{A}$ angepasst werden, da die verstrichene Zeit Δ_t darin enthalten ist. Dazu implementieren die Kindklassen die Funktion `recomputeA`, die bei `InverseDepthEkf` leer bleibt. Die Methode `update` implementiert die Gleichungen (3.4), (3.5) und (3.6). Sie ruft zwei Methoden auf, die von den Kindklassen implementiert werden müssen. Die erste davon, `recomputeH`, führt die Berechnung der aktuellen Jacobi-Matrix der Messwertgleichung aus. Die Zweite

übernimmt die Konvertierung des aktuellen Systemzustandes in eine Messwertrepräsentation (`computePredictedMeasurement`). Sie entspricht dem Term $\boldsymbol{h}(\bar{\boldsymbol{x}}_{t+1})$ in (3.5).

Den Kindklassen von `ExtKalmanFilter` bleibt die Implementierung von Methoden zur Beobachtung von Messwerten frei. `CameraPoseEkf` verwendet eine Methode zur Beobachtung einer Liste von Punktmerkmalen (`observeFeatures`) und eine weitere zur Beobachtung eines Inertialsensors (`observeInertial`), die vor `update` aufgerufen werden.

`InverseDepthEkf` stellt dagegen eine Methode zur Beobachtung einer Markerposition im Bild bereit. Außerdem muss `InverseDepthEkf` in jeder Iteration eine Schätzung der Kamerapose lesen; und zwar je nach Kontext aus dem markerbasierten Verfahren (Bild 3.8) oder aus `CameraPoseEkf` (Bild 3.11). Um diesen Sachverhalt abbilden zu können, wurde die abstrakte Basisklasse `CameraTracker` eingeführt, die eine Methode `getCamera` zum Auslesen der geschätzten Pose vorschreibt. Von dieser Klasse wurde `CameraPoseEkf` zusätzlich durch Mehrfachvererbung abgeleitet. Als markerbasiertes Trackingverfahren wurden entsprechende Funktionen der Bibliothek *ARToolkitPlus* in die Klasse `ARTpTracker` gekapselt, die ebenfalls Kindklasse von `CameraTracker` ist. In `InverseDepthEkf` kann so eine einfache Methode `observeTracker` bereitgestellt werden, die je nach Kontext Kameradaten von einer dieser beiden Klassen liest.

`ARTpTracker` ermöglicht zusammen mit den bereits erwähnten Klassen zum Einlesen von Bilddatenströmen eine sehr leichte Implementierung der Verfolgung eines Markers über mehrere Bilder. Ein entsprechendes Beispiel zeigt Codefragment 4.2. In den Zeilen 0 bis 3 werden die entsprechenden Klassen instantiiert sowie zwei passende Bildobjekte erzeugt. Die Zeilen 4 bis 7 lesen Kalibrierdaten ein. Die gesamte Markerverfolgung über den Datenstrom erfolgt in den Zeilen 8 bis 15, wo bei erfolgreicher Detektion eines Markers das Weltkoordinatensystem in ein Bild eingezeichnet wird.

Während der Kalman Filter Iterationen ist es möglich, dass sich die Größe von Systemzustand oder Messwertvektor ändert. Das tritt z. B. dann auf, wenn neue Marker in die Karte aufgenommen und alte Marker verworfen werden müssen. Prinzipiell müssten also bei jeder solchen Anpassung die internen Zustände und Kovarianzmatrizen der Klasse `ExtKalmanFilter` mit neuen Größen erzeugt und Teile der alten Daten kopiert werden. Dieser Vorgang wäre sehr aufwändig verglichen mit der Menge an Daten, die neu

```
ImageGrabber* grabber = new CvAviGrabber("movie.avi");
IplImage* frame = cvCreateImage(grabber->getImageSize(),8,3);
IplImage* output = cvCreateImage(grabber->getImageSize(),8,3);
ARTpTracker* tracker = new ARTpTracker(frame);
if (!tracker->readCalibrationData("quickcam.txt")) {
    std::cerr << "Could not read calibration file";
    return -1;
}
while(grabber->grab(frame)) {
    if (!tracker->readFrame(frame)) {
        std::cout << "No marker found.\n";
    } else {
        // draw world coordinate system axes to output image
        tracker->getCamera().drawWcsAxes(output);
    }
}
std::cout << "End of movie file.\n";
```

Codefragment 4.2: Verfolgung eines ARToolKitPlus Markers in einem Bilddatenstrom

hinzukommen. Daher wurde hier eine andere Lösung gefunden, indem maximale Größen der Karte und der Messwertvektoren vorab festgelegt werden. So können zu Beginn entsprechende Matrizen angelegt werden, die vorerst nur teilweise gefüllt sind. *OpenCV* bietet die Möglichkeit, Teilmatrizen zu erzeugen, die auf einen rechteckigen Ausschnitt einer übergeordneten Matrix verweisen. Dabei werden keine Daten kopiert, sondern lediglich ein neuer Header für die Matrixstruktur auf dem gleichen Speicher erzeugt. Jede Änderung in der übergeordneten Matrix wirkt sich also auf die entsprechende Submatrix aus und umgekehrt. Von dieser Möglichkeit wurde in der Basisklasse `ExtKalmanFilter` Gebrauch gemacht, so dass bei den Berechnungen der Kalman Filter stets nur diejenigen Ausschnitte der Matrizen berücksichtigt werden, die gültige Werte enthalten.

Um eine komfortable Beobachtung der verschiedenen Kalman Filter Instanzen in Experimenten zu ermöglichen, wurde zusätzlich die Klasse `EkfPlotter` entwickelt. Diese ermöglicht es, die Zustände einer von `ExtKalmanFilter` abgeleiteten Klasse zu verschiedenen Zeitpunkten abzugreifen und daraus später grafische Auswertungen im *PostScript*-Format zu erzeugen. Die Experimente im nächsten Kapitel machen davon ausgiebig Gebrauch, um z. B. die Entwicklung der Systemzustände wie in Bild C.15 auf Seite 122 sichtbar zu machen. Die Anwendung ist sehr einfach: Mit der Methode `catchState` werden wiederholt Zustände aufgezeichnet. Zur Ausgabe der Funktionsplots stehen mehrere Alternativen zur Verfügung, z. B. die Methode `plotState` zur Ausgabe der Entwicklung des Systemzustands über die Zeit.

4.5 Simulatoren

In Bild 4.1 wurden bereits zwei Simulatorenklassen gezeigt, die durch Mehrfachvererbung entstehen. Den entsprechenden Ausschnitt der Klassenhierarchie zeigt Bild 4.3 mit einer Auswahl relevanter Methoden und Attribute der beteiligten Klassen. Bei den Simulatoren handelt es sich jeweils um eine Komponente zur Bereitstellung von Sensordateninformation, die für Experimente an Stelle eines echten Sensors eingesetzt wird. Diese Simulatoren sollen später möglichst leicht gegen entsprechende “echte” Komponenten ausgetauscht werden können. Daher ist es sinnvoll, Simulatoren von der gemeinsamen Basisklasse des jeweiligen Sensortyps abzuleiten; das sind hier die Klassen `InertialSensor` und

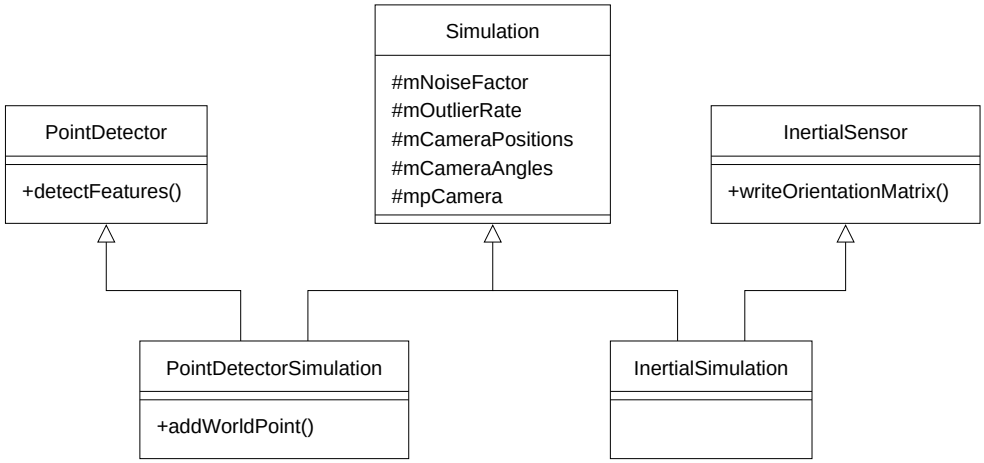


Bild 4.3: Schema der Mehrfachvererbung bei den Simulatorenklassen mit ausgewählten Attributen und Methoden.

`PointDetector`.

Trotzdem zeichnen sich alle Simulatoren durch Eigenschaften aus, die sie unabhängig vom Sensortyp aufweisen müssen. Zunächst sollten sie in der Lage sein, normalverteiltes Rauschen sowie Ausreißer mit Hilfe eines Zufallsgenerators erzeugen zu können. Diese Funktionalität wurde in die Basisklasse `Simulation` integriert, die z. B. Methoden zum Setzen und Auslesen des Rauschfaktors und der Ausreißerrate bereitstellt. Außerdem müssen alle Simulatoren Kenntnis von der simulierten Kamerabewegung haben, zu der sie Sensordaten simulieren sollen. Die Kamerabewegung wird ebenfalls in der Basisklasse `Simulation` bereitgestellt, die ein Objekt der Klasse `Camera` als Attribut führt, dessen intrinsische und extrinsische Parameter für die Kindklassen zugreifbar sind. Um die Position der Kamera als simulierte Bewegung zu verändern, stellt `Simulation` zwei Möglichkeiten bereit: Zum Einen lässt sich mit der Methode `readCameraMoves` eine Kamerabewegung als Liste von Positionen und Orientierungen aus einer einfachen Textdatei einlesen, die dann im Simulatorobjekt als Liste abgelegt wird. Die Bewegungen werden jeweils durch Aufruf der Methode `doNextStep` ausgelöst. Als zweite Möglichkeit lässt sich mit der Methode `setCamera` die Kamera der Simulation direkt positionieren. Diese Methode kann beispielsweise genutzt werden, um die Kamera der Simulatoren mit der

Kamera der Klasse `ARTpTracker` zu synchronisieren. So lassen sich simulierte Sensordaten zu einem echten Bilddatenstrom erzeugen, wenn dieser künstliche Marker enthält.

Durch die Mehrfachvererbung von der Basisklasse `Simulation` wird diese Funktionalität für die Simulatorenklassen verfügbar, ohne die Austauschbarkeit mit den “echten” Sensorkomponenten zu erschweren. Es liegt also ein typischer Fall vor, in dem Mehrfachvererbung sinnvoll eingesetzt werden kann.

Kapitel 5

Experimente und Ergebnisse

5.1 Abgleich von Punktmerkmalen

Ein wichtiges Qualitätsmerkmal für den Abgleich von Punktmerkmalen ist der Ausreißeranteil in der Menge gültiger Korrespondenzen. Letztlich ist das Ziel, die Parameter so einzustellen, dass bei möglichst geringem Ausreißeranteil möglichst viele Korrespondenzen gefunden werden. Für das folgende Experiment wurde eine größere Anzahl von Testläufen auf dem in Bild 5.1 abgebildeten Bildpaar ausgeführt. Es handelt sich um zwei Ausschnitte des gleichen Originalbildes an unterschiedlichen Positionen. So lassen sich die Ausreißerraten automatisch berechnen, da die Verschiebung zwischen korrespondierenden Merkmalen bekannt ist. Der Überlappungsbereich der Bilder betrug etwa 75%.

Für jede Parametereinstellung wurden Testläufe mit verschiedenen Schwellwerten des Deskriptorabstands gemacht, um unterschiedlich große Mengen gültiger Korrespondenzen zu erhalten. Ein höherer Schwellwert liefert mehr gültige Korrespondenzen, lässt aber Paare mit weniger ähnlichen Merkmalen zu. Für das Experiment wurden Schwellwerte in 10er-Abständen von 200 bis 1200 gewählt, womit eine enge und gleichmäßige Abtastung des typischen Wertebereichs gegeben ist. Eine Filterung nach dem Ortsabstand wurde nicht vorgenommen. Pro Schwellwert erfolgte dann eine Variation der Größe der Deskriptorkreise. Dazu wurden Radien zwischen 4 und 32 Pixeln in Schritten von 4 Pixeln getestet.

Bild 5.1: Bildpaar für die Experimente in Abschnitt 5.1

Als Vergleichsbasis des Experiments dienten Testläufe mit Deskriptoren, die zu vier Teilkreisen je ein Gradientenhistogramm mit 16 Urnen enthalten. Diese Deskriptoren sind unabhängig vom Radius des Kreises 64-stellig. Das Basisexperiment ist in Bild C.1 auf Seite 108 abgebildet. In diesem Experiment sieht man deutlich, dass die Ausreißerrate für die kleinsten und die größten Radien am Höchsten ist. Die beste Kurve bildet der Testlauf mit dem Radius 16. Erst ab einer “Abschöpfung” von mehr als 300 Korrespondenzen durch Vergrößerung des Deskriptorabstands entstehen hier Ausreißer. Die Rate beginnt ab einer Menge von 350 gültigen Korrespondenzen rapide zu steigen. Die anderen Radien verursachen bereits bei weniger gültigen Korrespondenzen Ausreißer, und die Steigung der Kurven pendelt sich bereits früher auf einen hohen Wert ein.

In Bild C.2 auf Seite 109 wurde das gleiche Experiment unter Variation der Anzahl von Kreisteilen durchgeführt. Das obere Diagramm zeigt die Werte für 6 Kreisteile, das untere für 8 Kreisteile.[1] Man sieht sehr deutlich, dass die Kurven insgesamt gestaucht werden, d. h. die Ausreißerraten sind bei gleicher Anzahl gültiger Korrespondenzen geringer als in den vorigen Testläufen. Während bei den Läufen mit sechs Kreisteilen wie im Basisexperiment derjenige mit dem Radius 16 die besten Ergebnisse zeigt, verwischt sich diese Ausprägung in den Testläufen mit acht Kreisteilen. Sehr kleine Radien (4 und 8) und sehr große Radien (28 und 32) führen aber auch hier zu schlechteren Ergebnissen. Es wird er-

[1] Diese Parameter führen zu 96- bzw. 128-stelligen Deskriptoren.

kennbar, dass eine Erhöhung der Anzahl von Kreisteilen eine in dem beobachteten Bereich nahezu proportionale Verbesserung der Ausreißerrate mit sich bringt.

In Bild C.3 auf Seite 110 erfolgte eine Variation der Anzahl von Ringen auf zwei beziehungsweise drei Ringe, so dass hier Deskriptoren der Längen 128 und 172 im Vergleich zum Basisexperiment standen, bei dem die Länge 64 betrug.[2] Hier lässt sich eine noch stärkere Stauchung beobachten, die aber offenbar in gleichem Maße der Erhöhung der Deskriptorlänge entspricht wie im vorangehenden Experiment. Auffallend ist, dass der Lauf mit Radius 16 im Fall von 3 Ringen in der Relation zurückfällt. Diese Beobachtung verdeutlicht, dass eine höhere Zahl von Ringen nur für große Radien sinnvoll ist, da ansonsten die Unterteilung in Teilfenster zu stark von der Diskretisierung durch Pixel beeinflusst wird.

Abschließend erfolgte eine Variation der Anzahl von Urnen pro Histogramm. Hier wurden als praktikable Beispiele eine kürzere Variante mit 8 statt 16 Urnen (Deskriptorlänge 32) und eine längere Variante mit 32 statt 16 Urnen (Deskriptorlänge 128) für die Testläufe gewählt. Die Ergebnisse sind in Bild C.4 auf Seite 111 abgebildet. Überraschenderweise sind die Ergebnisse für die Testläufe mit 8 Urnen, die sehr kurze Deskriptoren und somit eine sehr hohe Performanz bewirken, nicht viel schlechter. Die Ausreißerraten brechen etwas früher aus; würde man jedoch eine etwas kleinere Menge gültiger Korrespondenzen akzeptieren und einen Radius von 16 oder höher wählen, so wäre die Qualität vergleichbar. Ebenso wird durch die Testläufe mit 32 Urnen deutlich, dass eine höhere Anzahl von Urnen bei Betrachtung des erhöhten Berechnungsaufwands für die Abstandsberechnung nur geringen Nutzen bringt.

Die Experimente zeigen, dass für eine Parametereinstellung zur performanten Erzeugung von Deskriptoren eine Verringerung der Urnen zweckmäßiger ist als eine Verringerung der Anzahl der Teilfenster. Umgekehrt bewirkt offenbar eine Erhöhung der Anzahl von Ringen und Kreisteilen eine stärkere Verbesserung der Ausreißerraten als die Erhöhung der Anzahl von Urnen. Eine Verbesserung der Ausreißerraten erfordert stets einen höheren Rechenaufwand.

[2] Im unteren Diagramm, also dem Testlauf mit drei Ringen, entfällt das Experiment mit dem Radius vier. Das liegt daran, dass eine Unterteilung in drei Ringe bei solch einem niedrigen Radius keinen Sinn machen würde.

Die Ausreißerraten steigen deutlich, wenn das Bildpaar stärkeren Beleuchtungsschwankungen, Rotationen und Verzerrungen unterliegt. Dennoch bleiben die Zusammenhänge der Parameter, die hier aufgezeigt wurden, gültig. Als Ergänzung ist das in Bild C.1 dargestellte Experiment mit einem Bildpaar durchgeführt worden, dass sich durch eine manuelle Weichzeichnung (Gaussfilter mit $\sigma = 2$) unterscheidet und so stark verrauschte Eingaben repräsentiert. Den Funktionsplot der Ergebnisse bei gleichen Parametereinstellungen wie im Basisexperiment mit diesem Bildpaar zeigt Bild C.5 auf Seite 112. Eine Variation auf sechs und acht Kreisteile ergab die in Bild C.6 auf Seite 113 dargestellten Ergebnisse. Man erkennt den deutlichen Anstieg der Ausreißerraten gegenüber den vorherigen Experimenten. Auch hier lassen sich die besseren Ergebnisse mit Radien im mittleren Bereich sowie höherer Anzahl von Kreisteilen erzielen. Zusätzlich fällt auf, dass der “ideale Radius”, der im Schnitt zu den geringsten Ausreißerraten führt, bei diesem stark verrauschten Bildpaar etwas höher liegt als bei dem idealen Bildpaar.

Experimente mit weiteren Bildpaaren, die hier nicht aufgeführt sind, ergaben ähnliche Ergebnisse. Um darüber hinaus einen subjektiven Eindruck der Korrespondenzsuche zu vermitteln, wurde stellvertretend ein Bildpaar mit variiertem Betrachtungswinkel und Betrachtungsabstand sowie veränderten Beleuchtungseigenschaften für einen Versuch ausgewählt, das aus dem Bilddatenstrom einer *Logitech Pro* Internetkamera stammt. Es wurde versucht, mit vier verschiedenen Deskriptorengrößen ohne weitere Nachverarbeitung eine Menge guter Korrespondenzen daraus zu extrahieren. Die Ergebnisse sind in Bild C.7 auf Seite 114 abgebildet. Es wird deutlich, dass durch geschickte Wahl der Parameter bereits eine sehr stabile Teilmenge von Korrespondenzen gebildet werden kann. So sind im zweiten Bild von oben keine relevanten Ausreißer zu finden. Dieses Experiment benutzte zusätzlich zur Filterung des Deskriptorabstands einen maximalen Ortsabstand von 100 Pixeln.

5.2 Initialisierung von 3D-Markern

In diesem Abschnitt wird das Verhalten der Initialisierung von 3D-Markerpositionen aus Punktkorrespondenzen beschrieben. Dabei kommt das Modell eines Erweiterten Kalman Filters zum Einsatz, das in Abschnitt 3.6 entwickelt wurde. Die folgenden Experimente

benutzten dazu als entsprechende Implementierung die Klasse `InverseDepthEkf`, die in Beispielprogramme eingebunden und mit simulierten sowie echten Punktkorrespondenzen versorgt wurde.

5.2.1 Simulation

Zunächst erfolgt eine Untersuchung des Filters in einer simulierten Umgebung. Die Klassen `PointDetectorSimulation` und `InertialSimulation` liefern dazu Sensordaten mit bekannten Rausch- und Ausreißeranteilen. Das Prinzip dieser Simulatoren wurde in Abschnitt 4.5 beschrieben. Das Experiment wurde mit dem Beispielprogramm `simulateMarkers` durchgeführt, welches im Ordner `Programmcode/examples` auf der beiliegenden CD enthalten ist.

Um die Initialisierung von Merkmalen mit inverser Tiefe zu prüfen, wurden Bildsequenzen benutzt, in denen ein *ARToolkitPlus* Marker sichtbar ist. Der Marker wurde getrackt, und die so ermittelte Kamerapose zum manuellen Setzen der simulierten Kamera benutzt (siehe auch hierzu Abschnitt 4.5). Die Klasse `PointDetectorSimulation` führt eine Liste von virtuellen 3D-Weltpunkten, die anhand der bekannten intrinsischen und extrinsischen Kameraparameter direkt in Bildkoordinaten projiziert werden. Diese Punktprojektionen können mit normalverteiltem Rauschen unterschiedlicher Stärke verzerrt und als Punktkorrespondenzen an den Filter weitergegeben werden. Darüber hinaus können in unregelmäßigen Abständen Ausreißer, d. h. zufällig erzeugte falsche Merkmalpositionen erzeugt werden.

Die Schätzungen wurden von der Klasse `EkfPlotter` protokolliert und können so als Funktionsplot ausgewertet werden. Zunächst wurden die Punktbeobachtungen mit einem Rauschfaktor von 1.0 versehen, so dass bei jeder Messung normalverteilte Schwankungen der gemessenen Bildpositionen bis zu einem Pixel auftreten. In Bild C.8 auf Seite 115 ist für eine Schätzung die Entwicklung des Fehlers der rekonstruierten 3D-Position über etwa 200 Frames dargestellt. Die drei Funktionsplots zeigen Ergebnisse von Experimenten mit gleich bleibender Messwertrauschen-Kovarianz $\boldsymbol{R}$ von 1.0, aber variierender Systemrauschen-Kovarianz $\boldsymbol{Q}$. Die Systemrauschen-Kovarianz betrug im oben abgebildeten Experiment konstant $10 \cdot \mathrm{e}^{-3}$, dann darunter $10 \cdot \mathrm{e}^{-6}$ und schließlich $10 \cdot \mathrm{e}^{-12}$. Die

Ergebnisse zeigen, dass niedrigere Werte in $\boldsymbol{Q}$ wie erwartet die Schätzung stabilisieren. Das System reagiert weniger schnell auf die Messwerte und wird so durch die normalverteilten Fehler weniger stark beeinflusst. Gleichzeitig wird das System aber auch träger: Bis zum ersten Mal ein korrekter Zustand mit einem Fehler nahe Null erreicht ist, vergeht mehr Zeit als in den Fällen mit höheren Werten für $\boldsymbol{Q}$.

Umgekehrt ist die Wirkung bei gleichbleibender Systemrauschen-Kovarianz $\boldsymbol{Q}$ und niedrigeren Werten in $\boldsymbol{R}$. In Bild C.9 auf Seite 116 sind die Funktionsplots eines Experiments aufgeführt, bei dem die Messwertrauschen-Kovarianz wiederholt herabgesetzt wurde. Sie zeigen, dass die Oszillationen in der Schätzung stärker werden. Gleichzeitig werden jedoch schneller Zustände mit geringem Fehler erreicht.

Das Experiment mit Variation der Systemrauschen-Kovarianz wurde anschließend mit einer Ausreißerrate von 0.05 wiederholt, so dass in 5% der Fälle eine völlig falsche Punktkorrespondenz erzeugt wurde. Das Rauschen wurde dabei auf 0 gesetzt, um die Ausreißer als Verursacher der beobachteten Effekte identifizieren zu können. Die Entwicklungen des Fehlers dieser Schätzungen sind in Bild C.10 auf Seite 117 abgebildet. Man kann klar die Frames erkennen, in denen Ausreißer auftreten. Je geringer die Systemrauschen-Kovarianz ist, umso weniger wirken sich die Ausreißer aus, und umso schneller kann sich die Schätzung von den falschen Eingabedaten erholen. Die Ausreißer in dieser Simulation sind fatal, da sie völlig zufällige Positionen im Bild haben. Daher divergieren die Schätzungen im hier abgebildeten Zeitabschnitt, nachdem mehrere Ausreißer in kurzen Abständen auftreten.

5.2.2 Initialisierung aus SIFT-Merkmalen

Um die Qualität der geschätzten 3D-Positionen zu SIFT-Merkmalen zu überprüfen, ist die Kenntnis der korrekten Werte nötig. Um korrekte Positionen automatisch berechnen zu können, wurden mehrere Bildströme einer Szene aufgezeichnet, in der ein *ARToolkitPlus* Marker sowie verschiedene Texturen auf einer planaren Fläche angeordnet sind. Es waren keine Objekte im Bild vorhanden, die nicht Teil dieser Fläche sind. Da der Marker selbst ebenfalls planar ist, müssen die 3D-Positionen aller detektierten Merkmale in der x-/y-Ebene des Markerkoordinatensystems liegen. Sie lassen sich dann basierend auf

der ersten Beobachtung, die auch zur Initialisierung des Strahls diente, direkt ermitteln. Hierzu verwendet man die Projektionsgleichung 2.3 von Seite 11, in der alle mit p_z^W zu multiplizierenden Anteile wegen $p_z^W = 0$ verschwinden:

$$\begin{aligned} \mu &= \frac{f_x \left(r_{11} p_x^W + r_{12} p_y^W + t_x\right)}{r_{31} p_x^W + r_{32} p_y^W + t_z} + c_x \\ \nu &= \frac{f_y \left(r_{21} p_x^W + r_{22} p_y^W + t_y\right)}{r_{31} p_x^W + r_{32} p_y^W + t_z} + c_y \end{aligned}$$

Diese Gleichungen lassen sich bezüglich der Unbekannten p_x^W und p_z^W umstellen:

$$\begin{aligned} p_x^W \underbrace{\left(\mu r_{31} - f_x r_{11} - c_x r_{31}\right)}_{a_\mu} + p_y^W \underbrace{\left(\mu r_{32} - f_x r_{12} - c_x r_{32}\right)}_{b_\mu} &= \underbrace{f_x t_x + c_x t_z - \mu t_z}_{c_\mu} \\ p_x^W \underbrace{\left(\nu r_{31} - f_y r_{21} - c_y r_{31}\right)}_{a_\nu} + p_y^W \underbrace{\left(\nu r_{32} - f_y r_{22} - c_y r_{32}\right)}_{b_\nu} &= \underbrace{f_y t_y + c_y t_z - \nu t_z}_{c_\nu} \end{aligned}$$

Dann lässt sich eine der beiden Unbekannten durch Gleichsetzen bestimmen:

$$\begin{aligned} \frac{c_\mu - p_y^W b_\mu}{a_\mu} &= \frac{c_\nu - p_y^W b_\nu}{a_\nu} \\ \Leftrightarrow p_y^W &= \frac{c_\nu / a_\nu - c_\mu / a_\mu}{b_\nu / a_\nu - b_\mu / a_\mu} \end{aligned}$$

Schließlich erhält man

$$p_x^W = \frac{c_\mu - p_y^W b_\mu}{a_\mu} \quad .$$

Die Genauigkeit ist in der Praxis von der Positionsschätzung in *ARToolkitPlus* abhängig, welche die Basis der Berechnungen bildet.

Diese Berechnung wurde dann angewandt, um die Entwicklung der Schätzung initialer natürlicher Marker aus SIFT-Merkmalbeobachtungen zu untersuchen. Die Untersuchungen wurden mit dem Beispielprogramm `estimateMarkers` vorgenommen, das im Ordner `Programmcode/examples` auf der beiliegenden CD zu finden ist. Das Programm implementiert sequentiell den Zyklus der Neuerstellung von Markern, wie er in Bild 3.2 und insbesondere 3.8 dargestellt ist. `estimateMarkers` erlaubt die Einstellung einer Vielzahl von Parametern für die SIFT-Merkmaldetektion und -zuordnung sowie für die beteiligten Erweiterten Kalman Filter. Die verwendeten Parametereinstellungen sind in Tabelle

Klasse	Parameter	Einstellung
AppSiftDetector	MinOffset	0.05
	PcrThreshold	7.0
InverseDepthEkf	SystemCovariance	0.0001
	MsmtCovariance	variabel
PointFeatureMatcher	UniqRatio	0.9
	MaxDescriptorDistance	400
	MaxSpatialDistance	20
DynamicCircleDescriptorBuilder	BinsPerHistogram	8
	NumSubpatches	4
	mMaxOffset	16
	mNumRings	1

Tabelle 5.1: Parametereinstellungen der beteiligten Klassen für das in 5.2.2 beschriebene Experiment.

5.1 aufgeführt. Das Testprogramm selbst hat zwei weitere kritische Parameter. Dazu gehört zunächst die Anzahl zu verfolgender Schätzungen (`estimates`). Für die Beispiele wurde ein Wert von 70 verwendet. Sehr starke Auswirkung hat die maximal erlaubte Anzahl aufeinander folgender Frames, während derer keine Beobachtung zu einer Schätzung gefunden werden kann (`gap`). Bei Überschreiten dieser Anzahl wird die Schätzung verworfen. Für dieses Experiment wurde `gap` auf den Wert 5 gesetzt. Diese Zahl ist jedoch nicht absolut zu verstehen: Das Programm `estimateMarkers` führt für jede Schätzung zusätzlich das Attribut `bonus`, das mit `gap` initialisiert wird. Erfolgt eine gültige Beobachtung, wird `bonus` um mindestens eins erhöht. Eine stärkere Erhöhung von bis zu drei Bonuspunkten erfolgt, wenn der Deskriptorabstand bei der Zuordnung sehr gering ist. Solange `bonus` größer als Null ist, bleibt die Schätzung erhalten. Auf diese Weise werden Schätzungen, für die eine sehr gute Beobachtung vorliegt, etwas länger vorgehalten als andere.

In Tabelle 5.1 wurde für den Parameter der Messwertrauschen-Kovarianz “variabel” angegeben. Das Programm `estimateMarkers` nutzt einen konstanten Wert als Basis, der jedoch proportional zur Qualität der Punktkorrespondenzen variiert wird: Sehr geringe

Deskriptorenabstände, die eine gute und verlässliche Korrespondenz andeuten, führen zu einer proportionalen Verringerung der Kovarianz für die Messung in dieser Iteration. Analog dazu wird die Kovarianz bei hohen Deskriptorenabständen leicht erhöht. Damit wird das Wissen über die Verlässlichkeit einer Messung an den Filter weiter gegeben, der so in der Lage ist, gute Punktkorrespondenzen bei der Aktualisierung des Systemzustandes stärker zu gewichten als schlechte.

In Bild C.11 auf Seite 118 werden verschiedene Funktionsplots einer Schätzung über etwa 80 Frames aus dem Programm `estimateMarkers` dargestellt. Der zugehörige Marker ist in Bild 5.2 links durch den roten Kreis angedeutet. Bild C.11 zeigt oben den Fehler in der x-, y- und z-Komponente, darunter die Lebenszeit mit den Komponenten Alter, `bonus` und `gap` sowie unten die Deskriptorenabstände in jeder Iteration. Die Funktionsplots zeigen, dass sich die Schätzung nach etwa 30 Frames stabilisiert. Der Fehler in der z-Komponente schwankt sowohl vor als auch nach Frame 30 am stärksten. Das ist dadurch zu erklären, dass die Schätzung am stärksten über den Parameter ρ, also die inverse Tiefe, bestimmt wird. Da das Weltkoordinatensystem, das durch den künstlichen Marker für *ARToolkitPlus* gegeben ist, die z-Achse in diesem Beispiel in Richtung der Kamera definiert, zeigen sich Schwankungen der inversen Tiefe auch am Deutlichsten im z-Bereich des Weltkoordinatensystems. Hinzu kommt, dass bedingt durch die geringen Schwankungen zwischen Frames eines kontinuierlichen Bilddatenstroms die Parallaxen relativ gering sind. So liegt wenig Information für die Schätzung der Tiefe vor. Für die Schätzung der Orientierung jedoch ist mehr verwertbare Information verfügbar, da selbst Beobachtungen ohne Parallaxe - also Punkte im Unendlichen - Aussagen über die Orientierung erlauben. Hierzu wurden theoretische Aspekte in Abschnitt 3.6 erläutert.

In Bild C.11 lässt sich in den beiden unteren Funktionsplots auch erkennen, wie sich Punktkorrespondenzen mit hohem euklidischen Abstand auf die Schätzung auswirken: Da in diesen Fällen die Messwertrauschen-Kovarianz auch höher gesetzt wurde, verliert die Schätzung nach mehreren schlechten Beobachtungen leicht an Stabilität. Die lässt sich sehr gut ab Frame 40 beobachten, wo wieder die z-Komponente leicht ausbricht.

In Bild C.14 auf Seite 121 ist der Verlauf einer weiteren Schätzung aufgeführt. Dabei handelt es sich um den in Bild 5.2 in der Mitte markierten Marker. Diese Schätzung ist gegenüber der Vorangehenden relativ instabil und weist durchgängig höhere Fehler auf.

Bild 5.2: Beobachtete Marker bei den Experimenten in 5.2.2

Im unteren Diagramm lässt sich an der grünen Linie der Grund dafür erkennen: Es treten regelmäßig Beobachtungslücken von mehreren Frames auf, während denen kein Punktmerkmal zugeordnet werden kann. Zwischen Frame 20 und Frame 30 wurde zusätzlich ein falsches Merkmal zugeordnet, das einen niedrigen Deskriptorabstand aufweist. Das kann passieren, wenn Texturen mit sich wiederholenden Mustern im Bild vorhanden sind, die in eng beieinander liegenden Bildregionen mehrere ähnliche Deskriptoren erzeugen. Der Filter stabilisiert sich nach diesem Ausreißer innerhalb weniger Frames auf einen deutlich niedrigeren Fehler.

Bild C.12 auf Seite 119 zeigt das Beispiel einer sehr guten Schätzung, die dem in Bild 5.2 rechts markierten Merkmal entspricht. Hier waren durchgängig gute Zuordnungen möglich, und wiederholte, sehr genaue Korrespondenzen führten zu einer Anhebung des Parameters bonus. Regelmäßige sehr geringe Deskriptorenabstände verursachten niedrige Messwertrauschen-Kovarianzen und stabilisierten die Schätzung. So kann man deutlich erkennen, wie der vergleichsweise hohe Deskriptorabstand in den Frames 31 bis 33 eine etwas höhere Messwertrauschen-Kovarianz bewirkt. Der Fehler der Schätzung steigt an dieser Stelle insbesondere für die z-Komponente an. In Frame 34 erfolgt dann eine gute Korrespondenz mit sehr geringem Deskriptorabstand, die zu einer geringen Messwertrauschen-Kovarianz führt. Dadurch wirkt sich die Messung stark auf die Schätzung aus, und der Fehler in der Schätzung wird deutlich verringert. Der Zusammenhang von Deskriptorabstand und Messwertrauschen-Kovarianz lässt sich in Bild C.13 auf Seite 120 erkennen, wo Funktionsplots der Deskriptorabstände und der zugehörigen Messwertrauschen-Kovarianz gegenübergestellt sind.

Insgesamt aber machten die stabilen Schätzungen in diesem Experiment nach etwa 80 Frames einen Anteil von weniger als 10% aller Schätzungen aus. Das liegt insbesondere daran, dass durch die Parametereinstellungen auch schlechte Schätzungen lange "überleben", bevor sie verworfen werden und stattdessen eine neue Schätzung initialisiert wird. Die Überlebenszeit von Schätzungen lässt sich leicht verkürzen, indem beispielsweise der `gap`-Parameter oder der Schwellwert für die Deskriptorenabstände verringert wird. Das ist jedoch nur bis zu einem gewissen Grad sinnvoll, denn davon sind auch die guten Schätzungen betroffen. Sinnvoller wäre es, ein Maß für die Qualität einer laufenden Schätzung zu finden, mit dem bereits nach wenigen Frames stabile und instabile Marker zuverlässig unterschieden werden können. Eine solche Unterscheidung ist aber nicht mit einem einzigen Kriterium möglich. In künftigen Arbeiten sollte man daher das Zusammenspiel mehrerer kritischer Eckdaten untersuchen und daraus einen kombinierten Schwellwert ableiten. Zu diesen Eckdaten könnten insbesondere die Systemkovarianz des Kalman Filters ($\boldsymbol{P}$), der mittlere Deskriptorabstand über mehrere Frames sowie das Alter der Schätzung in Frames zählen.

5.3 Schätzung der Kamerabewegung

In Abschnitt 3.7 wurde das Modell eines Erweiterten Kalman Filters zur Schätzung der Kamerabewegung mit Punktmerkmalen und Inertialsensordaten beschrieben. Diesen Kalman Filter implementiert die Klasse `CameraPoseEkf`, die in Abschnitt 4.4 erläutert wurde. Die Klasse wird in den nun folgenden Experimenten untersucht. Pro Durchlauf des in Bild 3.2 auf Seite 32 rechts dargestellten Zyklus wird eine Iteration des Filters durchlaufen. Dieser Zyklus wurde als Beispielprogramm implementiert, das die Klasse `CameraPoseEkf` in einer Schleife mit Sensordaten aus `InertialSimulation` und `PointDetectorSimulation` versorgt. Die Simulatoren arbeiten mit einer virtuellen Kamerabewegung, die sie aus einer Textdatei lesen. Die Datei ist auf der beiliegenden CD im Ordner `Programmcode/etc` enthalten (`moves.txt`). Das Beispielprogramm `simulateCamera` befindet sich im Unterordner `examples`.

Als erstes Experiment wurden zu der simulierten Kamerabewegung von den Simulatorenklassen Sensordaten ohne Rauschen und Ausreißer erzeugt. Die Messwertrauschen-

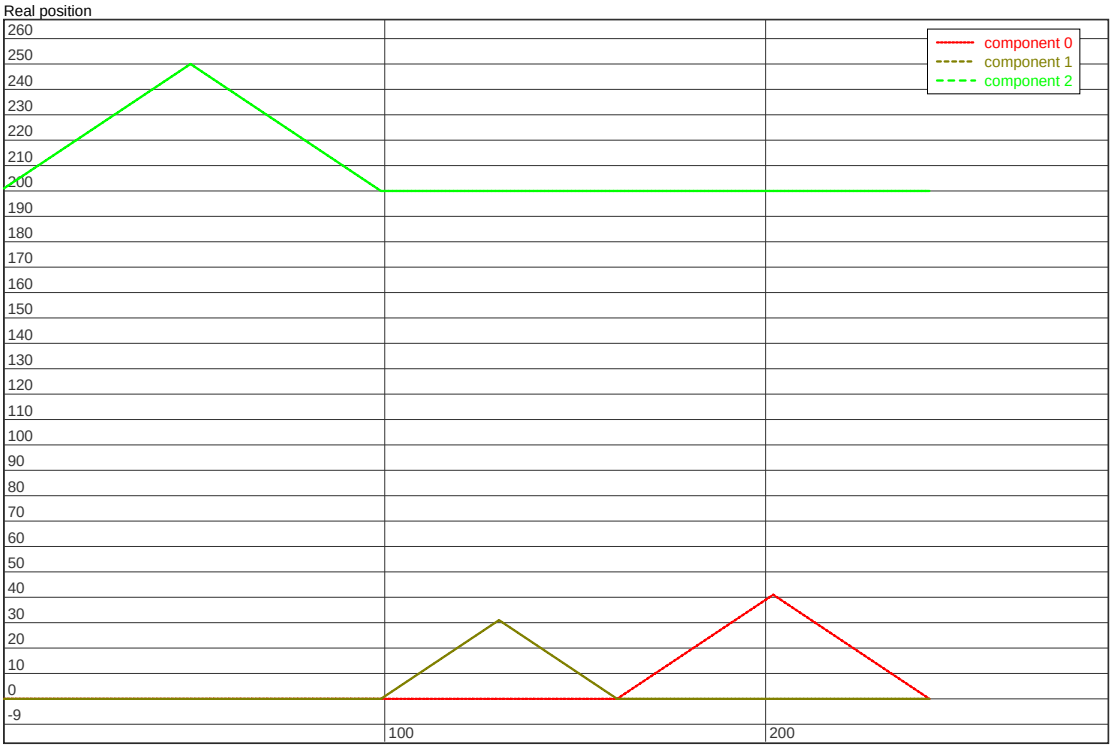


Bild 5.3: Simulierte Kamerabewegung zu den in Abschnitt 5.3 erläuterten Experimenten.

Kovarianz des Kalman Filters $\boldsymbol{R}$ wurde konstant gesetzt, während die Systemrauschen-Kovarianz $\boldsymbol{Q}$ mit jeweils einem Zehntel, einem Hundertstel und einem Tausendstel dieses Wertes variiert wurde. Die simulierte Kamerabewegung zeigt Bild 5.3. Diese Bewegung diente den Simulatoren als Grundlage zur Erzeugung der “künstlichen” Sensordaten. In Bild C.15 auf Seite 122 ist die Entwicklung der geschätzten Kamerabewegung für alle drei Fälle aufgeführt. In den Diagrammen wird erkennbar, dass der Filter schnell zu der tatsächlichen Kamerabewegung hin konvergiert. Die Verringerung der Systemrauschen-Kovarianz im Verhältnis zur Messwertrauschen-Kovarianz bewirkt eine Stabilisierung und Glättung der rekonstruierten Bewegung. Gleichzeitig erkennt man jedoch insbesondere an der z-Komponente (grüne Kurve), dass die glatteren Rekonstruktionen auch etwas langsamer konvergieren.

Anschließend wurde das Experiment mit einer Ausreißerrate des Punktdetektors von 0.05 durchgeführt, so dass durchschnittlich jede zwanzigste Punktkorrespondenz völlig falsch war. Hier wurde nun die Systemrauschen-Kovarianz auf einem konstanten Wert von $10{\cdot}\mathrm{e}^{-3}$ belassen und die Messwertrauschen-Kovarianz schrittweise erhöht (0.1, 10 und 1000). Da das Verhältnis zwischen $\boldsymbol{Q}$ und $\boldsymbol{R}$ ausschlaggebend für das Verhalten des Filters ist, ließe sich ein sehr ähnlicher Effekt auch durch Verringerung von $\boldsymbol{Q}$ erreichen. Die Ergebnisse zeigt Bild C.16 auf Seite 123 erneut durch die Funktionsplots der geschätzten Kamerabewegung. Die Ausreißer sind schwerwiegend und führen zu hohen Oszillationen der rekonstruierten Bewegung. Dennoch konvergiert der Filter offensichtlich in allen drei Fällen. Die z-Komponente leidet am stärksten unter den Ausreißern, da sie maßgeblich die Tiefeninformation enthält, die durch die Parallaxe der Punktkorrespondenzen schlechter bestimmt ist als die Rotation. Die Rotation hingegen schlägt sich in diesem Experiment maßgeblich in der x- und y-Komponente nieder. Auch in den Experimenten des vorangehenden Abschnitts war der größte Fehler in der z-Komponente beobachtet worden. Wie erwartet erzeugt die Erhöhung der Messwertrauschen-Kovarianz einen ähnlichen Effekt wie die Verringerung der Systemrauschen-Kovarianz: Die rekonstruierte Bewegung wird glatter und Ausreißer wirken sich deutlich weniger aus. Dafür muss eine erhöhte Trägheit des Filters hingenommen werden, so dass Veränderungen der Kameraposition erst etwas später abgebildet werden.

Ein weiteres Experiment simulierte statt Ausreißern normalverteiltes Rauschen in den Sensordaten. Dabei wurde genau wie im vorangehenden Beispiel die Messwertrauschen-Kovarianz variiert. Zunächst wurde normalverteiltes Rauschen auf die simulierten Punktkorrespondenzen addiert, so dass die Position korrespondierender Merkmale bis zu 10 Pixel vom korrekten Wert abwich (Bild C.17 auf Seite 124). In einem weiteren Testlauf wurde für die Simulation des Inertialsensors ein Rauschen aktiviert, welches die Koeffizienten in der Orientierungsmatrix zufällig mit Störungen im Bereich von 0.05 verändert (Bild C.18 auf Seite 125). Beide Testläufe liefern ähnliche Ergebnisse wie die vorangehenden Beispiele und unterstreichen die bereits beschriebenen Effekte.

Kapitel 6

Zusammenfassung

Im Rahmen dieser Diplomarbeit wurde ein umfangreiches Gesamtkonzept zur markerlosen Selbstlokalisation unter Verwendung einer Kamera und eines Inertialsensors erarbeitet. Dieses Konzept wurde in eine klare Struktur gebracht und ausführlich beschrieben. Es erfolgte eine Implementierung der beteiligten Prozesse als Klassenbibliothek in C++, die durch ihren modularen Aufbau sehr leicht erweitert werden kann. Diese Bibliothek wurde verwendet, um das Verhalten der Komponenten des Systems in Experimenten und Simulationen zu untersuchen. Die Ergebnisse wurden ausführlich beschrieben und um Anregungen für weiterführende Arbeiten ergänzt.

Das in dieser Arbeit entwickelte Verfahren zur Extraktion von SIFT-Merkmalen aus Bildern ist performant und bietet viele Möglichkeiten der dynamischen Anpassung an unterschiedliches Bildmaterial. Es wurden eine Vielzahl unterschiedlicher Parametereinstellungen für das Verfahren untersucht, und die beobachteten Effekte wurden bewertet. Um die Rekonstruktion weiter zu stabilisieren, könnte eine zusätzliche Subpixel-Lokalisation der Merkmale sinnvoll sein. Im Originalverfahren nach [Low04] erfolgt hierzu die Einpassung einer dreidimensionalen quadratischen Funktion im Skalenraum. Für erste Experimente würde auch eine zweidimensionale Verfeinerung ausreichen, wie sie von der Bibliothek *OpenCV* zur Verfügung gestellt wird.[1] Eine solche Verfeinerung wurde allerdings zugunsten der Performanz bewusst ausgelassen.

[1] Die entsprechende Methode in *OpenCV* heißt `cvFindCornerSubPix`.

Ein praktischer Einsatz des *Xsens MTx* ist im Rahmen dieser Arbeit noch nicht erfolgt. Die Probleme wurden in Abschnitt 4.2.2 im Zusammenhang der Implementierung beschrieben. Die Stabilität und Präzision der gelieferten Orientierungsdaten sollte auf jeden Fall nach präziser Montage und Kalibrierung mit der verwendeten Kamera erneut überprüft werden. Für die Experimente wurde hier die Simulation eines Inertialsensors benutzt, die dem System auch stark verrauschte Eingabedaten lieferte. Das System reagierte relativ robust auf diese Fehler in den Inertialdaten.

Die Ergebnisse der vorliegenden Arbeit lassen allerdings darauf schließen, dass ein Inertialsensor für die Selbstlokalisation nicht zwingend erforderlich ist. Bei den Experimenten in Kapitel 5 stellte sich wiederholt heraus, dass die Rekonstruktion bezüglich der Rotation stabiler ist als bezüglich der Translation. Dies zeigte sich z. B. durch stärkere Fehler bei der Rekonstruktion der z-Werte gegenüber den x- oder y-Werten. Bei der Herleitung des Rekonstruktionsmodells zur Initialisierung natürlicher Marker in Abschnitt 3.6 wurde darüber hinaus die Bedeutung von Punkten im Unendlichen erörtert, die zwar Information bezüglich der Orientierung, jedoch nicht bezüglich der Tiefe liefern. Diese Erkenntnisse deuten darauf hin, dass der kritische Kern des Rekonstruktionsproblems die Translationskomponente ist, zu deren Bestimmung die Inertialsensordaten kaum beitragen. Eine Variante der Erweiterten Kalman Filter, welche die Orientierung anstelle des Inertialsensors aus den Bildmerkmalen bestimmt, ließe sich für diesen Fall mit vertretbaren Anpassungen erstellen. Es müsste die Orientierung des in 3.7 beschriebenen Kalman Filter Modells aus der Zustandsübergangsmatrix $\boldsymbol{A}$ in den Systemzustand verschoben und zusätzlich die Winkelbeschleunigung modelliert werden. Das Modell wäre dann dem aus [Dav03] sehr ähnlich.

Das Problem der markerlosen Selbstlokalisation in Außenszenen ist heute ohne die Verwendung weiteren Vorwissens noch nicht leicht zu lösen. Diese Arbeit zeigt aber, dass die Teilprobleme theoretisch gelöst werden können, und dass sich im Fall idealer Sensordaten auch jetzt schon brauchbare Ergebnisse erzielen lassen. Dennoch ist ein hoher Aufwand erforderlich, um die Parametereinstellungen der zum Einsatz kommenden Komponenten an die realen Eingabedaten anzupassen. Im Bereich der Zuordnung von Punktmerkmalen muss z. B. ein Gleichgewicht zwischen Performanz und Präzision gefunden werden. Mit dem Konzept dynamischer Deskriptoren stellt diese Arbeit Möglichkeiten dafür bereit, die

bereits im Experiment untersucht wurden. Für die Erweiterten Kalman Filter bedeutet eine Anpassung an die Eingabedaten in erster Linie eine sehr genaue Einstellung der System- und Messwertrauschen-Kovarianzen in jeder Iteration. Hierzu wurden konkrete Vorschläge gemacht. Durch die Entwicklung von Sensordaten-Simulatoren sowie einer Klasse zur grafischen Auswertung der Zustände in den Kalman Filtern stehen komfortable Werkzeuge zur Verfügung, um weitere Experimente durchzuführen. Im Anschluss an diese Arbeit wäre eine umfangreiche Auswertung zur Einstellung dieser Kovarianzen empfehlenswert. Ideal wäre der Entwurf eines separaten statistischen Modells zur Optimierung dieser Parameter, das selbst als Kalman Filter realisiert ist. Eine solche Parameteroptimierung bezeichnet man als *Systemidentifikation*.

Obwohl der in Abschnitt 2.3.1 entwickelte algebraische Lösungsansatz in dieser Arbeit nicht für die Umsetzung ausgewählt wurde, bietet er interessante Möglichkeiten zur nachträglichen Ergänzung des vorhandenen Systems. Da im Unterschied zu dem verwendeten probabilistischen Ansatz keine kumulierten Fehler (*Drift*) zu erwarten sind, und das algebraische Verfahren isoliert auf wenige Frames angewendet werden kann, ließen sich damit gut sporadische Stabilisierungen des Gesamtsystems erreichen.

Anhang A

Symbole und Abkürzungen

A.1 Verwendete Symbole

μ, ν repräsentieren die x- und y-Position (also Spalte und Reihe) der Projektion eines Weltpunkts in einem digitalen Bild.

θ wird im Zusammenhang mit der Parametrisierung natürlicher Marker durch inverse Tiefe auf einer Halbgeraden im Raum benutzt und repräsentiert den *Drehwinkel* der Halbgeraden im Kamerakoordinatensystem, d. h. den Winkel auf dem Scheitelkreis bezogen auf das optische Kamerazentrum.

ϕ wird im Zusammenhang mit der Parametrisierung natürlicher Marker durch inverse Tiefe auf einer Halbgeraden im Raum benutzt und repräsentiert den *Kippwinkel* der Halbgeraden im Kamerakoordinatensystem, d. h. die Steigung des Strahls relativ zur horizontalen x/z-Ebene.

$\boldsymbol{A}$ ist die Zustandsübergangsmatrix in einem Kalman Filter.

c_x, c_y repräsentieren die vertikale und horizontale Position des Hauptpunkts, der zu den intrinsischen Kameraparametern gehört.

f_x, f_y repräsentieren die vertikale und horizontale Brennweite der Kamera, die bereits mit der physikalischen Pixelgrösse (dx) diskretisiert wurden. Sie gehören zu den intrinsischen Kameraparametern.

$\boldsymbol{K}$ ist die Gewichtungsmatrix in einem Kalman Filter (*Kalman-Gain*).

$\boldsymbol{O}$ ist die Orientierung einer Kamera. Sie ist die Transponierte der Rotation $\boldsymbol{R}$.

$\boldsymbol{P}_t$ ist die System-Kovarianzmatrix in einem Kalman Filter zu einem Zeitpunkt t.

$\boldsymbol{Q}$ ist die Systemrauschen-Kovarianzmatrix in einem Kalman Filter.

$\boldsymbol{R}$ ist die Messwertrauschen-Kovarianzmatrix in einem Kalman Filter. Im Zusammenhang mit den extrinsischen Kameraparametern wird $\boldsymbol{R}$ als Bezeichnung für die Rotation der Kamera benutzt.

$\boldsymbol{r}$ ist die Position der Kamera. Sie ist nicht identisch mit der Translation $\boldsymbol{t}$ (siehe Abschnitt 2.1.1).

r_{ij} ist die Kurzschreibweise für den Koeffizienten der Kamerarotation $\boldsymbol{R}$ in Zeile i und Spalte j.

$\boldsymbol{t}$ ist der Translationsvektor der Kamera. Er ist nicht identisch mit der Kameraposition $\boldsymbol{r}$ (siehe Abschnitt 2.1.1).

$\boldsymbol{y}_i$ bezeichnet einen natürlichen Marker in euklidischer 3D-Repräsentation an der Position i in der Karte.

$\boldsymbol{w}$ ist ein Vektor, der das unbekannte Rauschen des Systemmodells eines Kalman Filters repräsentiert.

Alle hier nicht aufgeführten Symbole sind direkt im Text in einem vorangehenden oder nachfolgenden Absatz erläutert.

A.2 Abkürzungsverzeichnis

AR steht für *Augmented Reality* und wird im Deutschen mit “Erweiterte Realität” übersetzt. Der Begriff fasst Techniken zusammen, die dem Benutzer virtuelle Informationen in die primär visuelle Wahrnehmung der realen Welt einblenden.

COM ist das *Component Object Model* der *Microsoft Windows* Plattform. Dabei handelt es sich um eine proprietäre Technik auf Betriebssystemebene zur Bereitstellung von Klassen aus dynamisch gelinkten Bibliotheken.

EKF ist die Abkürzung für “Erweiterter Kalman Filter” oder *Extended Kalman Filter*.

GPS ist die weitläufige Bezeichnung für satellitengestützte Navigationssysteme (*Global Positioning System*).

SIFT steht für *Scale Invariant Feature Transform* und bezeichnet ein Verfahren zur Extraktion von Interessenspunkten aus Bildern [Low04].

SLAM (*Simultaneous Localization and Mapping*) bezeichnet die insbesondere in der Robotik zum Einsatz kommende Disziplin, parallel zur Selbstlokalisation eine Karte der Umgebung zu erstellen.

STL ist eine Abkürzung für *Standard Template Library*. Obwohl der Begriff anderen Ursprungs ist, sind damit meist Komponenten der C++-Standardbibliothek gemeint, die auf generischer Programmierung basieren. Ein bekanntes Beispiel sind Containerklassen wie `std::vector`.

UKF ist die Abkürzung für *Unscented Kalman Filter* [JU97].

WLAN steht für *Wireless Local Area Network* und ist eine Kurzbezeichnung für lokale Funknetze. Diese basieren meist auf dem Standard 802.11 des *Institute of Electrical and Electronics Engineers* (IEEE).

Anhang B

Berechnungen

B.1 Ableitung der Messwertgleichung aus Abschnitt 3.6.4

Im Folgenden sind die einzelnen partiellen Ableitungen der Jacobi-Matrix aufgeführt, die in Abschnitt 3.6.4 beschrieben wird. Zunächst die partiellen Ableitungen bezüglich der Berechnung der Spaltenpositionen h_{μ_i}:

$$
\begin{aligned}
\frac{\partial h_{\mu_i}}{\partial \hat{r}_{i_x}} &= \frac{\left(f_x r_{11} + c_x r_{31}\right) p_z^P - r_{31} p_x^P}{\left(p_z^P\right)^2} \\
\frac{\partial h_{\mu_i}}{\partial \hat{r}_{i_y}} &= \frac{\left(f_x r_{12} + c_x r_{32}\right) p_z^P - r_{32} p_x^P}{\left(p_z^P\right)^2} \\
\frac{\partial h_{\mu_i}}{\partial \hat{r}_{i_z}} &= \frac{\left(f_x r_{13} + c_x r_{33}\right) p_z^P - r_{33} p_x^P}{\left(p_z^P\right)^2} \\
\frac{\partial h_{\mu_i}}{\partial \hat{\theta}_i} &= \left(\frac{f_x}{\hat{\rho}_i} \begin{pmatrix} r_{11} \\ r_{12} \\ r_{13} \end{pmatrix}^T + \frac{c_x}{\hat{\rho}_i} \begin{pmatrix} r_{31} \\ r_{32} \\ r_{33} \end{pmatrix}^T \right) \begin{pmatrix} -\sin\hat{\theta}_i \cdot \cos\hat{\phi}_i \\ -\sin\hat{\theta}_i \cdot \sin\hat{\phi}_i \\ \cos\hat{\theta}_i \end{pmatrix} \frac{1}{p_z^P} \\
&\quad - \frac{1}{\hat{\rho}_i} \begin{pmatrix} r_{31} \\ r_{32} \\ r_{33} \end{pmatrix}^T \begin{pmatrix} -\sin\hat{\theta}_i \cdot \cos\hat{\phi}_i \\ -\sin\hat{\theta}_i \cdot \sin\hat{\phi}_i \\ \cos\hat{\theta}_i \end{pmatrix} \frac{p_x^P}{\left(p_z^P\right)^2} \\
\frac{\partial h_{\mu_i}}{\partial \hat{\phi}_i} &= \left(\frac{f_x}{\hat{\rho}_i} \begin{pmatrix} r_{11} \\ r_{12} \end{pmatrix}^T + \frac{c_x}{\hat{\rho}_i} \begin{pmatrix} r_{31} \\ r_{32} \end{pmatrix}^T \right) \begin{pmatrix} -\sin\hat{\phi}_i \cdot \cos\hat{\theta}_i \\ \cos\hat{\phi}_i \cdot \cos\hat{\theta}_i \end{pmatrix} \frac{1}{p_z^P} \\
&\quad - \frac{1}{\hat{\rho}_i} \begin{pmatrix} r_{31} \\ r_{32} \end{pmatrix}^T \begin{pmatrix} -\sin\hat{\phi}_i \cdot \cos\hat{\theta}_i \\ \cos\hat{\phi}_i \cdot \cos\hat{\theta}_i \end{pmatrix} \frac{p_x^P}{\left(p_z^P\right)^2} \\
\frac{\partial h_{\mu_i}}{\partial \hat{\rho}_i} &= \left(-\frac{f_x}{\hat{\rho}_i^2} \begin{pmatrix} r_{11} \\ r_{12} \\ r_{13} \end{pmatrix}^T - \frac{c_x}{\hat{\rho}_i^2} \begin{pmatrix} r_{31} \\ r_{32} \\ r_{33} \end{pmatrix}^T \right) \begin{pmatrix} \cos\hat{\theta}_i \cdot \cos\hat{\phi}_i \\ \cos\hat{\theta}_i \cdot \sin\hat{\phi}_i \\ \sin\hat{\theta}_i \end{pmatrix} \frac{1}{p_z^P} \\
&\quad + \frac{1}{\hat{\rho}_i^2} \begin{pmatrix} r_{31} \\ r_{32} \\ r_{33} \end{pmatrix}^T \begin{pmatrix} \cos\hat{\theta}_i \cdot \cos\hat{\phi}_i \\ \cos\hat{\theta}_i \cdot \sin\hat{\phi}_i \\ \sin\hat{\theta}_i \end{pmatrix} \frac{p_x^P}{\left(p_z^P\right)^2}
\end{aligned}
$$

Analog dazu die partiellen Ableitungen bezüglich der Berechnung der Reihenpositionen h_{ν_i}:

$$
\begin{aligned}
\frac{\partial h_{\nu_i}}{\partial \hat{r}_{i_x}} &= \frac{\left(f_y r_{21} + c_y r_{31}\right) p_z^P - r_{31} p_y^P}{\left(p_z^P\right)^2} \\
\frac{\partial h_{\nu_i}}{\partial \hat{r}_{i_y}} &= \frac{\left(f_y r_{22} + c_y r_{32}\right) p_z^P - r_{32} p_y^P}{\left(p_z^P\right)^2} \\
\frac{\partial h_{\nu_i}}{\partial \hat{r}_{i_z}} &= \frac{\left(f_y r_{23} + c_y r_{33}\right) p_z^P - r_{33} p_y^P}{\left(p_z^P\right)^2} \\
\frac{\partial h_{\nu_i}}{\partial \hat{\theta}_i} &= \left(\frac{f_y}{\hat{\rho}_i} \begin{pmatrix} r_{21} \\ r_{22} \\ r_{23} \end{pmatrix}^T + \frac{c_y}{\hat{\rho}_i} \begin{pmatrix} r_{31} \\ r_{32} \\ r_{33} \end{pmatrix}^T \right) \begin{pmatrix} -\sin\hat{\theta}_i \cdot \cos\hat{\phi}_i \\ -\sin\hat{\theta}_i \cdot \sin\hat{\phi}_i \\ \cos\hat{\theta}_i \end{pmatrix} \frac{1}{p_z^P} \\
&\quad - \frac{1}{\hat{\rho}_i} \begin{pmatrix} r_{31} \\ r_{32} \\ r_{33} \end{pmatrix}^T \begin{pmatrix} -\sin\hat{\theta}_i \cdot \cos\hat{\phi}_i \\ -\sin\hat{\theta}_i \cdot \sin\hat{\phi}_i \\ \cos\hat{\theta}_i \end{pmatrix} \frac{p_y^P}{\left(p_z^P\right)^2} \\
\frac{\partial h_{\nu_i}}{\partial \hat{\phi}_i} &= \left(\frac{f_y}{\hat{\rho}_i} \begin{pmatrix} r_{21} \\ r_{22} \end{pmatrix}^T + \frac{c_y}{\hat{\rho}_i} \begin{pmatrix} r_{31} \\ r_{32} \end{pmatrix}^T \right) \begin{pmatrix} -\sin\hat{\phi}_i \cdot \cos\hat{\theta}_i \\ \cos\hat{\phi}_i \cdot \cos\hat{\theta}_i \end{pmatrix} \frac{1}{p_z^P} \\
&\quad - \frac{1}{\hat{\rho}_i} \begin{pmatrix} r_{31} \\ r_{32} \end{pmatrix}^T \begin{pmatrix} -\sin\hat{\phi}_i \cdot \cos\hat{\theta}_i \\ \cos\hat{\phi}_i \cdot \cos\hat{\theta}_i \end{pmatrix} \frac{p_y^P}{\left(p_z^P\right)^2} \\
\frac{\partial h_{\nu_i}}{\partial \hat{\rho}_i} &= \left(-\frac{f_y}{\hat{\rho}_i^2} \begin{pmatrix} r_{21} \\ r_{22} \\ r_{23} \end{pmatrix}^T - \frac{c_y}{\hat{\rho}_i^2} \begin{pmatrix} r_{31} \\ r_{32} \\ r_{33} \end{pmatrix}^T \right) \begin{pmatrix} \cos\hat{\theta}_i \cdot \cos\hat{\phi}_i \\ \cos\hat{\theta}_i \cdot \sin\hat{\phi}_i \\ \sin\hat{\theta}_i \end{pmatrix} \frac{1}{p_z^P} \\
&\quad + \frac{1}{\hat{\rho}_i^2} \begin{pmatrix} r_{31} \\ r_{32} \\ r_{33} \end{pmatrix}^T \begin{pmatrix} \cos\hat{\theta}_i \cdot \cos\hat{\phi}_i \\ \cos\hat{\theta}_i \cdot \sin\hat{\phi}_i \\ \sin\hat{\theta}_i \end{pmatrix} \frac{p_y^P}{\left(p_z^P\right)^2}
\end{aligned}
$$

B.2 Ableitung der Messwertgleichung aus Abschnitt 3.7.1

Hier wird die Jacobi-Matrix mit den partiellen Ableitungen der Messwertgleichung aus Abschnitt 3.7.1 aufgeführt. Die Matrix wird aus Platzgründen transponiert dargestellt. Zur vereinfachten Darstellung werden die Zähler von h_{μ_i} mit n_{μ_i} (n wie *numerator*) und die Nenner mit d_{μ_i} (d wie *denominator*) abgekürzt; analog dazu n_{ν_i} und d_{ν_i}.

$$
\begin{pmatrix}
\frac{f_x d_{\mu_1}}{d_{\mu_1}^2} & 0 & \dots & \frac{f_x d_{\mu_n}}{d_{\mu_n}^2} & 0 \\
0 & \frac{f_y d_{\nu_1}}{d_{\nu_1}^2} & \dots & 0 & \frac{f_y d_{\nu_n}}{d_{\nu_n}^2} \\
\frac{c_x d_{\mu_1} - n_{\mu_1}}{d_{\mu_1}^2} & \frac{c_y d_{\nu_1} - n_{\nu_1}}{d_{\nu_1}^2} & \dots & \frac{c_x d_{\mu_n} - n_{\mu_n}}{d_{\mu_n}^2} & \frac{c_y d_{\nu_n} - n_{\nu_n}}{d_{\nu_n}^2} \\
\frac{f_x \Delta_t d_{\mu_1}}{d_{\mu_1}^2} & 0 & \dots & \frac{f_x \Delta_t d_{\mu_n}}{d_{\mu_n}^2} & 0 \\
0 & \frac{f_y \Delta_t d_{\nu_1}}{d_{\nu_1}^2} & \dots & 0 & \frac{f_y \Delta_t d_{\nu_n}}{d_{\nu_n}^2} \\
\frac{c_x \Delta_t d_{\mu_1} - \Delta_t n_{\mu_1}}{d_{\mu_1}^2} & \frac{c_y \Delta_t d_{\nu_1} - \Delta_t n_{\nu_1}}{d_{\nu_1}^2} & \dots & \frac{c_x \Delta_t d_{\mu_n} - \Delta_t n_{\mu_n}}{d_{\mu_n}^2} & \frac{c_y \Delta_t d_{\nu_n} - \Delta_t n_{\nu_n}}{d_{\nu_n}^2} \\
\frac{p_{11} d_{\mu_1} - r_{31} n_{\mu_1}}{d_{\mu_1}^2} & \frac{p_{21} d_{\nu_1} - r_{31} n_{\nu_1}}{d_{\nu_1}^2} & \dots & 0 & 0 \\
\frac{p_{12} d_{\mu_1} - r_{32} n_{\mu_1}}{d_{\mu_1}^2} & \frac{p_{22} d_{\nu_1} - r_{32} n_{\nu_1}}{d_{\nu_1}^2} & \dots & 0 & 0 \\
\frac{p_{13} d_{\mu_1} - r_{33} n_{\mu_1}}{d_{\mu_1}^2} & \frac{p_{23} d_{\nu_1} - r_{33} n_{\nu_1}}{d_{\nu_1}^2} & \dots & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
0 & 0 & \dots & \frac{p_{11} d_{\mu_n} - r_{31} n_{\mu_n}}{d_{\mu_n}^2} & \frac{p_{21} d_{\nu_n} - r_{31} n_{\nu_n}}{d_{\nu_n}^2} \\
0 & 0 & \dots & \frac{p_{12} d_{\mu_n} - r_{32} n_{\mu_n}}{d_{\mu_n}^2} & \frac{p_{22} d_{\nu_n} - r_{32} n_{\nu_n}}{d_{\nu_n}^2} \\
0 & 0 & \dots & \frac{p_{13} d_{\mu_n} - r_{33} n_{\mu_n}}{d_{\mu_n}^2} & \frac{p_{23} d_{\nu_n} - r_{33} n_{\nu_n}}{d_{\nu_n}^2}
\end{pmatrix}^T
$$

Anhang C

Bilder und grafische Auswertungen

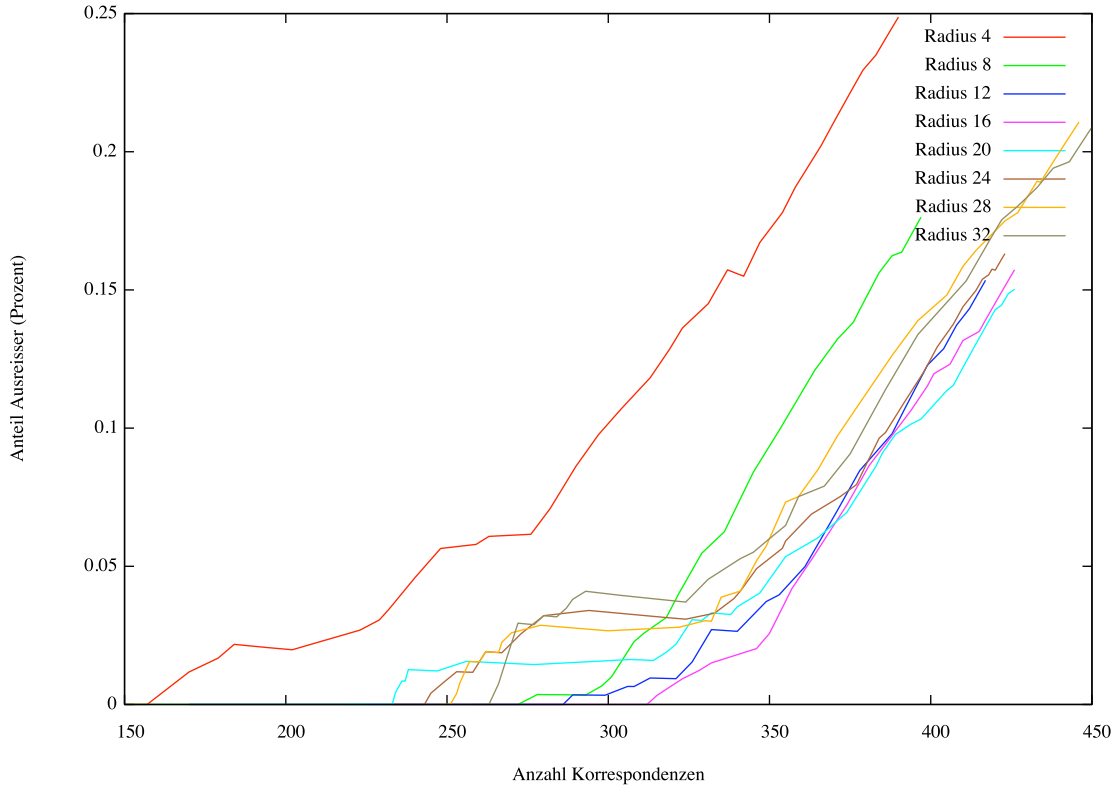


Bild C.1: Plot zum Basisexperiment aus Abschnitt 5.1 bezüglich der Erhöhung des Ausreißeranteils von Punktmerkmalkorrespondenzen bei sinkendem Schwellwert des Deskriptorabstands. Dieses Experiment nutzt Deskriptoren mit 4 Teilkreisen, einem Ring und 16 Urnen pro Histogramm. Die Linien zeigen die Ergebnisse bei Variation des Radius.

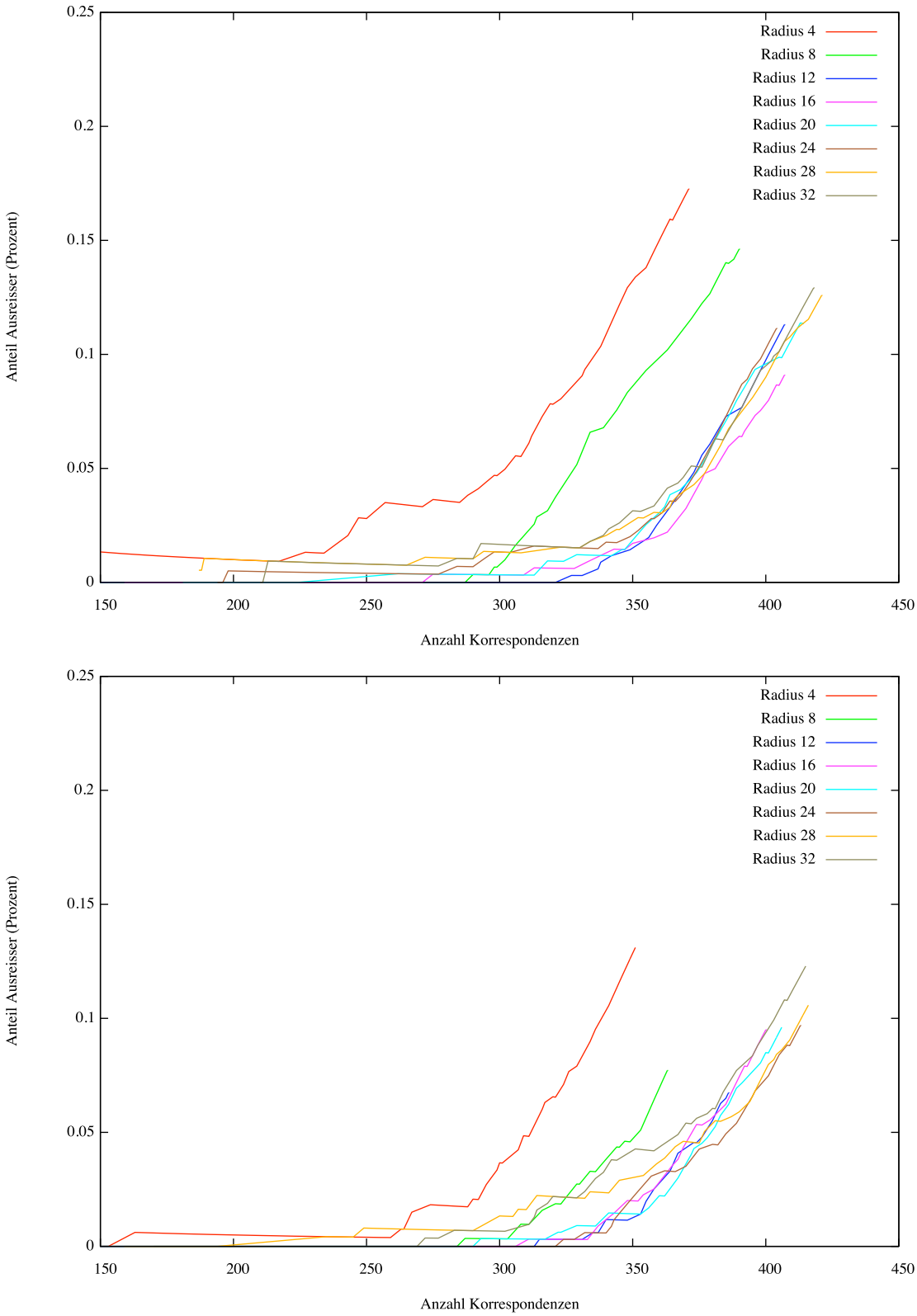


Bild C.2: Plot zum Experiment aus Abschnitt 5.1. Abweichend vom Basisexperiment (Bild C.1) wurden hier jeweils sechs statt vier Kreisteile (oben) und acht statt vier Kreisteile (unten) für die Deskriptoren verwendet.

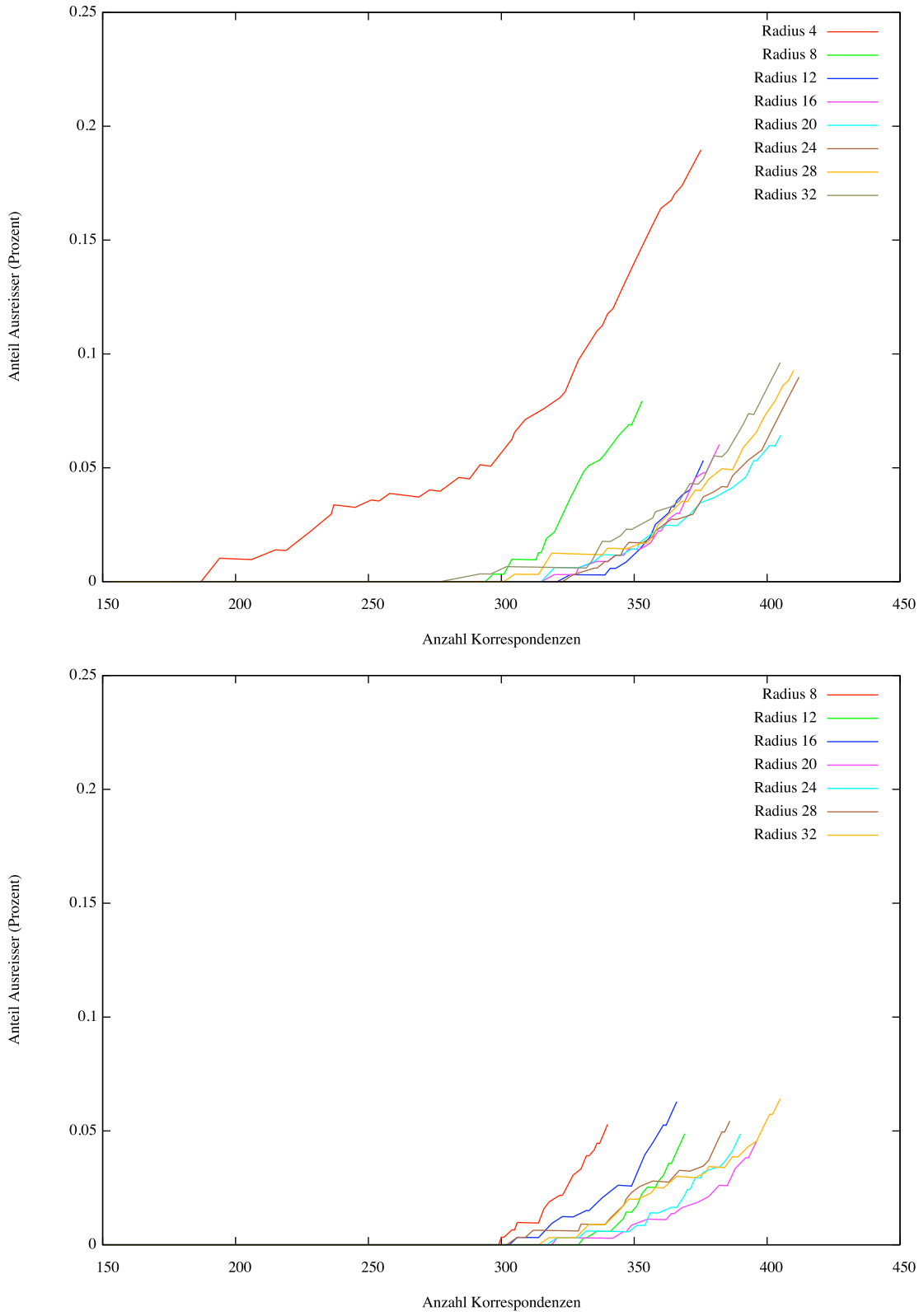


Bild C.3: Plot zum Experiment aus Abschnitt 5.1. Abweichend vom Basisexperiment (Bild C.1) wurden hier jeweils zwei statt einem Ring (oben) und drei statt einem Ring (unten) für die Deskriptoren verwendet.

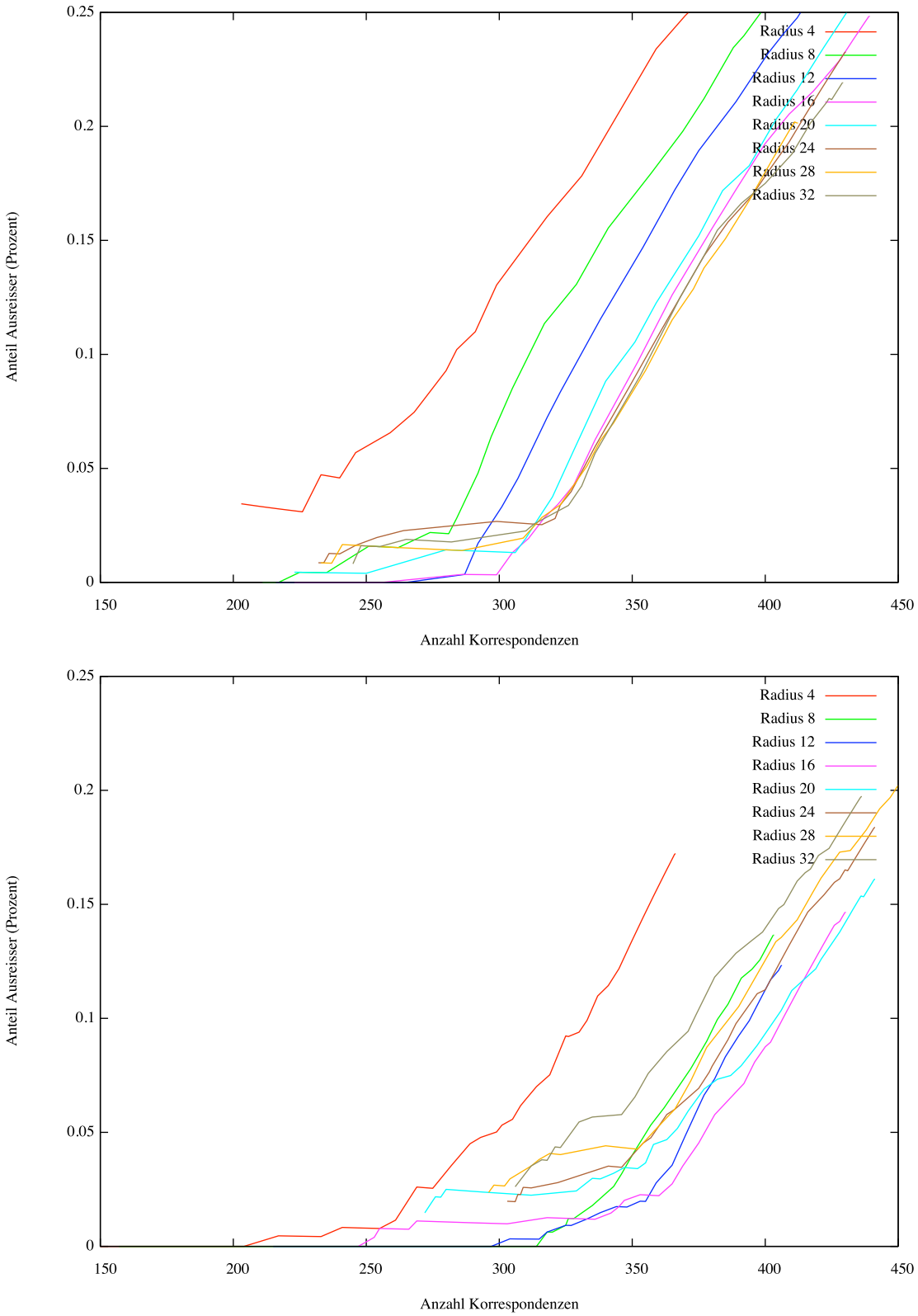


Bild C.4: Plot zum Experiment aus Abschnitt 5.1. Abweichend vom Basisexperiment (Bild C.1) wurden hier jeweils acht statt 16 Urnen (oben) und 32 statt 16 Urnen (unten) für die Deskriptoren verwendet.

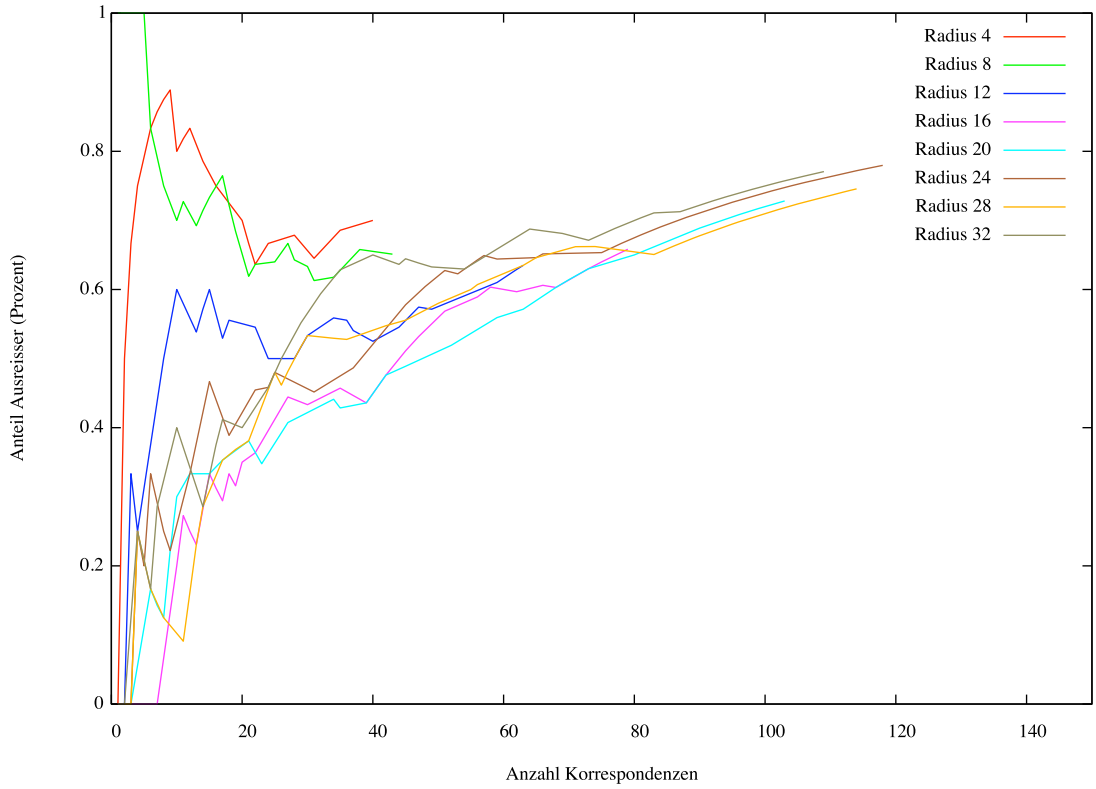


Bild C.5: Plot zum Experiment aus Abschnitt 5.1. Abweichend vom Basisexperiment (Bild C.1) wurde hier ein Bildpaar verwendet, das sich durch starke Unschärfe unterscheidet.

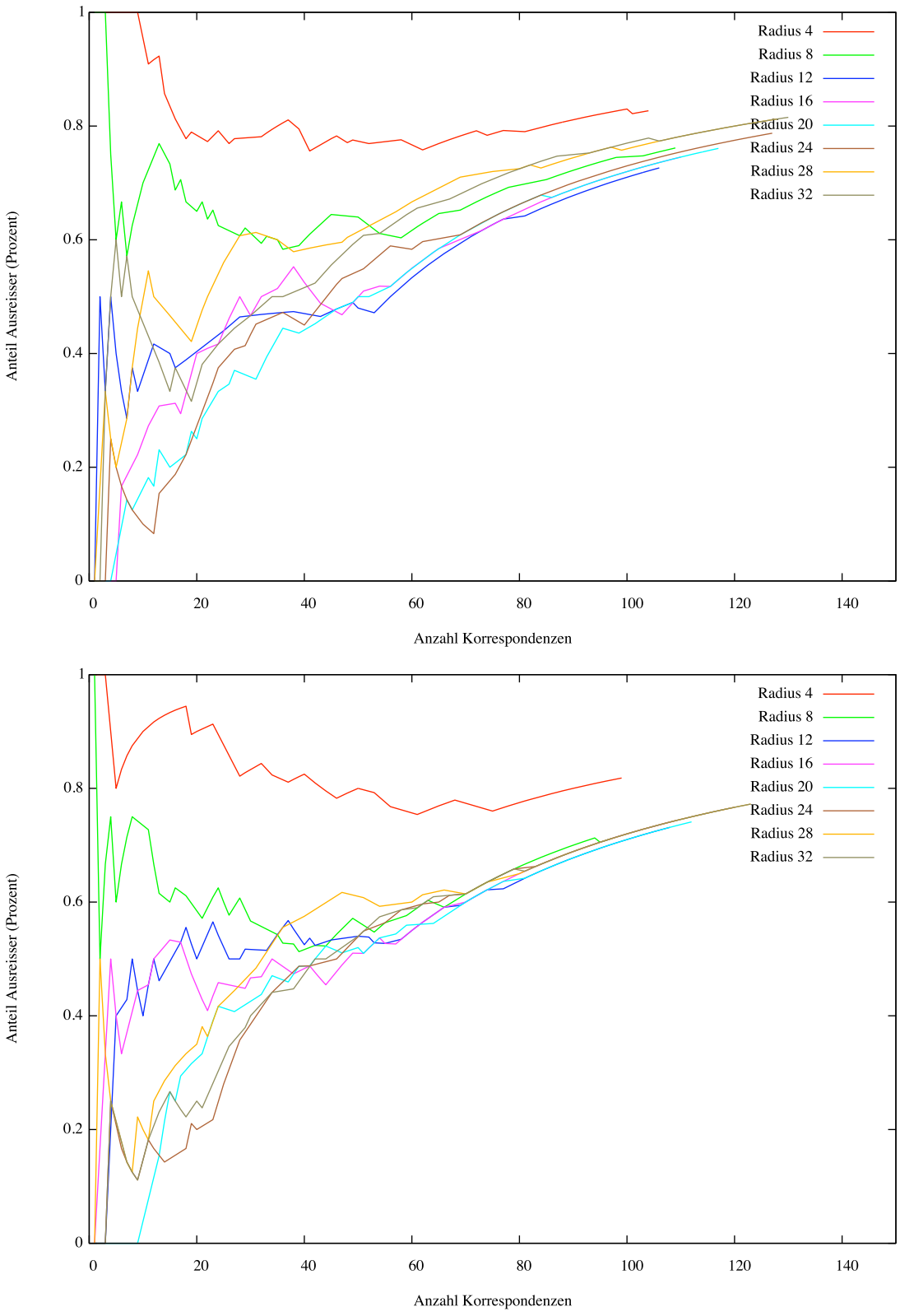
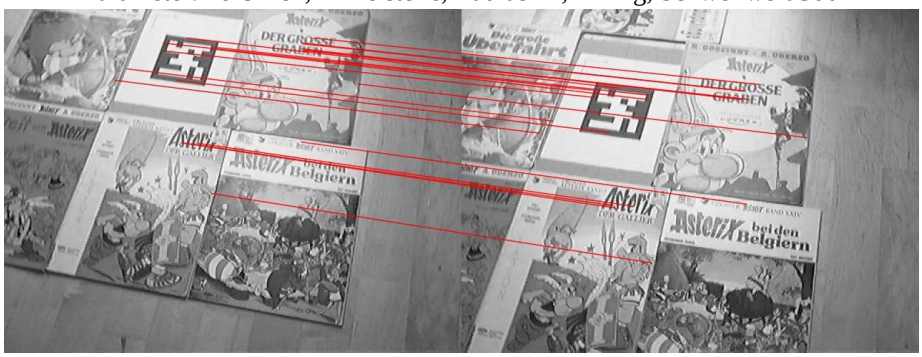
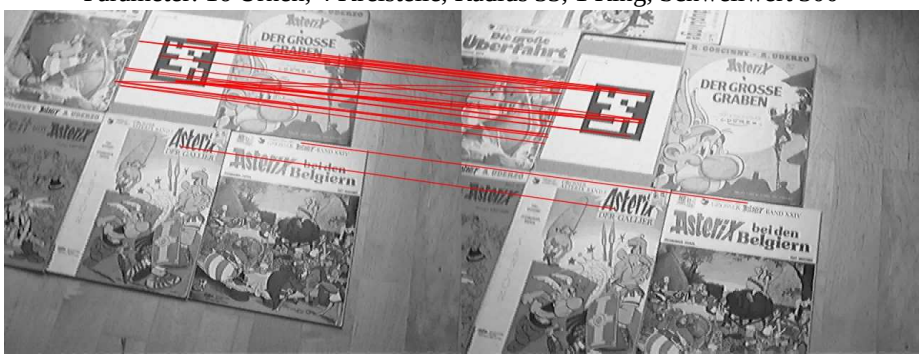


Bild C.6: Plot zum Experiment aus Abschnitt 5.1. Abweichend von dem Experiment in Bild C.5 wurden hier sechs statt vier Teilkreise (oben) und acht statt vier Teilkreise (unten) für die Deskriptoren verwendet.

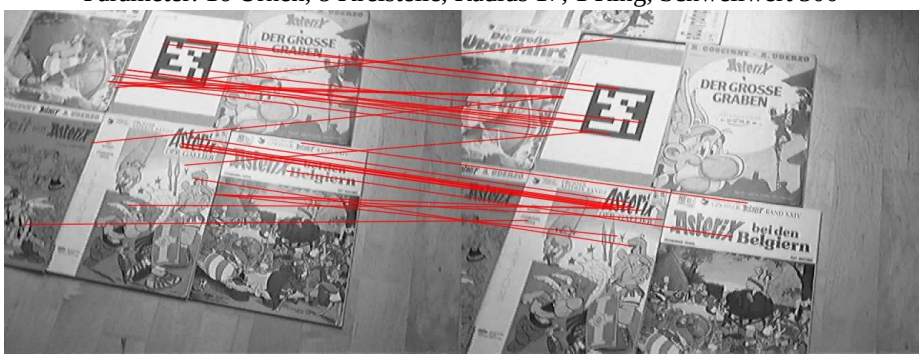
Parameter: 16 Urnen, 4 Kreisteile, Radius 17, 1 Ring, Schwellwert 300



Parameter: 16 Urnen, 4 Kreisteile, Radius 33, 1 Ring, Schwellwert 300



Parameter: 16 Urnen, 8 Kreisteile, Radius 17, 1 Ring, Schwellwert 500



Parameter: 8 Urnen, 4 Kreisteile, Radius 17, 1 Ring, Schwellwert 280

Bild C.7: Ergebnisse der Experimente aus 5.1 zur subjektiven Qualität des Merkmalabgleichs bei verschiedenen Parametereinstellungen zur Erzeugung von Deskriptoren.

Bild C.8: Funktionsplots zu den Experimenten aus Abschnitt 5.2.1 bezüglich des Fehlers bei der Schätzung einer 3D-Markerposition mit verrauschten Werten. Die Systemkovarianz beträgt oben $10 \cdot \mathrm{e}^{-3}$, in der Mitte $10 \cdot \mathrm{e}^{-6}$ und unten $10 \cdot \mathrm{e}^{-12}$ bei gleich bleibender Messwertrauschen-Kovarianz von 1.

Bild C.9: Funktionsplots zu den Experimenten aus Abschnitt 5.2.1 bezüglich des Fehlers bei der Schätzung einer 3D-Markerposition mit verrauschten Werten. Die Systemkovarianz beträgt konstant $10 \cdot \mathrm{e}^{-6}$, die Messwertrauschen-Kovarianz beträgt von oben nach unten 1, $10 \cdot \mathrm{e}^{-2}$ und $10 \cdot \mathrm{e}^{-3}$.

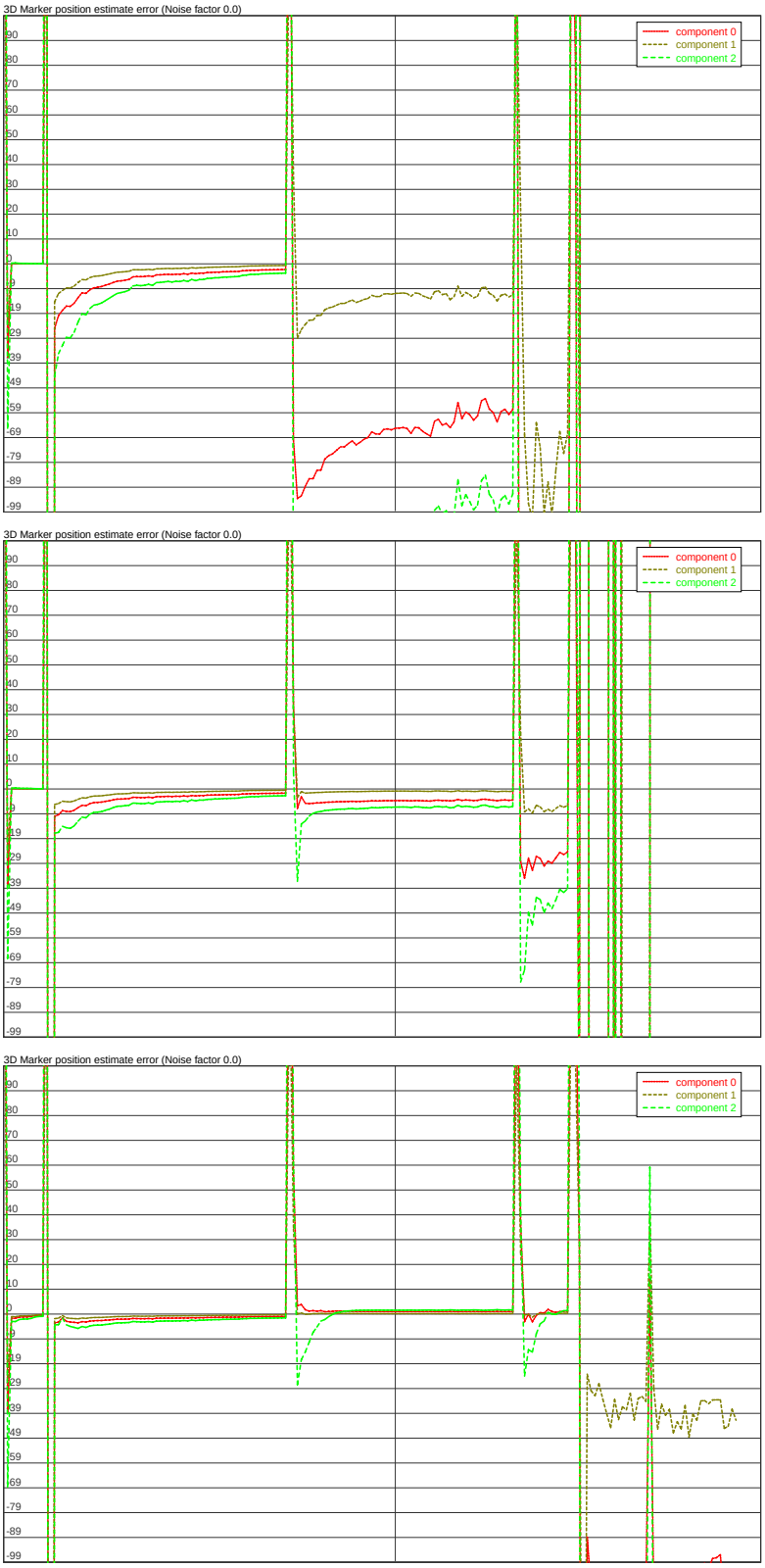


Bild C.10: Funktionsplots zu den Experimenten aus Abschnitt 5.2.1 bezüglich der Schätzung einer 3D-Markerposition mit Ausreißern. Die Messwertrauschen-Kovarianz bleibt konstant bei 1, während die Systemkovarianz von oben nach unten $10 \cdot \mathrm{e}^{-3}$, $10 \cdot \mathrm{e}^{-4}$ und $10 \cdot \mathrm{e}^{-5}$ beträgt.

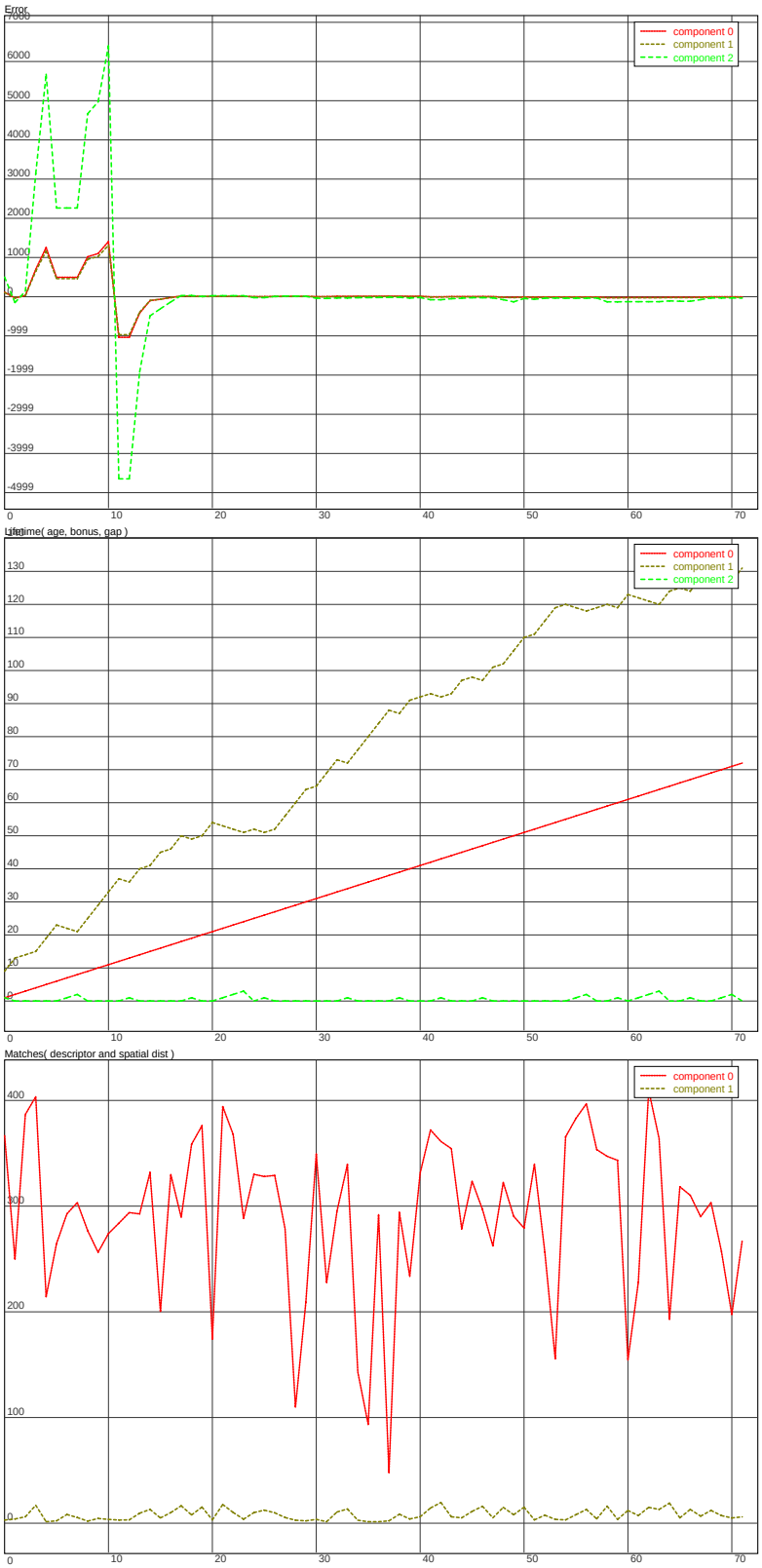


Bild C.11: Verlauf der Schätzung eines natürlichen Markers bei den Experimenten in Abschnitt 5.2.2.

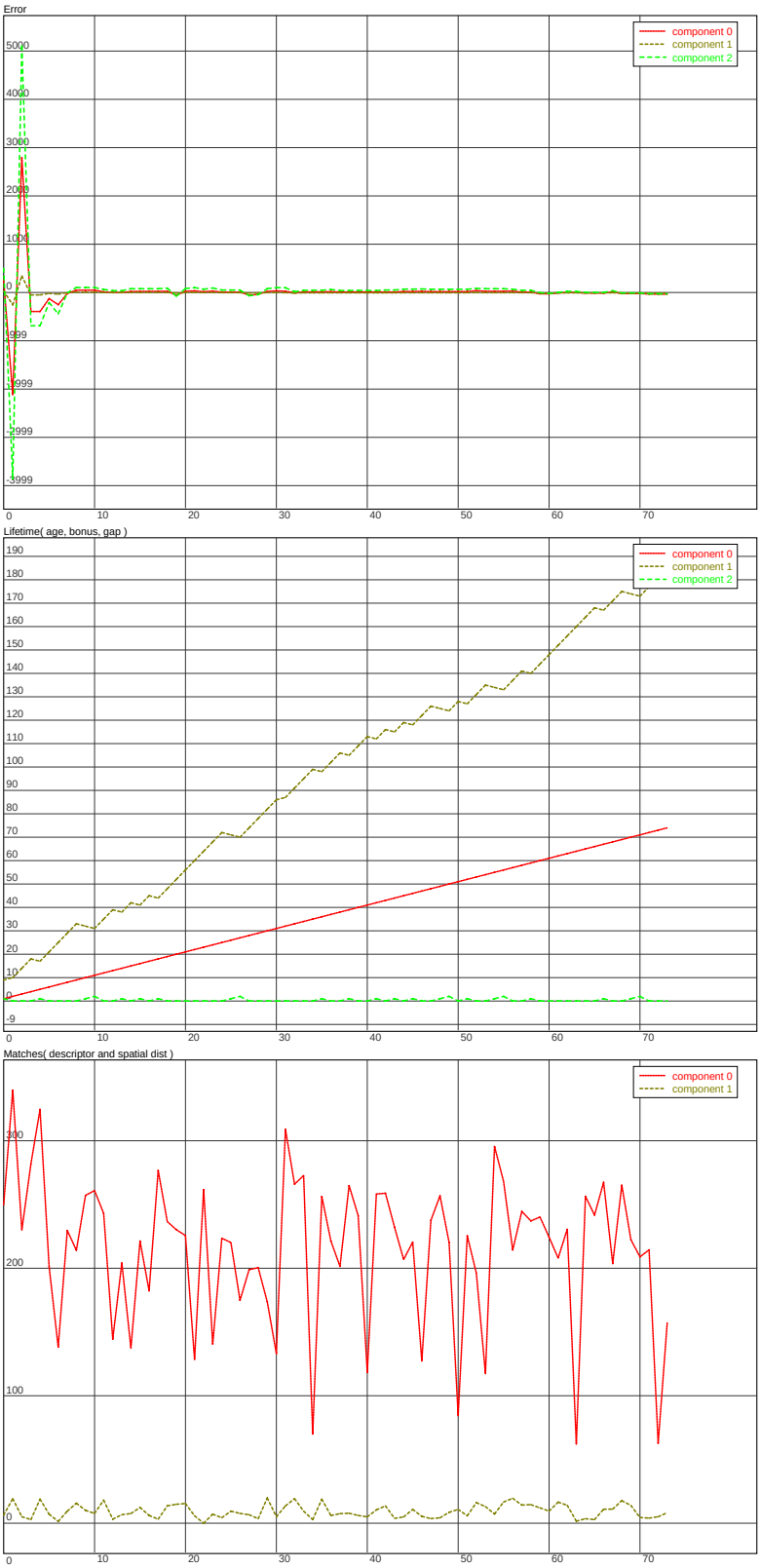


Bild C.12: Verlauf der Schätzung eines natürlichen Markers bei den Experimenten in Abschnitt 5.2.2.

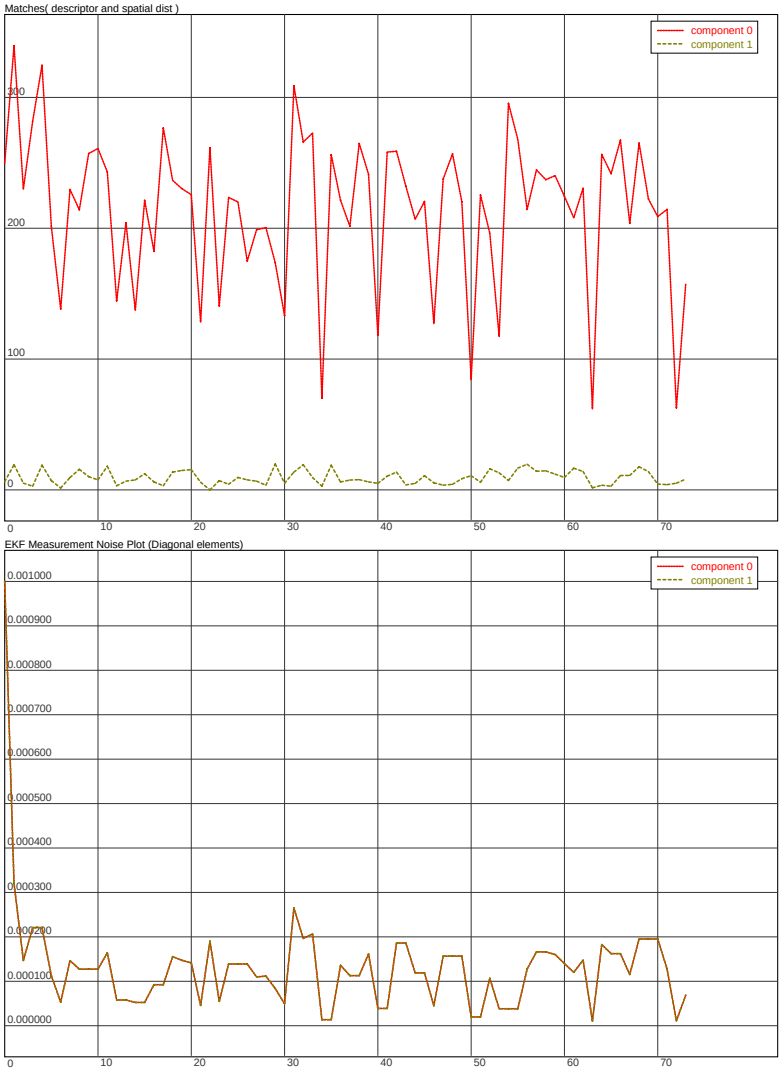


Bild C.13: Zusammenhang zwischen Deskriptorabstand und Messwertrauschen-Kovarianz während der Schätzung des natürlichen Markers aus Bild C.12

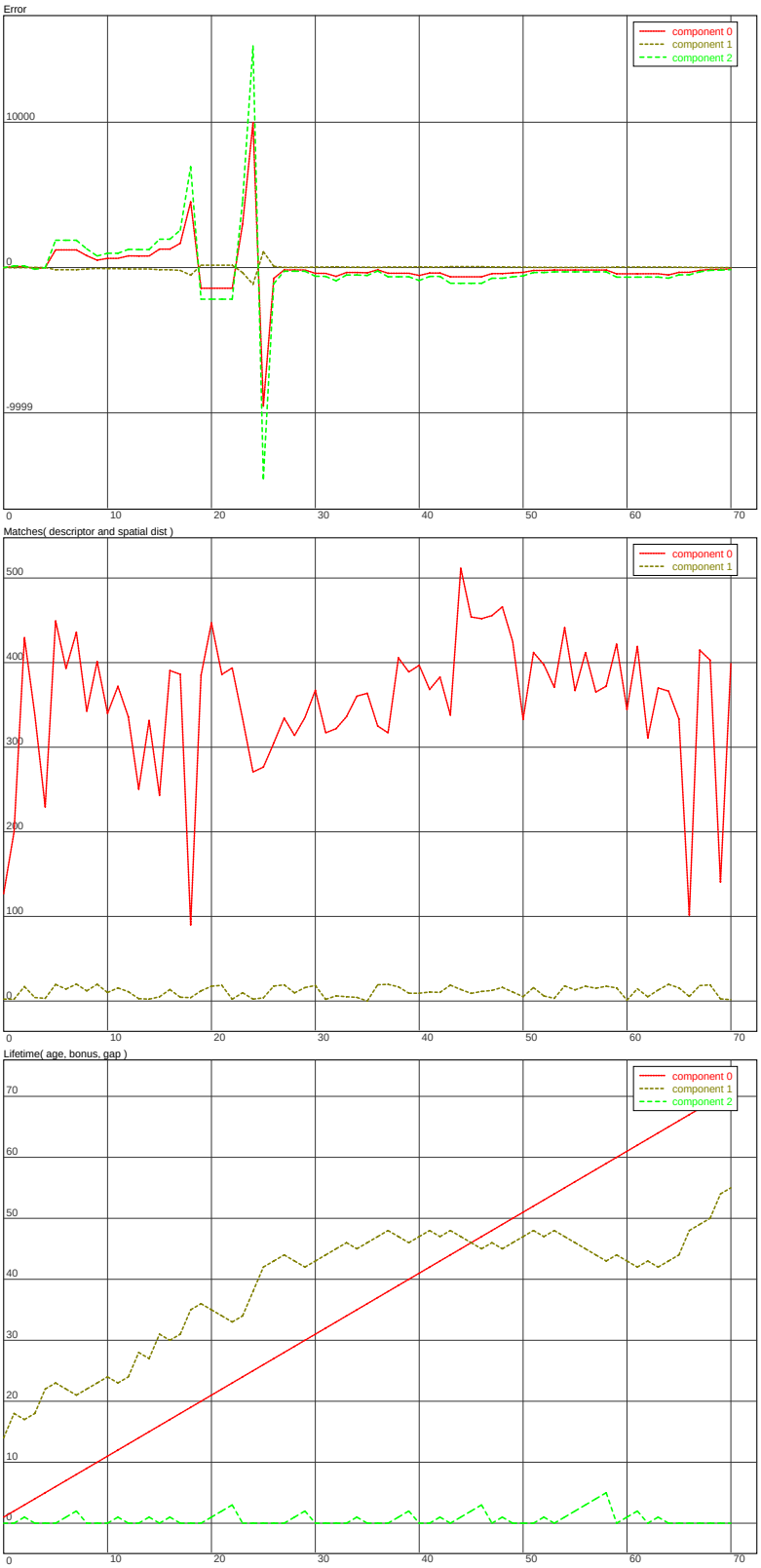


Bild C.14: Verlauf der Schätzung eines natürlichen Markers bei den Experimenten in 5.2.2.

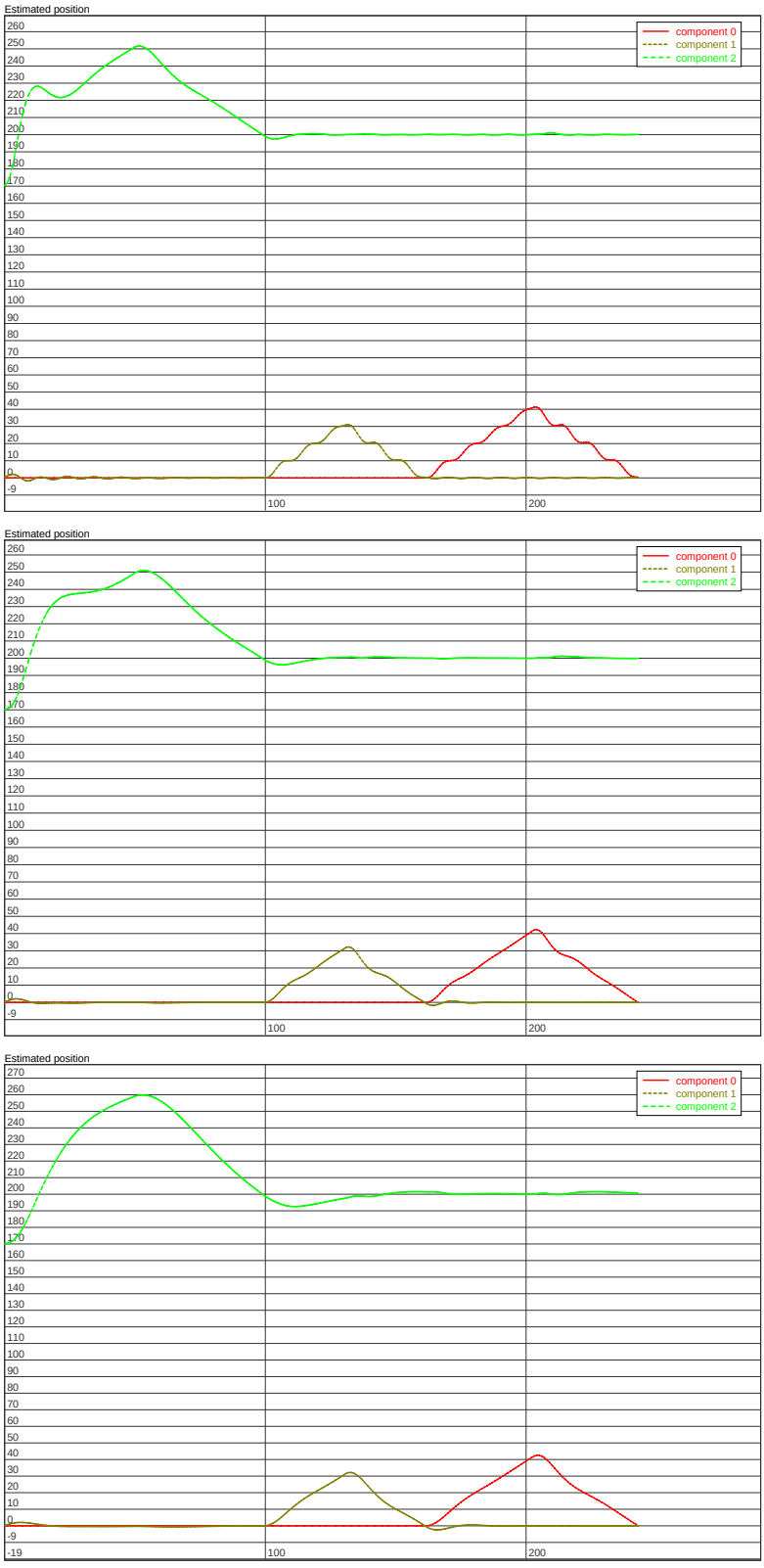


Bild C.15: Funktionsplots zum Experiment in Abschnitt 5.3 zur Rekonstruktion der Kamerabewegung ohne Rauschen und Ausreißer unter Variation der Systemrauschen-Kovarianz Q.

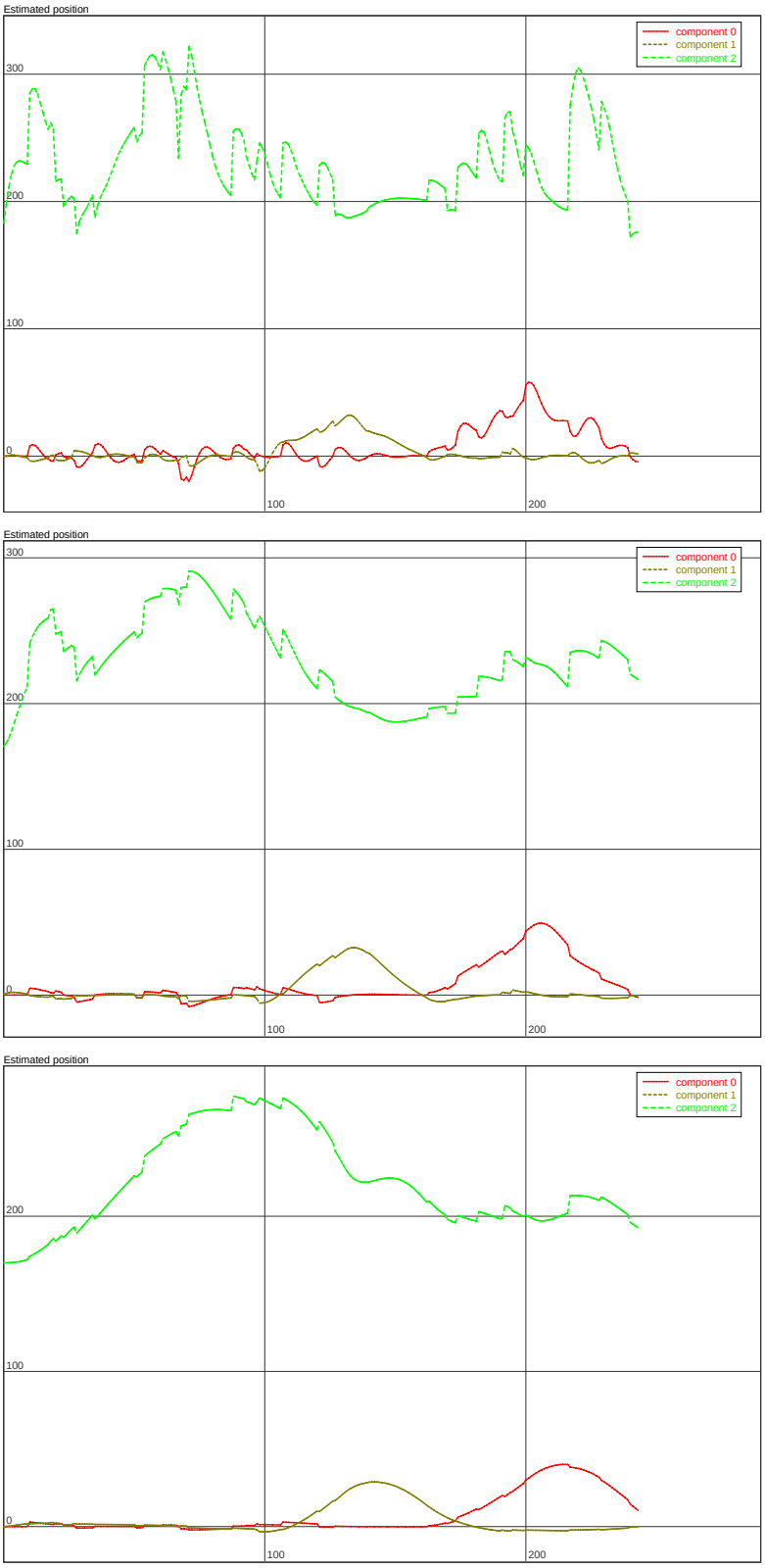


Bild C.16: Funktionsplots zum Experiment in Abschnitt 5.3 zur Rekonstruktion der Kamerabewegung mit Ausreißern unter Variation der Messwertrauschen-Kovarianz $\boldsymbol{R}$.

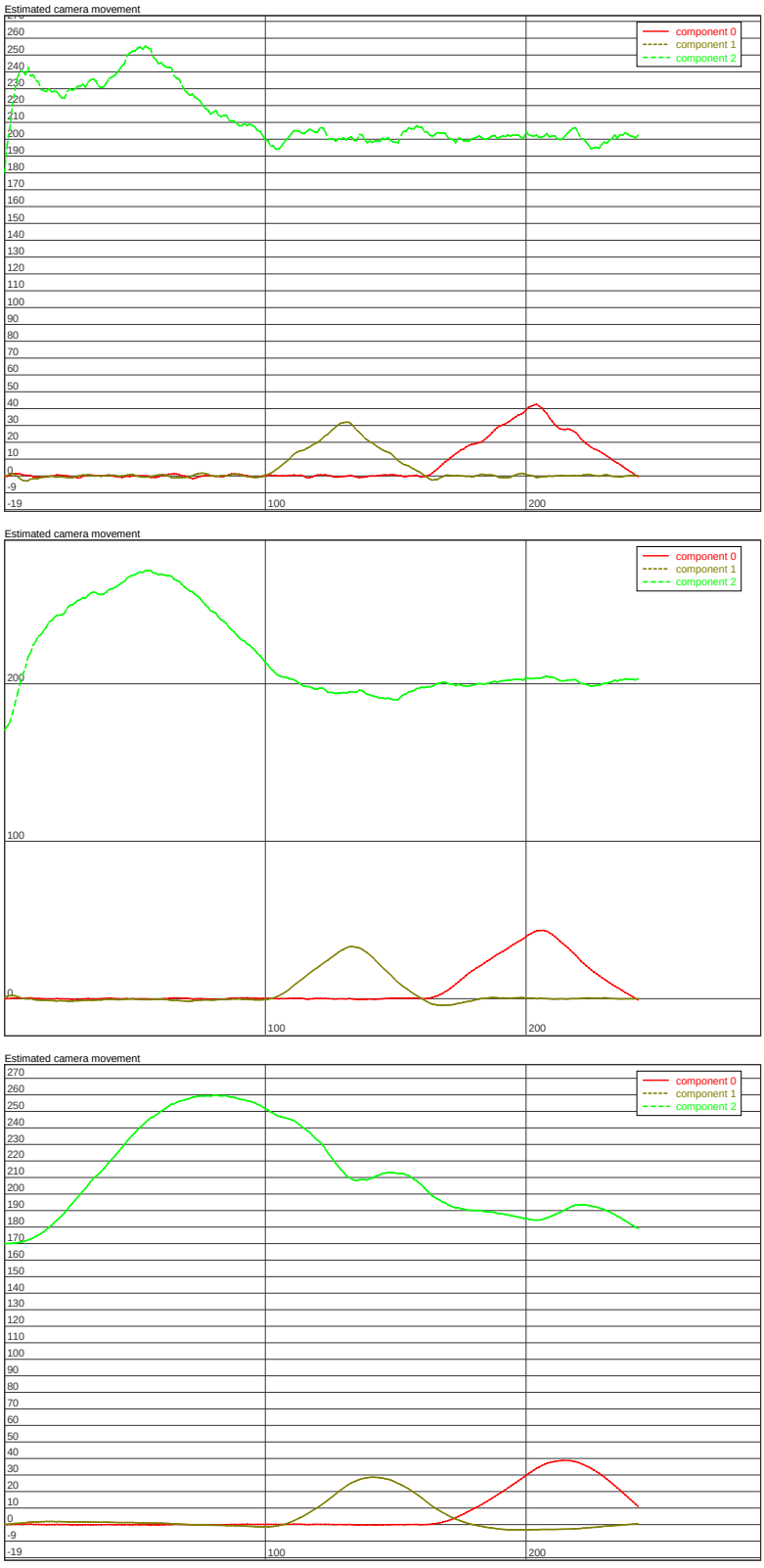


Bild C.17: Funktionsplots zum Experiment in Abschnitt 5.3 zur Rekonstruktion der Kamerabewegung mit starkem Rauschen in den Positionen der simulierten Punktmerkmale unter Variation der Messwertrauschen-Kovarianz $\boldsymbol{R}$.

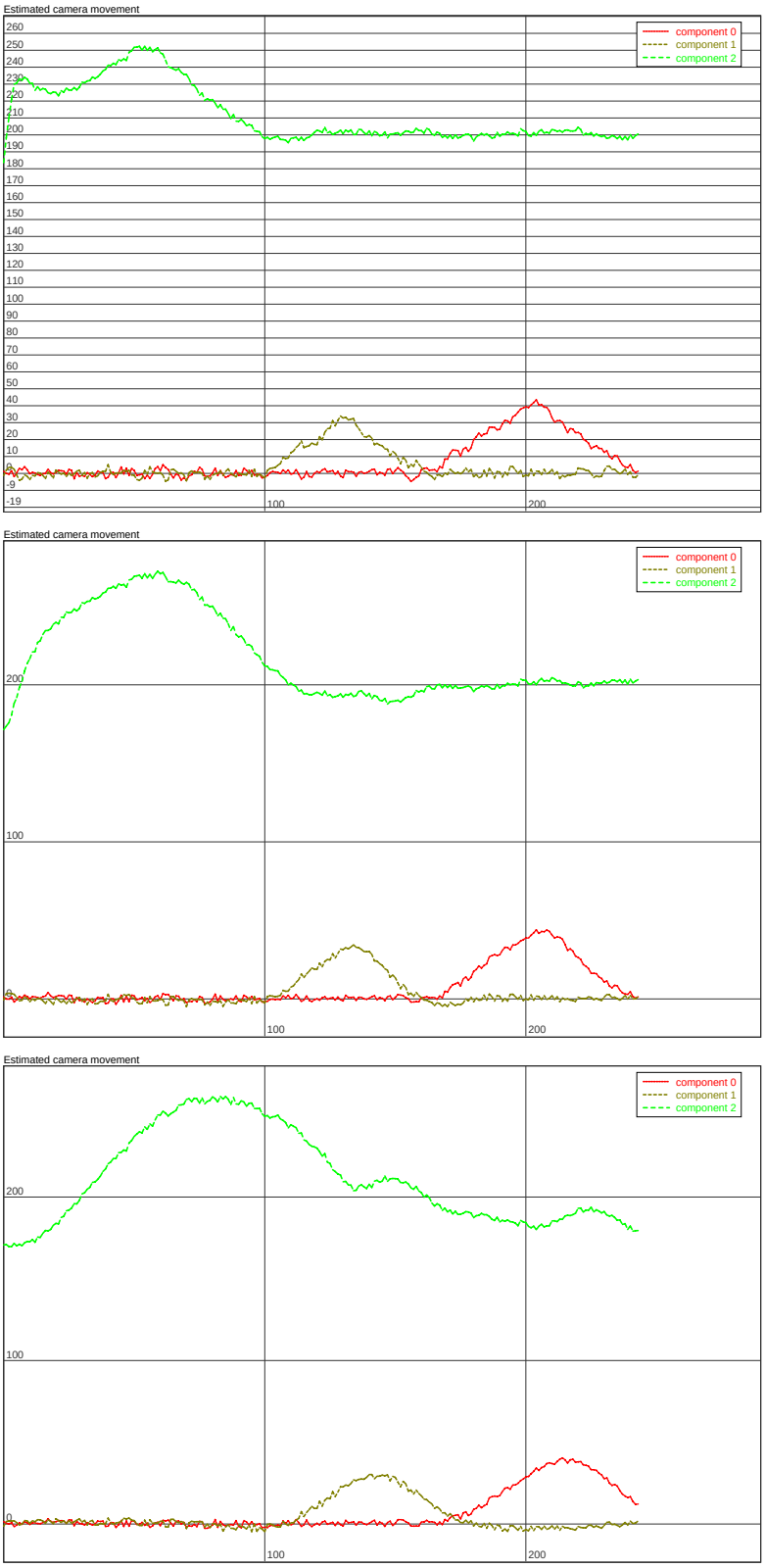


Bild C.18: Funktionsplots zum Experiment in Abschnitt 5.3 zur Rekonstruktion der Kamerabewegung mit starkem Rauschen in der Orientierungsmatrix des simulierten Inertialsensors unter Variation der Messwertrauschen-Kovarianz $\boldsymbol{R}$.

Anhang D

Inhalt der beiliegenden CD-ROM

Verzeichnis	**Inhalt**
`Ausarbeitung/`	
`Dickscheid2006.pdf`	Diese Diplomarbeit in elektronischer Form
`Arbeit/`	LaTeX-Quellen der schriftlichen Ausarbeitung
`Vortrag/`	LaTeX-Quellen der Folien zum Oberseminarvortrag
`Programmcode/`	Quelldateien der entwickelten C++-Klassenbibliothek
`include/`	Deklarationen der implementierten Klassen
`src/`	Definitionen der implementierten Klassen
`examples/`	Quelldateien der Beispielprogramme aus Kapitel 4
`etc/`	Muster-Konfigurationsdateien zu den Beispielprogrammen
`projectfiles/`	Projektdateien für *Microsoft Visual Studio 2005* und *Apple Xcode 1.5*
`tools/`	Lokale Version des freien Build-Systems *scons* (`http://www.scons.org`)
`DoxygenDoku/`	API-Dokumentation im HTML-Format
`Literatur/`	Elektronische Fassungen frei verfügbarer Literatur
`Sonstiges/`	
`Testdaten/`	Diverses Bildmaterial der Experimente und Beispiele

Tabelle D.1: Inhalt der beiliegenden CD-ROM.

Literaturverzeichnis

[AMT04] ALENYÀ, Guillem ; MARTÍNEZ, Elisa ; TORRAS, Carme: Fusing visual and inertial sensing to recover robot egomotion. In: *Journal of Robotic Systems* 21 (2004), S. 23–32

[BPS05] BLESER, Gabi ; PASTAMOV, Yulian ; STRICKER, Didier: Real-time 3D Camera Tracking for Industrial Augmented Reality Applications. In: *13-th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG)*, 2005

[BWBS06] BLESER, Gabi ; WOHLLEBER, Cedric ; BECKER, Mario ; STRICKER, Didier: Fast and Stable Tracking for AR fusing Video and Inertial Sensor Data. In: *14th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG'2006)*, 2006

[DA06] DOMKE, Justin ; ALOIMONOS, Yiannis: Integration of Visual and Inertial Information for Egomotion: a Stochastic Approach. In: *I*EEE International Conference on Robotics and Automation (ICRA)[pro06]

[Dav03] DAVISON, Andrew: Real-time simultaneous localisation and mapping with a single camera. In: *9th international Conference on Computer Vision (ICCV)*. Nice, France, 2003

[Dic05] DICKSCHEID, Timo: *Automatische Referenzpunktverfeinerung in Panoramabildern mittels SIFT-Operator*, Universität Koblenz-Landau, Studienarbeit, 9 2005

[DMM03] DAVISON, Andrew ; MAYOL, Walterio ; MURRAY, David: Real-Time Localisation and Mapping with Wearable Active Vision. In: *Proc. IEEE International Symposium on Mixed and Augmented Reality*, IEEE Computer Society Press, 10 2003

[FB81] FISCHLER, Martin A. ; BOLLES, Robert C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. In: *Communications of the ACM* 24 (1981), Nr. 6, S. 381–395

[GGB06] GRABNER, Michael ; GRABNER, Helmut ; BISCHOF, H.: Fast approximated SIFT. In: *7th Asian Conference of Computer Vision*, 2006

[Har92] HARRIS, C.: *Tracking with Rigid Objects*. MIT Press, 1992

[Har97] HARTLEY, Richard I.: In Defense of the Eight-Point Algorithm. In: *Pattern Analysis and Machine Intelligence* 19 (1997), 6, Nr. 6, S. 580–593

[HS88] HARRIS, C. ; STEPHENS, M.: A combined corner and edge detector. In: *Fourth Alvey Vision Conference*. Manchester, UK, 1988, S. 147–151

[HS03] HARTLEY, Richard I. ; SCHAFFALITZKY, Frederik: *PowerFactorization: 3D reconstruction with missing or uncertain data*. 2003

[HZ03] HARTLEY, Richard I. ; ZISSERMAN, Andrew: *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003

[IB98] ISARD, Michael ; BLAKE, Andrew: Condensation - conditional density propagation for visual tracking. In: *International Journal for Computer Vision* 29 (1998), Nr. 1, S. 5–28

[JFS03] JIN, H. ; FAVARO, P. ; SOATTO, Stefano: *A semi-direct approach to structure from motion*. 2003

[JU97] JULIER, Simon ; UHLMANN, J.: A new extension of the Kalman filter to nonlinear systems. In: *Int. Symp. Aerospace/Defense Sensing, Simul. and Controls, Orlando, FL*, 1997

[KH05] KIM, Jae-Hak ; HARTLEY, Richard I.: Translation Estimation from Omnidirectional Images. In: *DICTA*, IEEE Computer Society, 2005, S. 22

[KL81] KANADE, Takeo ; LUCAS, Bruce: An Iterative Image Registration Technique with an Application to Stereo Vision. In: *Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI '81)*, 1981

[Koe84] KOENDERINK, Jan J.: The structure of images. In: *Biological Cybernetics* 50 (1984), S. 363–370

[LF01] LUONG, Quang-Tuan ; FAUGERAS, Olivier D.: *The geometry of multiple images: the laws that govern the formation of multiple images of a scene and some of their applications*. Massachusetts : Massachusetts Institute of Technology Press, 2001

[Low04] LOWE, David G.: Distinctive Image Features from Scale-Invariant Keypoints. In: *International Journal of Computer Vision* 60 (2004), Nr. 2, S. 91–110

[May79] MAYBECK, Peter S.: *Mathematics in Science and Engineering*. Bd. 141: *Stochastic models, estimation, and control*. 525 B St., San Diego, CA 92101-4412, USA : Academic Press Professional, Inc., 1979

[MCD06] MONTIEL, J.M.M. ; CIVERA, Javiel ; DAVISON, Andrew: Unified Inverse Depth Parametrization for Monocular SLAM. In: *Online Proceedings Robotics, Science and Systems (RSS 2006)*, 2006

[MD06] MONTIEL, J.M.M. ; DAVISON, Andrew: A visual compass based on SLAM. In: *I*EEE International Conference on Robotics and Automation (ICRA)[pro06]

[Mor81] MORAVEC, Hans P.: *Robot Rover Visual Navigation*. Umi Research Press, 1981

[Nis03] NISTÉR, David: Preemptive RANSAC for Live Structure and Motion Estimation. In: *ICCV*, 2003, S. 199–206

[NNB04] NISTÉR, David ; NARODITSKY, Oleg ; BERGEN, James R.: Visual Odometry. In: *CVPR (1)*, 2004, S. 652–659

[PC05] PUPILLI, Mark ; CALWAY, Andrew: Real-Time Camera Tracking Using a Particle Filter. In: *Proceedings of the British Machine Vision Conference*, BMVA Press, 9 2005, S. 519–528

[pro06] *IEEE International Conference on Robotics and Automation (ICRA)*. 2006

[SFZ00] SIMON, Gilles ; FITZGIBBON, Andrew W. ; ZISSERMAN, Andrew: Marker-less tracking using planar structures in the scene. In: *Proceedings of IEEE and ACM International Symposium on Augmented Reality*, 2000, S. 120–128

[SP05] SCHWEIGHOFER, G. ; PINZ, A.: Robust Pose Estimation from a Planar Target / EMT, TU Graz, Austria. 2005 (TR-EMT-2005-01). – Forschungsbericht. – Submitted to IEEE PAMI

[SS03] STRELOW, D. ; SINGH, S.: Online motion estimation from image and inertial measurements. In: *Workshop on Integration of Vision and Inertial Sensors* (2003)

[ST94] SHI, Jianbo ; TOMASI, Carlo: Good Features to Track. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*. Seattle, 6 1994, S. 593–600

[ST96] STURM, Peter F. ; TRIGGS, Bill: A Factorization Based Algorithm for Multi-Image Projective Structure and Motion. In: *Computer Vision - ECCV'96, 4th European Conference on Computer Vision* Bd. 2. Cambridge, UK : Springer, 4 1996, S. II: 709–720

[TBF05] THRUN, Sebastian ; BURGARD, Wolfram ; FOX, Dieter: *Probabilistic Robotics*. MIT Press, 2005

[TK92] TOMASI, Carlo ; KANADE, Takeo: Shape and Motion from Image Streams: a Factorization Method, Full Report on the Orthographic Case / Carnegie Mellon University. 1992 (CMU-CS-92-104). – Forschungsbericht

[TV98] TRUCCO, E. ; VERRI, A.: *Introductory Techniques for 3–D Computer Vision*. New York : Prentice Hall, 1998

[VJ01] VIOLA, Paul ; JONES, Michael: Rapid object detection using a boosted cascade of simple features. In: *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* Bd. 1, IEEE Computer Society, 2001, S. 511–518

[VL04] VACCHETTI, Luca ; LEPETIT, Vincent: Stable Real-Time 3D Tracking Using Online and Offline Information. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26 (2004), Nr. 10, S. 1385–1391. – Member-Pascal Fua

[WB04] WELCH, Greg ; BISHOP, Gary: *An Introduction to the Kalman Filter*. 4 2004

[YN01] YOU, S. ; NEUMANN, U.: Fusion of Vision and Gyro Tracking for Robust Augmented Reality Registration. In: *Virtual Reality Conference (VR)*, 2001