UNIVERSITÄT KOBLENZ · LANDAU
Fachbereich 4: Informatik

AGAS
Arbeitsgruppe Aktives Sehen

Modellbasiertes Tracking mittels Linien- und Punktkorrelationen

Diplomarbeit

zur Erlangung des Grades eines/r

Diplom-Informatikers / Diplom-Informatikerin
im Studiengang Computervisualistik

vorgelegt von

Dag Ewering

Betreuer:	Prof. Dr.-Ing. Dietrich Paulus, Institut für Computervisualistik, Fachbereich Informatik
Erstgutachter:	Prof. Dr.-Ing. Dietrich Paulus, Institut für Computervisualistik, Fachbereich Informatik
Zweitgutachter:	Dipl.-Inf. Tobias Feldmann, Institut für Computervisualistik, Fachbereich Informatik

Koblenz, im September 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 ..... Unterschrift

Inhaltsverzeichnis

Kapitel 1

Einleitung

Für viele Anwendungen ist es notwendig, 3-D-Objekte in einer Videosequenz zu verfolgen. Soll z. B. ein Roboter Gegenstände manipulieren, so müssen diese verfolgt werden, um die Arme des Roboters korrekt auszurichten. Eine andere Anwendung, für die das Verfolgen von Objekten zentral ist, ist die Navigation auf der Basis von Kopfpositionen. Beim sogenannten *Head-Tracking* muss kontinuierlich die genaue Orientierung eines Gesichtes bestimmt werden. In anderen Anwendungen werden Objekte verfolgt, um eine *Erweiterte Realität*[1] zu realisieren. Die Realität wird dazu so *erweitert*, dass Videodaten, die eine reale Szene abbilden, von virtuellen Informationen überlagert werden. Häufig steht dabei ein bestimmtes Objekt im Mittelpunkt, von dem im Allgemeinen ein Modell vorliegt. Um z. B. eingeblendete Zusatzinformationen an einem Objekt auszurichten, muss dieses verfolgt werden. Im Folgenden wird die deutsche Bezeichnung *Verfolgung* gleichbedeutend mit dem englischen Begriff *Tracking* verwendet.

Unter der Verfolgung eines Objektes in einer Videosequenz versteht man, dass kontinuierlich dessen relativer Ort zur Szene bestimmt wird. Für das 3-D-Tracking heißt dies, dass man aus Bilddaten bestimmt, an welcher Position sich die Kamera befindet, die die betrachteten Bilddaten aufgenommen hat. Neben der Position wird meist auch die Orientierung dieser Kamera bestimmt. Die beiden genannten Größen (Rotation und Translation der Kamera im Weltkoordinatensystem) bezeichnet man als *äußere Orientierung* oder *Po-*

[1] engl. augmented reality

se.

Im Rahmen dieser Diplomarbeit wird ein Ansatz zum Verfolgen von Objekten vorgestellt, der die Pose auf der Basis eines 3-D-Modells bestimmt. Zu dem Zweck werden Referenzdaten benötigt, die einmalig von dem zu verfolgenden Objekt erstellt werden müssen. Zum Erstellen der Referenzdaten wird ein Referenzdaten-Modul genutzt, dessen Funktionsweise und Architektur in dieser Diplomarbeit vorgestellt wird.

Durch technische Hilfsmittel ist in vielen Fällen eine Groblokalisierung gegeben der Kamera. Diese ist für die meisten Anwendungen jedoch zu ungenau. Sie kann als Ausgangspunkt verwendet werden, um mit den Möglichkeiten der Bildverarbeitung eine exakte Pose zu bestimmen. Es wird im Rahmen dieser Diplomarbeit davon ausgegangen, dass beim Verfolgen eines Objektes zu jeder Zeit eine ungefähre Pose bekannt ist.

Die Ansätze zur Realisierung eines Tracking-Systems lassen sich grob in markerbasierte und markerlose Verfahren unterteilen. Der in dieser Diplomarbeit realisierte Ansatz basiert nicht auf der Verwendung von Markern. Er ist modellbasiert und realisiert das Tracking auf der Basis von extrahierten *Merkmalen*[2]. Als Merkmale werden Kombinationen von lokalen Punktdeskriptoren und Linien verwendet, womit der verwendete Ansatz das Verfolgen von Objekten ähnlich wie der Ansatz von Vacchetti [VLF04] realisiert.

Im Verlauf der Diplomarbeit wird insbesondere geprüft, inwieweit es sinnvoll ist, ein punktbasiertes Tracking um ein Tracking von Linien zu erweitern. Es werden verschiedene Möglichkeiten vorgestellt und untersucht, wie eine solche Erweiterung realisiert werden kann.

Die Diplomarbeit besteht neben diesem einleitenden Kapitel aus vier weiteren Kapiteln. Kapitel 2 beschreibt den Stand der Technik im Bereich der Objektverfolgung. Es werden zentrale Konzepte und Methoden vorgestellt und verschiedene Ansätze zum Verfolgen von Objekten betrachtet. Eine detaillierte Vorstellung des eigenen Ansatzes und der Architektur der Implementierung erfolgt in Kapitel 3. Zur Evaluation des vorgestellten Ansatzes wird ein im Rahmen dieser Diplomarbeit implementiertes Programm genutzt. Die Evaluation der Objektverfolgung wird in Kapitel 4 vorgestellt. Nach der Bewertung der Testergebnisse erfolgt eine abschließende Zusammenfassung in Kapitel 5.

[2] engl. features

Kapitel 2

Stand der Wissenschaft

2.1 Überblick Tracking-Ansätze

Zur Bestimmung der äußeren Orientierung werden heutzutage in vielen Fällen Marker genutzt, die vor der Aufnahme der Videoinformationen in der Szene platziert werden. Mit dem Wissen über die Position der Marker im Raum und im Bild kann die Pose der Kamera bestimmt werden. Die Detektion der Marker im Bild leistet z. B. das ARToolKit [KB99]. Die Platzierung von Markern ist jedoch nicht in allen Umgebungen möglich, weswegen auch in dieser Diplomarbeit Objekte markerlos verfolgt werden sollen. Möchte man auf Marker verzichten, so müssen zur Bestimmung der Pose statt Markern natürliche Merkmale in der Szene verfolgt werden.

Beim markerlosen Tracking unterscheidet man zwei unterschiedliche Herangehensweisen: bildbasierte Ansätze und modellbasierte Ansätze. Beim globalen bildbasierten Ansatz wird das zu verfolgende Objekt durch eine große Menge von Bildern, Histogrammen oder anderen Beschreibungen gespeichert. Zur Bestimmung der Pose werden diese Repräsentationen über verschiedene Ähnlichkeitsmaße mit den Video-Bilddaten verglichen. Lepetit sieht in [LPF04] einen Vorteil des bildbasierten Ansatzes darin, dass er performant implementiert werden kann. Die Bilddaten, mit denen das Objekt verglichen wird, würden jedoch viel Speicherplatz benötigen. Weiterhin funktioniere das Verfahren bei Verdeckungen des zu verfolgenden Objektes schlecht.

Beim modellbasierte Ansatz wird ein 3-D-Modell des zu verfolgenden Objektes genutzt, um die Kamerapose zu bestimmen. Der modellbasierte Ansatz wird in [LPF04] auch als *lokaler Ansatz* bezeichnet, da statt ganzen Bildern nur einzelne lokale Merkmale verglichen werden. Mit Hilfe des 3-D-Modells können diesen im Bild extrahierten Merkmalen 3-D-Informationen zugeordnet werden. Auf der Basis dieser Korrespondenzen kann die Kamerapose geschätzt werden.

Lepetit gibt an, dass Tracking-Verfahren, die einen modellbasierten Ansatz realisieren, im Allgemeinen nicht die Performanz aufweisen, die man mit einem bildbasierten Ansatz erreichen kann. Laut Lepetit [LF05] existieren jedoch mittlerweile Implementationen, die Echtzeit-Ansprüchen genügen, weswegen dieser Ansatz auch in dieser Diplomarbeit verfolgt wird. Einen Vorteil gegenüber dem bildbasierten Ansatz sieht er in der Robustheit gegenüber Verdeckungen. Weiterhin sei mit dem modellbasierten Ansatz eine potentiell genauere Bestimmung der Kamerapose möglich.

2.2 Modellbasiertes Tracking

Wie bereits erwähnt basiert das modellbasierte Tracking auf einem 3-D-Modell des zu verfolgenden Objektes. Dieses Modell muss im Vorhinein erzeugt werden. Die Erzeugung der Modelle kann durch manuelles Ausmessen oder automatische Erzeugung mit Methoden des Maschinellen Sehens[1] geleistet werden. CAD-Tools können diesen Vorgang unterstützen.

Die Vorgehensweise beim modellbasierten Tracking geschieht meist in drei Schritten. Im ersten Schritt werden Merkmale im Bild extrahiert. Diesen Merkmalen werden im zweiten Schritt geometrische Primitive im Modell zugeordnet. Im letzten Schritt wird auf der Basis dieser Korrespondenzen die Kamerapose bestimmt bzw. geschätzt.

Während sich viele Tracking-Algorithmen bzgl. der erwähnten geometrischen Primitive auf Punkte beschränken, existieren auch Ansätze, die auf der Verfolgung von Linien und anderen geometrischen Primitiven basieren [VLF04].

Das folgende Kapitel zeigt auf, mit welchen Algorithmen die Kamerapose geschätzt wer-

[1] engl. computer vision

den kann, wenn eine Menge von Punktkorrespondenzen zwischen Modell- und Bildpunkten bekannt ist. Beim Tracking liefert die Kamera jedoch Merkmale, für die keine Punktkorrespondenzen bekannt sind. Wie man mit Hilfe von Referenzbildern auch für diese Bilder Punktkorrespondenzen erhält, wird in Abschnitt 2.4 erläutert. Nach der Betrachtung punktbasierter Ansätze zur Objektverfolgung wird der Stand der Wissenschaft des linienbasierten Trackings vorgestellt.

2.3 Poseberechnung auf Basis von Punktkorrespondenzen

2.3.1 Lochkamera-Modell

Die im weiteren Verlauf vorgestellten Tracking-Ansätze gehen stets davon aus, dass sich die verwendete Kamera als Lochkamera modellieren lässt. Bei der Abbildung von Welt- in Pixelkoordinaten wird von einer perspektivischen Projektion ausgegangen. Die Abbildung kann mathematisch als Multiplikation mit einer Projektionsmatrix realisiert werden. Diese Projektionsmatrix beschreibt das Produkt der 3×4-Matrix $[\boldsymbol{R}|\boldsymbol{t}]$ und der 3×3-Kalibriermatrix $\boldsymbol{K}$:

$$\boldsymbol{P} = \boldsymbol{K}[\boldsymbol{R}|\boldsymbol{t}] \tag{2.1}$$

Die Matrix $[\boldsymbol{R}|\boldsymbol{t}]$ realisiert die Abbildung vom Welt- in das Bildkoordinatensystem. Sie beschreibt die Rotation $\boldsymbol{R}$ und die Translation $\boldsymbol{t}$ der Kamera. Diese werden als extrinsische Parameter bezeichnet und beschreiben zusammen die Pose der Kamera im Weltkoordinatensystem. Die Kalibriermatrix $\boldsymbol{K}$ beschreibt die intrinsischen Parameter der Kamera: die Brennweite, Pixelskalierung in horizontaler und vertikaler Richtung, die Pixelkoordinaten der Bildmitte und den Winkel zwischen den Achsen.

Im Folgenden werden Methoden vorgestellt, um die Projektionsmatrix $\boldsymbol{P}$ bzw. die extrinsischen Kameraparameter $\boldsymbol{R}$ und $\boldsymbol{t}$ für ein gegebenes Bild zu schätzen.

2.3.2 Lineare Kalibrierung

Eine analytische Methode zur Berechnung der Projektionsmatrix ist die *Direct Linear Transformation* (Abkürzung: DLT) [HZ03]. Mit ihr lässt sich die Projektionsmatrix analytisch auf der Basis von sechs Punktkorrespondenzen bestimmen.

Die lineare Abbildung eines Punktes $\boldsymbol{M}$ in Weltkoordinaten durch die Projektionsmatrix $\boldsymbol{P}$ in einen Punkt in Pixelkoordinaten $\boldsymbol{m}$ lässt sich allgemein wie folgt formulieren:

$$\boldsymbol{m} = \begin{pmatrix} x \\ y \\ w \end{pmatrix} = \boldsymbol{P} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \boldsymbol{PM} \tag{2.2}$$

Die Pixelkoordinaten sind als homogene Koordinaten angegeben, wobei für einen beliebigen i-ten Punkt gilt, dass $x_i = \frac{x}{w}$ bzw. $y_i = \frac{y}{w}$. Man erhält durch einfache Umformungen die folgenden Gleichungen:

$$x_i = \frac{p_{11}X + p_{12}Y + p_{13}Z + p_{14}}{p_{31}X + p_{32}Y + p_{33}Z + p_{34}} \tag{2.3}$$

$$x_i = \frac{p_{11}X + p_{12}Y + p_{13}Z + p_{14}}{p_{31}X + p_{32}Y + p_{33}Z + p_{34}} \tag{2.4}$$

Für die Elemente der Projektionsmatrix $\boldsymbol{P}$ ergibt sich damit pro Gleichung eine Bedingung. Da jede Punktkorrespondenz zwischen einer Welt- und einer Pixelkoordinate zwei Bedingungen liefert, lässt sich die Projektionsmatrix ab einer Anzahl von sechs solcher Korrespondenzen linear berechnen. Es ergibt sich somit ein Gleichungssystem der Form $\boldsymbol{Ap} = 0$:

$$\begin{pmatrix} -X & -Y & -Z & -1 & 0 & 0 & 0 & 0 & x_iX & x_iY & x_iZ & x_i \\ 0 & 0 & 0 & 0 & -X & -Y & -Z & -1 & y_iX & y_iY & y_iZ & y_i \end{pmatrix} \begin{pmatrix} p_{11} \\ \vdots \\ p_{34} \end{pmatrix} = 0 \tag{2.5}$$

Die Bestimmung der Elemente der Projektionsmatrix entspricht nun der Bestimmung des Nullraums der Messmatrix. Da $\boldsymbol{A}$ den Rang 11 hat, hat das Gleichungssystem keine eindeutige Lösung. Mit der DLT lässt sich die Projektionsmatrix demnach nur bis auf einen Skalierungsfaktor bestimmen.

Aufgrund der Ungenauigkeit der 2-D-Koordinaten verwendet man im Allgemeinen mehr als sechs Punktkorrespondenzen. Das zu lösende Gleichungssystem ist überbestimmt. Eine Lösung mit kleinstem quadratischem Fehler erhält man, wenn man die Inversion der Matrix mit Hilfe der Singulärwertzerlegung $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T$ löst. Der Nullraum der Matrix $\boldsymbol{A}$ ist der Spaltenvektor von $\boldsymbol{V}$, der zum kleinsten Singulärwert in $\boldsymbol{D}$ gehört.

2.3.3 Normalisierung

Um die Qualität der linearen Kalibrierung zu erhöhen, ist es sinnvoll, eine Normalisierung der Eingabedaten durchzuführen [HZ03]. Dadurch erreicht man, dass die berechnete Projektionsmatrix unabhängig von der Skalierung und der Translation der Punktkorrespondenzen ist. Weiterhin erhöht die Normalisierung die numerische Stabilität, da die Messmatrix $\boldsymbol{A}$ besser konditioniert ist.

Normalisierte Pixelkoordinaten erhält man durch eine Translation und eine Skalierung der ursprünglichen Pixelkoordinaten. Durch die Translation wird der Schwerpunkt der Punkte in den Ursprung des Pixelkoordinatensystems verlegt. Die 2-D-Punkte werden weiterhin so skaliert, dass ihr durchschnittlicher Abstand zum Ursprung $\sqrt{2}$ beträgt. Zur Normalisierung der Modellpunkte verlegt man auch hier den Schwerpunkt in den Ursprung des Koordinatensystems. Die Punkte im Weltkoordinatensystem werden außerdem so skaliert, dass ihr durchschnittlicher Abstand zum Ursprung $\sqrt{3}$ beträgt. Die Translation und Skalierung kann jeweils als Matritzenmultiplikation implementiert werden. Sei $\boldsymbol{T_i}$ die Matrix, die die Transformation der Bildkoordinaten realisiert, und $\boldsymbol{T_w}$ die Matrix, die die Transformation der Weltkoordinaten realisiert. Dann müssen die Transformationen nach dem Schätzen der Projektionsmatrix $\boldsymbol{P'}$ wieder wie folgt rückgängig gemacht werden:

$$\boldsymbol{P} = \boldsymbol{T_i}^{-1}\boldsymbol{P'}\boldsymbol{T_w} \tag{2.6}$$

2.3.4 POSIT

Lepetit nutzt in [LPF04] den *Pose from Orthography and Scaling with ITerations*-Algorithmus [DD92] (POSIT) zur modellbasierten Schätzung der Kamerapose. Sind die intrinsischen Kameraparameter bekannt, so reichen bereits vier Punktkorrespondenzen, um die extrinsischen Kameraparameter zu schätzen. POSIT geht dabei in zwei Schritten vor: im ersten schätzt er, ausgehend von der vereinfachenden Annahme einer skaliert orthographischen Projektion, Translation und Rotation. Im zweiten Schritt wird diese berechnete Pose genutzt, um die angenommene orthographische Projektion zu optimieren. Beide Schritte werden bis zur Konvergenz der Parameter wiederholt.

2.3.5 Minimierung des Reprojektionsfehlers

Ein möglicher Ansatz zur Schätzung der Pose ist es, verschiedene Projektionsmatritzen danach zu bewerten, wie gut sie die gegebenen Modellpunkte auf die korrespondierenden Pixelkoordinaten abbilden. Die geschätzte Pose ergibt sich schließlich aus der Projektionsmatrix, die als beste bewertet wurde. Zur Bewertung werden alle 3-D-Modellpunkte mit der zu bewertenden Projektionsmatrix auf Bildpunkte abgebildet. Da der zu einem Modellpunkt korrespondierende Bildpunkt bekannt ist, kann der Abstand zwischen diesem und dem errechneten Pixelwert als Qualitätsmaß verwendet werden.

Die Kamerapose kann also durch eine Minimierung des Projektionsfehlers geschätzt werden. Man spricht an dieser Stelle auch vom *Reprojektionsfehler*, da man versucht, die reale Kameraprojektion nachzustellen. Sei $\boldsymbol{P}$ eine Projektionsmatrix und $\boldsymbol{M_i}$ der zu $\boldsymbol{m_i}$ korrespondierende 3-D-Punkt, dann ist der Reprojektionsfehler r der Projektion Φ wie folgt definiert:

$$r = \sum_{i=1}^{k} (\boldsymbol{m_i} - \Phi(\boldsymbol{P}, \boldsymbol{M_i}))^2 \tag{2.7}$$

Die Skizze in Abbildung 2.1 veranschaulicht die relevanten Größen. Die zu schätzende Projektionsmatrix entspricht schließlich der Matrix, die den Reprojektionsfehler minimiert:

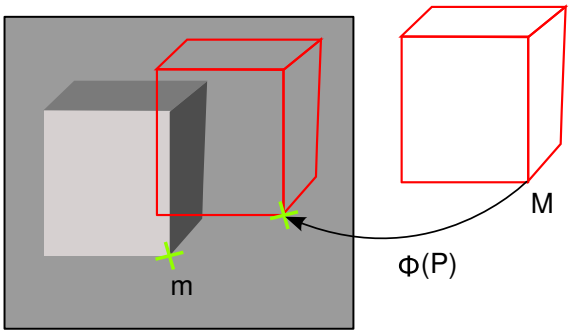

Bild 2.1: Verwendete Fehlerfunktion zur nicht-linearen Schätzung der Kamerapose

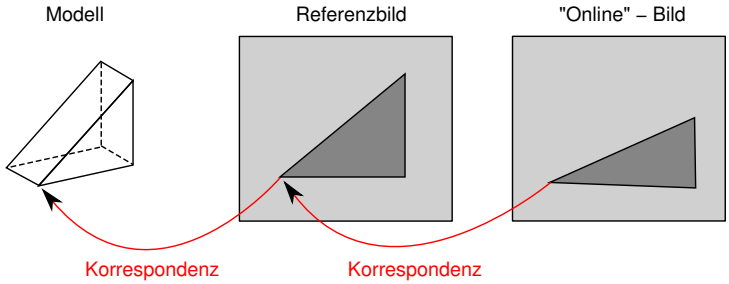

Bild 2.2: Bestimmung von Punktkorrespondenzen zwischen Online-Bild und Modell

$$\boldsymbol{P} = \arg\min_{\boldsymbol{P}} \sum_{i=1}^{k} (\boldsymbol{m_i} - \Phi(\boldsymbol{P}, \boldsymbol{M_i}))^2 \tag{2.8}$$

2.4 Poseschätzung als Minimierungsproblem

2.4.1 Referenzbilder

Bestimmt man für ein bestimmtes Kamerabild Korrespondenzen zwischen Modell- und Bildpunkten (z. B. durch Ausmessen), so können die extrinsischen Kameraparameter ver-

lässlich bestimmt werden. Verschiedene Methoden dazu wurden in Kapitel 2.3 vorgestellt. Dieses Vorgehen ist jedoch online nicht realisierbar[2].

Es ist jedoch möglich, Punktkorrespondenzen zwischen dem aktuell betrachteten Videobild und einem Referenzbild zu bestimmen. Sind für die Punkte des Referenzbildes die korrespondierenden Modellpunkte bekannt, so kann über diesen "Umweg" die Kamerapose für das aktuelle Videobild geschätzt werden. Eine graphische Beschreibung des Vorgehens zeigt Abbildung 2.2.

Die modellbasierte Poseberechnung durch Nutzung von *Referenzbildern*[3] beschreibt Lepetit in [LVTF03]. Vor dem eigentlichen Tracking müssen von dem zu verfolgenden Objekt aus verschiedenen Positionen Aufnahmen gemacht werden. Für diese Aufnahmen sollte dann eine große Anzahl von Punktkorrespondenzen bestimmt werden. Auf die Bestimmung dieser Korrespondenzen wird in Abschnitt 2.5 näher eingegangen.

Die Schätzung der Kamerapose auf der Basis dieser Punktkorrespondenzen realisieren sowohl Lepetit [LVTF03] als auch Wuest [WVS05] durch die Minimierung des Reprojektionsfehlers.

2.4.2 Optimierungsverfahren

Wird die Schätzung der Kamerapose durch Minimierung des Reprojektionsfehlers realisiert, so benötigt man ein Optimierungsverfahren. Dieses Optimierungsverfahren ermittelt die Parameter, für die eine Fehlerfunktion minimal wird. Es liefert beim modellbasierten Tracking also die Pose, die die Fehlerfunktion minimiert.

Es existieren eine Reihe von Optimierungsverfahren, wobei man zwischen lokalen und globalen Optimierverfahren unterscheidet. Mit Hilfe von globalen Optimierverfahren lässt sich das absolute Minimum einer Zielfunktion bestimmen. Da eine globale Optimierung für die Minimierung des Reprojektionsfehlers zu rechenintensiv wäre, ist es üblich, das Minimum durch ein lokales Optimierverfahren zu bestimmen. Lokale Optimierverfahren suchen bei gegebenen Anfangsparametern nach einem lokalen Minimum einer Zielfunk-

[2] Der Begriff "online" hat hier die Bedeutung "beim Tracking" - "offline" steht für "vor dem Tracking". Online durchgeführte Operationen sind im Gegensatz zu offline durchgeführten Operationen zeitkritisch.

[3] engl. keyframe

tion. Als initiale Parameter für die Optimierung verwendet man meist die Parameter des vorherigen Kamerabildes. Dies macht Sinn, da sich die extrinsischen Kameraparameter im Normalfall von Bild zu Bild nur geringfügig ändern. Die Abhängigkeit der geschätzten Pose vom Startparametervektor kann jedoch zu falschen Ergebnissen führen. Je weiter die initiale Pose von der aktuellen Kamerapose entfernt ist, desto wahrscheinlich ist es, dass das lokale und das globales Minimum nicht identisch sind.

Zum Verfolgen von Objekten können verschiedene Methoden der nicht-linearen Optimierung verwendet werden. Wuest nutzt in [WVS05] zur Minimierung des Reprojektionsfehlers den Levenberg-Marquardt-Algorithmus, eine Variante des Gauß-Newton-Verfahrens. Malciu verwendet in [MP00] den Downhill-Simplex-Algorithmus [MN65] zur modellbasierten Schätzung der Kopfposition in Videosequenzen. Der Downhill-Simplex-Algorithmus kommt im Gegensatz zum Levenberg-Marquardt-Algorithmus ohne Ableitungen der Fehlerfunktion aus.

2.4.3 Robustheit gegenüber Ausreißern

Bei der Bestimmung von Punktkorrespondenzen mit dem Referenzbild erhält man in vielen Fällen auch falsche Punktkorrespondenzen. Basiert die Poseschätzung auf einer großen Anzahl fehlerhafter Korrespondenzen, so wird auch die berechnete Pose fehlerhaft sein. Es gibt verschiedene Ansätze, um die Robustheit der Objektverfolgung gegenüber Ausreißern zu erhöhen. Zwei sehr verbreitete Ansätze sind der *Random Sample Consensus Algorithmus* (RANSAC) und die Nutzung von M-Schätzern.

RANSAC

RANSAC [FB81] ist eine Standardmethode im Bereich *Struktur aus Bewegung*, um die Robustheit gegenüber Ausreißern zu erhöhen. Lepetit verwendet z. B. in [LPF04] eine RANSAC-basierte Methode, um bei der modellbasierten Objektverfolgung fehlerhafte Punktkorrespondenzen zu eliminieren. Die Vorgehensweise ist dabei die folgende:

1. Wähle zufällig n Punktkorrespondenzen.

2. Schätze die Pose auf der Basis dieser Korrespondenzen.
3. Bewerte nun für alle Punktkorrespondenzen, ob bei der geschätzten Pose, der 3-D-Punkt auf den korrespondierenden 2-D-Punkt abgebildet wird.
4. Ist die Anzahl solcher Punkte ausreichend groß, so akzeptiere die Schätzung und beende den Algorithmus.
5. Ansonsten: wiederhole die Schritte 1 bis 4 m-mal.
6. RANSAC schlägt fehl.

Wird die Schätzung akzeptiert, so werden die Punktkorrespondenzen, deren Modellpunkt durch die geschätzte Projektionsmatrix nicht auf den korrespondierenden 2-D-Punkt abgebildet wird, zur Poseberechnung nicht weiter betrachtet.

M-Schätzer

Die Berechnung der Kamerapose kann nicht-linear durch eine Minimierung des Reprojektionsfehlers realisiert werden. Der bisher vorgestellte Ansatz berechnet den Reprojektionsfehler als Summe von quadrierten Punktabständen. Die Fehlerfunktion hat in dem Fall die allgemeine Form:

$$r = \sum_{i=1}^{k} r_i^2 \tag{2.9}$$

Da der Reprojektionsfehler für jeden Punkt im Quadrat in die Summe eingeht, haben Ausreißer einen großen Einfluss auf den berechneten Fehler. Um den Einfluss von Ausreißern zu verringern, ist es häufig sinnvoll, einen M-Schätzer zu verwenden. M-Schätzer sind *Maximum-Likelihood-artige* Schätzer, die jedoch robuster gegen Ausreißer sind als Maximum-Likelihood-Schätzer. Die Idee dabei ist es, die Reprojektionsfehler einzelner Punkte nicht zu quadrieren, sondern sie mit einer Funktion ρ zu gewichten.

$$r_t = \sum_{i=1}^{k} \rho(r_i) \tag{2.10}$$

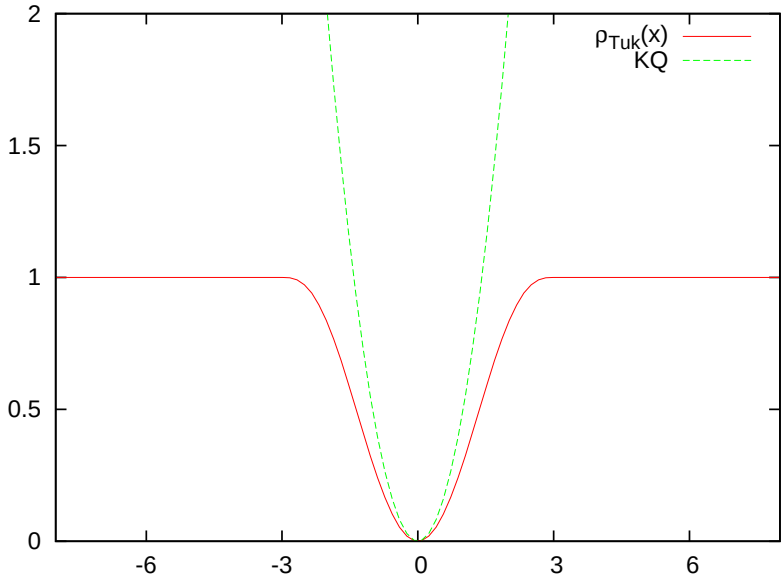

Bild 2.3: Vergleich Tukey-Schätzers – KQ-Schätzer (Methode der kleinsten Quadrate).

Bleser [Ble04] verwendet im Tracking-Kontext erfolgreich den Tukey-Schätzer:

$$\rho_{Tuk}(x) = \begin{cases} \frac{c^2}{6}\left[1 - \left(1 - \left(\frac{x}{c}\right)^2\right)^3\right] & \text{für } |x| \leq c \\ \frac{c^2}{6} & \text{sonst} \end{cases} \tag{2.11}$$

Während kleine Fehler quadratisch in die Summe eingehen, steigt die Funktion ab einem zu definierenden Punkt nicht mehr. Mit Hilfe des Parameters c kann definiert werden, ab wann ein Fehler als potentieller Ausreißer angesehen wird. Er sollte sinnvollerweise in Abhängigkeit von der erwarteten Standardabweichung gewählt werden. Abbildung 2.3 zeigt den Verlauf der Tukey-Funktion. Lepetit gibt in [LF05] an, dass auch der Huber-Schätzer in vielen Anwendungen genutzt wird:

$$\rho_{Hub}(x) = \begin{cases} x^2/2 & \text{für } |x| \leq c \\ c(|x| - c/2) & \text{sonst} \end{cases} \tag{2.12}$$

2.4.4 Jitter und Drift

Bilder, die man nacheinander aus Videodaten erhält, unterscheiden sich meist nur wenig. Diese Tatsache lässt sich ausnutzen, indem man Korrespondenzen zwischen aufeinander folgenden Bildern bestimmt und die Suche nach korrespondierenden Punkten auf eine kleine Region beschränkt. Ein Tracking, das sich ausschließlich auf Korrespondenzen zum vorherigen Bild stützt, ist deshalb im Allgemeinen schneller als Verfahren, die auf einem Modell und Referenzbildern basieren. Bei dem geschilderten Vorgehen ist es jedoch möglich, dass die geschätzter Pose von der realen Pose “wegdriftet”. Da stets auf den Korrespondenzen des vorherigen Bildes aufgebaut wird, werden auch die fehlerhaften Korrespondenzen übernommen. Lepetit beschreibt in [LVTF03], dass sich dieser Fehler schnell aufsummieren kann und die Qualität des Trackings stark abnimmt. Dieses Aufsummieren des Fehlers bezeichnet man als *Drift*.

Ein Vorteil dieser Vorgehensweise ist es, dass die berechnete Pose von Bild zu Bild nur kleine Variationen aufweist. Dies führt im Bereich der Augmented Reality zu einem flüssigen visuellen Eindruck bei der Positionierung der erweiterten Objekte. Betrachtet man ausschließlich Korrespondenzen zwischen Referenzbild und Online-Bild kann es schon bei kleinen Fehlern zu Sprüngen kommen. Diese sprunghaften Änderungen der Kamerapose bezeichnet man als *Jitter*.

2.5 Bestimmung von Punktkorrespondenzen

2.5.1 Extraktion lokaler Features

Zur Extraktion von Punkten, die stabil verfolgt werden können, existieren eine Reihe von Punktdetektoren. Solche interessante Punkte sind in den meisten Fällen Punkte, deren Gradienten in mehrere Richtungen groß sind. Häufig verwendete Eckendetektoren sind der Harris-Eckendetektor [HS88] und der Kanade-Lucas-Tomasi-Detektor (KLT) [ST94].

Mit Hilfe des KLT-Operators kann man darüber hinaus eine Vorhersage über den ungefähren Ort des interessanten Pixels im darauf folgenden Bild machen. So lässt sich unter der Annahme einer kontinuierlichen Bildfolge die Verschiebung im nächsten Bild berech-

nen. Bleser verwendet in [Ble04] aufgrund seiner guter Performanz den KLT-Tracker zum Verfolgen eines Automaten.

Bei der Interpretation der Punktverfolgung als Klassifikationsproblem sind insbesondere zwei Aspekte zu beachten. Zum einen gibt es verschiedene Möglichkeiten Punkte zu beschreiben, zum anderen gibt es verschiedene Möglichkeiten die Ähnlichkeit dieser Beschreibungen (oder: Deskriptoren) zu bewerten. Der folgende Abschnitt gibt einen Überblick über verschiedene Deskriptoren und Ähnlichkeitsmaße.

2.5.2 Deskriptoren

Um Korrespondenzen von Punkten des Referenz- und des aktuell betrachteten Videobildes zu bestimmen, müssen die Bildregionen in der lokalen Nachbarschaft der Punkte verglichen werden. Diese markante Regionen bezeichnet man als Merkmale.

Ein einfacher Ansatz ist es, einen Punkt durch die Intensitätswerte der Punktumgebung zu beschreiben. Die so extrahierten Teilbilder bezeichnet man auch als *Patch*. Der Vergleich von kompletten Patches ist jedoch sehr aufwendig. Da bei der Verfolgung potentiell eine sehr große Menge von Vergleichen geleistet werden muss, ist dieser Ansatz in der Regel unbrauchbar. Die Intensitätsinformationen pro Pixel sollten deshalb durch eine geeignete Abbildung reduzieren werden. Dabei sollte der Informationsverlust jedoch möglichst gering gehalten werden.

Nach der Erfassung der Features werden für diese also Deskriptoren berechnet, die diese visuellen Merkmale kodieren. Es gibt eine Vielzahl verschiedener Ansätze zur Berechnung von Deskriptoren. Eine Auswahl dieser Ansätze wird in diesem Abschnitt vorgestellt.

Eigen-Image-basierte Deskriptoren

Bleser untersucht in [Ble04], wie gut sich verschiedene Deskriptoren zur Verfolgung von Punkten eignen. Sie erhält dabei für Eigen-Image-basierte Deskriptoren sehr gute Ergebnisse. Auch Lepetit verwendet in [LLF05] Eigen-Image-basierte Deskriptoren zur Beschreibung von lokalen Features.

Die Idee der Eigen-Image-Repräsentation geht auf Turk [TP91] zurück, der sie zur Klassifizierung von Gesichtsbildern verwendet. Der im Folgenden beschriebene Algorithmus zur Berechnung und zum Vergleich von Eigen-Image-basierten Deskriptoren basiert auf den Ausführungen in [Pau01].

Zur Berechnung des Deskriptors $\boldsymbol{c}$ für einen Bildausschnitt werden im ersten Schritt die Intensitätswerte des Patches als Vektor f umgeordnet. Wird dies für alle betrachteten Bildausschnitte durchgeführt, so erhält man eine Menge von Patch-Vektoren. Auf der Basis dieser Patch-Vektoren kann der Mittelwert-Patch-Vektor μ berechnet werden. Den Deskriptor $\boldsymbol{c}$ zu einem Patch-Vektor $\boldsymbol{f}$ erhält man schließlich durch Multiplikation der Differenz zwischen $\boldsymbol{f}$ und μ mit einer Matrix Φ:

$$\boldsymbol{c} = \Phi(\boldsymbol{f} - \mu) \tag{2.13}$$

Die Spalten der Matrix Φ enthalten die Eigenvektoren einer Matrix $\boldsymbol{V}$. Gegeben eine Trainingsmenge von N_a Vektoren, so ergibt sich die Matrix $\boldsymbol{V}$ als:

$$\boldsymbol{V} = [(\boldsymbol{f_1} - \mu)|...(\boldsymbol{f_{N_a}} - \mu)] \tag{2.14}$$

Nach der Berechnung des Mittelwert-Patch-Vektors μ wird dieser von den Patch-Vektoren der Trainingsmenge subtrahiert. Die resultierende Matrix $\boldsymbol{V}$ setzt sich spaltenweise aus diesen Vektoren zusammen.

Die Bildvektoren werden beim Eigen-Image-basierten Ansatz als Linearkombination des Mittelwert-Patch-Vektors und der Eigenvektoren der Matrix $\boldsymbol{V}$ dargestellt. Der für einen Bildvektor berechnete Deskriptor beinhaltet die jeweiligen Koeffizienten dieser Linearkombination. Um die Größe der Deskriptoren zu verkleinern, setzt man die Matrix Φ nur aus den ersten N_v Eigenvektoren zusammen. Als Abstandsmaß für Eigen-Image-basierte Deskriptoren wird im Allgemeinen der euklidische Abstand verwendet.

Deskriptoren im Fourier-Raum

Spies beschreibt in [SR00] einen Algorithmus zur Erkennung von Gesichtern, der auf einer Analyse von Fourier Spektren basiert. Als Deskriptoren der Bilder werden die Wer-

te der Fourier-Transformation für bestimmte Frequenzen verwendet. Die Auswahl dieser Frequenzen wird auf der Basis einer Trainingsmenge von Bildern vorgenommen. Es werden die Frequenzen ausgewählt, deren Varianz am größten ist. Spies erhält sogar bei einer Beschränkungen auf 27 Frequenzen noch sehr gute Resultate.

Bleser zeigt in [Ble04], dass es erfolgversprechend ist, Fourier-Deskriptoren auch bei der modellbasierten Objektverfolgung einzusetzen.

SIFT

Lowe [Low04] beschreibt mit der *Scale Invariant Feature Transform* (SIFT) einen Ansatz, der sowohl die Detektion als auch die Extraktion von lokalen Feature-Deskriptoren beinhaltet. Als Deskriptor von SIFT-Features verwendet Lowe Histogramme von Orientierungen. Jedes Histogramm besteht dabei aus acht Bins und beschreibt eine 4×4-Pixelnachbarschaft. Ein SIFT-Deskriptor besteht schließlich aus vier solchen Histogrammen. Durch eine Normalisierung dieser Werte wird die Invarianz gegenüber Beleuchtungsänderungen verbessert.

SIFT-Deskriptoren sind aufgrund der Invarianz gegenüber Skalierung, Rauschen, Beleuchtungsstärke und des Sichtpunktes sehr robust. Beim Vergleich verschiedener Deskriptoren in [MS05] erzielt SIFT die besten Ergebnisse. Der Aufbau der Bildpyramiden ist jedoch sehr rechenaufwändig, weswegen SIFT laut Lepetit [LLF05] für Echtzeit-Anwendungen nicht schnell genug sei. Da der in dieser Diplomarbeit vorgestellte Ansatz die Pose nicht ausschließlich punktbasiert schätzt, ist der Einsatz von SIFT-Deskriptoren an der Stelle zu rechenaufwendig.

Erweiterungen

Um die Klassifikation zu verbessern, werden in [LVTF03] Modellpunkte nicht nur auf einen einzigen Deskriptor abgebildet. Da die Aufnahmeperspektive von Referenzbild und betrachteter Aufnahme stark variieren kann, können sich korrespondierende Patches stark voneinander unterschieden. Um die Wahrscheinlichkeit zu erhöhen, dass auch diese Korrespondenzen extrahiert werden, transformiert Lepetit die extrahierten Patches. Jedem inter-

essanten Punkt werden also zusätzlich zu dem Originalpatch mehrere synthetisch erzeugte Patches zugeordnet. Die transformierten Patches sollten eine möglichst hohe Ähnlichkeit mit den Patches haben, die man aus anderen Perspektiven um den betrachteten Punkt erhalten würde. Nimmt man an, dass das Objekt an der betrachteten Stelle planar ist, so kann dies durch eine affine oder homographische Transformation realisiert werden.

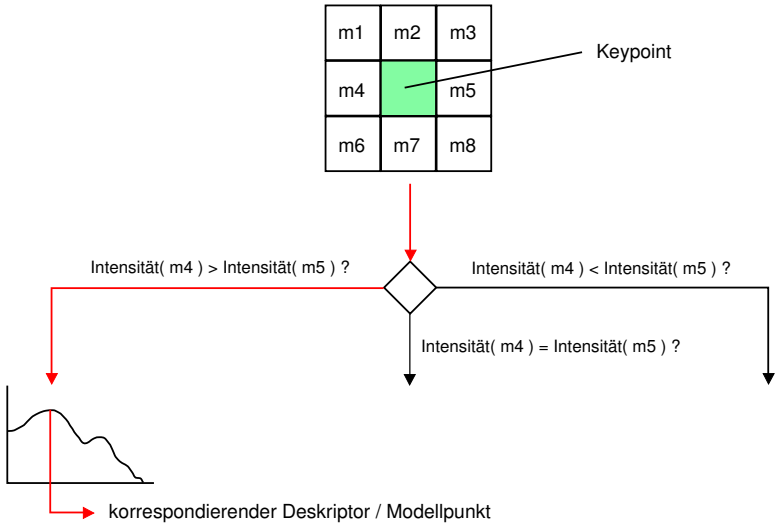

Bild 2.4: Randomized Trees als Klassifikator.

Ein Nachteil dieses Vorgehens ist, dass es aufgrund der zusätzlich online durchzuführenden Vergleiche rechenintensiver ist. Lepetit verwendet deshalb in [LLF05] zur Klassifikation von Keypoints (d. h. interessanten Punkten) *Randomized Trees*. Zu klassifizierende Keypoints werden in diesen Bäumen nach unten bewegt bis ihnen am Ast des Baumes ein Modellpunkt zugeordnet wird. Die Tests an den Knoten vergleichen Pixelintensitäten in der Umgebung des betrachteten Keypoints. Auf der Basis von Trainingsdaten wird offline für jedes Blatt eine Verteilung von bedingten Wahrscheinlichkeiten bestimmt. Die Keypoints werden schließlich dem Deskriptor bzw. Modellpunkt zugeordnet, für den die Wahrscheinlichkeit maximal ist. Abbildung 2.4 skizziert das Vorgehen. Die Auswahl der Tests zum Aufbau eines Randomized Trees wird ebenfalls auf der Basis der Trainingsdaten durchgeführt. Lepetit gibt an, dass die Klassifikation mit Randomized Trees eine

echtzeitfähige Objektverfolgung ermöglicht.

2.6 Linienbasiertes Tracking

Im industriellen Umfeld ist die Umgebung, die zur Poseberechnung genutzt werden soll, meist wenig texturiert. Punktbasierte Ansätze funktionieren hier meist nur schlecht. Die Tatsache, dass man in diesem Umfeld häufig scharfe Kanten vorfindet, lässt es sinnvoll erscheinen, die Konturen von Objekten zu verfolgen. Einen erster Ansatz zur Verfolgung von Kanten beschreibt Harris in [Har92]. Das vorgestellte Tracking System RAPiD schätzt die Pose auf der Basis von Modellpunkten, die sich auf den Kanten des Modells befinden. Zu diesen so genannten *Kontrollpunkten* werden korrespondierende Bildpunkte im Kamerabild gesucht. Eine ungefähre Schätzung der Position des korrespondierenden Punktes liefert die letzte Kamerapose. Von dieser Position ausgehend wird zur Bestimmung der genauen Position eine eindimensionale Suche nach einer Kante durchgeführt. Die so extrahierten Punktkorrespondenzen erlauben eine lineare Berechnung der Kamerapose.

Der Tracking-Algorithmus in [WVS05] basiert auf dem RAPiD-Tracker, schätzt die Pose jedoch durch eine Minimierung des Reprojektionsfehlers. Der zu minimierende Fehler wird als Summe von Abständen zwischen Punkten und Linien berechnet. Zur Bestimmung von Korrespondenzen zwischen einer Modelllinie und Bildpunkten werden die Konturen des zu verfolgenden Objektes in das Kamerabild projiziert. Bei der Suche nach solchen Linien beschränkt sich auch Wuest auf eine eindimensionale Suche. Er verwendet den Begriff *Suchlinien*[4] für diese orthogonal zu den projizierten Linien verlaufenden Linien. Entlang dieser Suchlinien werden die Punkte ermittelt, die mit hoher Wahrscheinlichkeit zu der Linie im Bild gehören, die die aktuell projizierte Modelllinie abbildet. Dazu wird für die Pixel auf diesen Linien eine Antwort berechnet. Pixel, für die eine hohe Antwort berechnet wird, sind mit hoher Wahrscheinlichkeit Teil der gesuchten Kante. Zur Berechnung der Antwort faltet Wuest das Bild nacheinander mit einer gerichteten Gauss- und einer Gradientenmaske der Form [-1 0 1]. Die Gaussmasken berücksichtigen die Orientierung der projizierten Linie insofern, dass sie die Pixelwerte nur parallel zu dieser Linie

[4] engl. search lines

glätten.

Eine beispielhafte Extraktion von Punkten nach dem beschriebenen Verfahren zeigt Abbildung 2.5. Zwei Modelllinien bzw. Kanten wurden auf der Basis der letzten Pose in das Bild projiziert. Orthogonal zu diesen zwei grün eingezeichneten Linien verlaufen rote Suchlinien. Für jede Suchlinie wurde der Punkt mit der höchsten Antwort mit einem blauen Kreuz markiert.

Die Tracking-Systeme von Marchand [MC05] und Reitmayr [RD06] extrahieren ebenfalls zu Modelllinien korrespondierende Punkte. Marchand faltet zur Berechnung der Antwort die betroffenen Pixel mit orientierten Gradientenmasken der Größe 7×7. Er verwendet wie Wuest aus Gründen der Performanz vorberechnete Masken. Reitmayr verwendet zur Punktextraktion den Canny-Kantendetektor [Can86].

Wenn unter den entlang einer Suchlinie berechneten Antworten kein eindeutiges Maximum auszumachen ist, ist die Wahrscheinlichkeit hoch, dass man der betrachteten Linie einen falschen Bildpunkt zuordnet. Aus dem Grunde berücksichtigt Wuest [WVS05] bei der Schätzung der Pose mehrere Hypothesen. Eine Kombination von punkt- und linienbasiertem Tracking beschreibt Vacchetti in [VLF04]. Dazu erweitert er seinen punktbasierten Tracker [LVTF03] so, dass er zusätzlich Kanten berücksichtigt. Auch dieser Tracking-Algorithmus extrahiert zur Schätzung der Lage von Linien mehrere Hypothesen. Vacchetti gibt an, dass der Algorithmus trotz des höheren Aufwands echtzeitfähig sei.

Im Gegensatz zu den bisher betrachteten Ansätzen verwendet Reitmayr in [RD06] statt eines Drahtgittermodells ein grob texturiertes Modell. Hierdurch kann die Qualität der Linienextraktion verbessert werden, da mit Hilfe der Texturen online entschieden werden kann, welche Linien gut extrahierbar sind. Um dies entscheiden zu können, wird das Modell auf der Basis der bisherigen Pose gerendert. Nach der Extraktion von Linien im gerenderten Bild wird im Videobild nur nach den dort gefundenen Linien gesucht.

Die betrachteten Ansätze sind dem ursprünglichen RAPiD-Ansatz alle sehr ähnlich. Als alternatives Vorgehen könnte man bottom-up eine Linienextraktion durchführen und Korrespondenzen zwischen diesen und den Modelllinien bestimmen. Lepetit gibt in [LF05] jedoch an, dass solche Ansätze aufgrund von mangelhafter Verlässlichkeit und geringer Geschwindigkeit keine praktische Anwendung mehr finden.

Bild 2.5: Extraktion von zu Modelllinien korrespondierenden Bildpunkten. Die projizierten Modelllinien sind grün eingezeichnet. Die Maxima auf den orthogonal verlaufenden roten Suchlinien sind mit blauen Kreuzen markiert.

Kapitel 3

Eigener Ansatz

3.1 Einleitung

Das Ziel dieser Diplomarbeit ist es, eine einfach und sinnvoll strukturierte Architektur zu entwickeln, mit Hilfe derer ein beliebiges Objekt verfolgt werden kann. Das Verfahren, mit dem das Tracking geleistet wird, ist modellbasiert und berechnet die Kamerapose auf der Basis von extrahierten Punkt- und Linienkorrespondenzen. Die Punktkorrespondenzen werden durch ein Feature Matching mit Referenzbildern bestimmt. Die Vorgehensweise ist insofern vergleichbar mit der von Vacchetti [VLF04].

Die Umsetzung des Trackings basiert auf zwei Komponenten: einem Referenzdaten-Modul und einem Tracking-Modul. Mit Hilfe des Referenzdaten-Moduls wird nach der Analyse eines Eingabebildes ein Referenzdatensatz erzeugt. Dieser Referenzdatensatz wird vom Tracking-Modul genutzt, um online die Kamerapose für die Bilder des Eingabe-Datenstroms zu bestimmen. Eine Übersicht über die genannten Anwendungsfälle zeigt das Anwendungsfalldiagramm aus Abbildung 3.1. Der beschriebene Ansatz macht also anders als der von Bleser [Ble04] verwendete Ansatz keine Unterscheidung zwischen Initialisierung und Tracking. Die Pose wird ausschließlich auf der Basis von Korrespondenzen zu Referenzbildern geschätzt. Die Qualität der Objektverfolgung kann somit nicht durch einen Drift der Kamerapose beeinträchtigt werden.

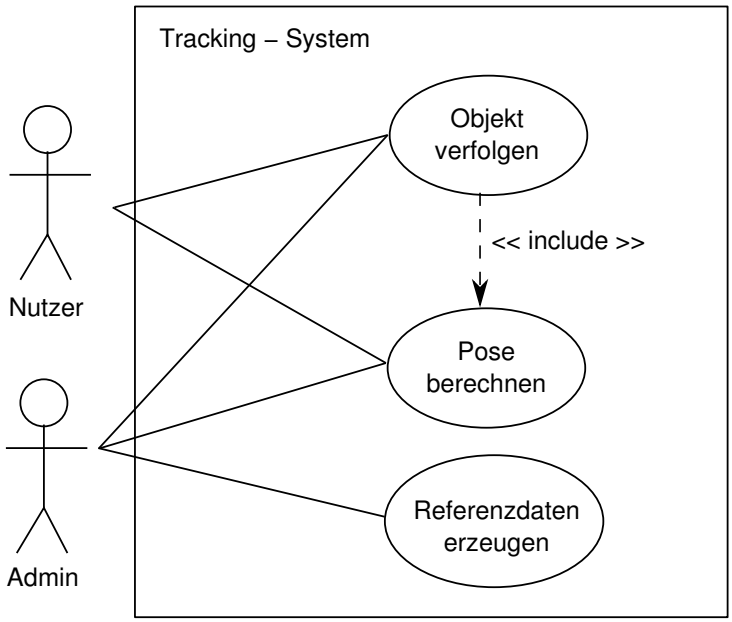

Bild 3.1: Zu betrachtende Anwendungsfälle.

In den folgenden zwei Abschnitten erfolgt eine detaillierte Beschreibung der Umsetzung der beiden erwähnten Module.

3.2 Referenzdaten-Modul

Das Tracking-Modul benötigt für die Schätzung der Kamerapose sowohl Korrespondenzen zwischen Deskriptoren und Modellpunkten, als auch die Information, aus welchen Linien das 3-D-Modell besteht. Für diese Informationen wird im Folgenden der Begriff *Referenzdaten* verwendet. Das Referenzdaten-Modul erzeugt pro Referenzbild einen *Referenzdatensatz*. Das Tracking-Modul kennt die ungefähre Pose und bestimmt basierend auf dieser Information, den entsprechenden Referenzdatensatz (vgl. Abbildung 3.6).

Das Aktivitätsdiagramm in Abbildung 3.2 gibt einen Überblick über den Datenfluss und die Schnittstellen des Referenzdaten-Moduls. Das Referenzdaten-Modul erhält als Einga-

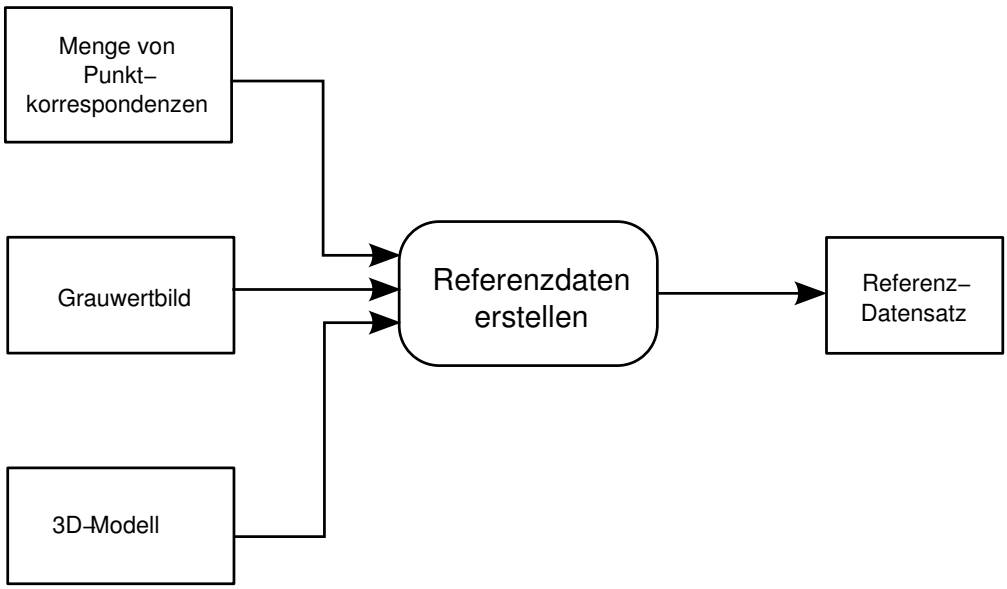

Bild 3.2: Übersicht über den Datenfluss des Referenzdaten-Moduls.

be Grauwertbilder und ein einfaches 3-D-Modell. Weiterhin wird eine Menge von Punktkorrespondenzen zwischen 3-D-Modell und Eingabebild als Eingabe benötigt. Auf der Basis dieser Korrespondenzen wird die Projektionsmatrix bzw. die Pose für das Referenzbild geschätzt. Mit Hilfe kommerzieller Tools kann die Poseberechnung zwar weitaus komfortabler geleistet werden – das realisierte Programm soll jedoch unabhängig von externen Programmen sein.

Das erwähnte 3-D-Modell wird aus einer Datei eingelesen und besteht aus einer Menge von Linien und Dreiecken – das Klassendiagramm aus Abbildung 3.3 zeigt den Aufbau der Datenstruktur. Das 3-D-Modell erhält man üblicherweise durch manuelles Ausmessen. Das Referenzdaten-Modul liest die Modelldaten schließlich aus einer externen Datei ein. Es folgt eine detaillierte Beschreibung der einzelnen Schritte, die zur Erstellung der Referenzdaten nötig sind.

Im ersten Schritt wird die Projektionsmatrix des Referenzbildes linear auf der Basis von Punktkorrespondenzen berechnet. Die Korrespondenzen zwischen 3-D-Punkten auf dem

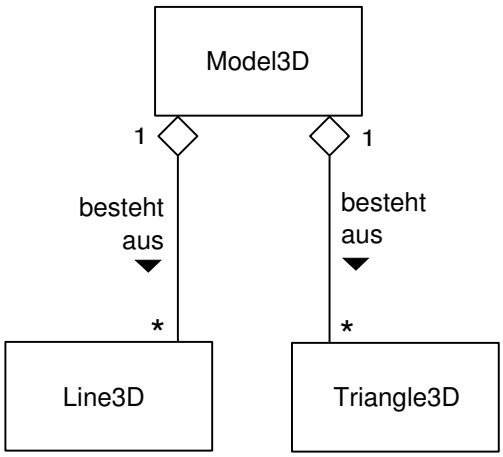

Bild 3.3: Klassendiagramm der Datenstruktur “Model3D”.

Modell und Bildpunkten im Referenzbild werden im Allgemeinen durch Vermessen des Modells manuell bestimmt. Die Daten werden in einer Datei eingetragen und vom Referenzdaten-Modul eingelesen. Auf der Basis dieser Korrespondenzen wird die Projektionsmatrix geschätzt, wobei die in Abschnitt 2.3.2 vorgestellte DLT verwendet wird. Zur Erhöhung der numerischen Stabilität werden die Punktkorrespondenzen vor dem Aufstellen der Messmatrix normalisiert. Details zur Normalisierung der Eingabedaten wurden in Abschnitt 2.3.3 erläutert.

Im nächsten Schritt sollten mit Hilfe der berechneten Projektionsmatrix die extrinsischen Kameraparameter für das Referenzbild bestimmt werden. Dabei wird im Folgenden davon ausgegangen, dass die intrinsischen Parameter der verwendeten Kamera bekannt sind. Sei $\boldsymbol{K}$ die Kalibriermatrix und $[\boldsymbol{R}|\boldsymbol{t}]$ die Matrix, die die extrinsischen Kameraparameter enthält, dann kann die Projektionsmatrix $\boldsymbol{P}$ als Produkt dargestellt werden:

$$\boldsymbol{P} = \lambda \boldsymbol{K}[\boldsymbol{R}|\boldsymbol{t}] \tag{3.1}$$

Die DLT liefert im Allgemeinen keine normalisierte Matrix, d. h. $\lambda \neq 1$. Die Einträge der Projektionsmatrix seien wie folgt nummeriert:

$$\boldsymbol{P} = \begin{pmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{pmatrix} \tag{3.2}$$

Dann läßt sich der Faktor λ mit Hilfe der Projektionsmatrix wie folgt bestimmen:

$$\lambda = \left\| \begin{pmatrix} p_{31} \\ p_{32} \\ p_{33} \end{pmatrix} \right\| \tag{3.3}$$

Eine detaillierte Behandlung der Thematik *projektive Kamera* erfolgt z. B. in [HZ03].

Das Tracking-Modul verwendet zur Beschreibung der Rotation jedoch nicht die Rotationsmatrix, sondern Euler-Winkel. Aufgrund der Ungenauigkeit von Pixelkoordinaten ist die geschätzte Rotationsmatrix im Allgemeinen nicht orthogonal. Die Berechnung der drei Rotationsparameter ist somit nicht trivial. Die Orthogonalität kann mit Hilfe einer Singulärwertzerlegung erzwungen werden, indem man die Singulärwerte auf den Wert 1 setzt.

Um die Ungenauigkeit der so geschätzten Pose zu verringern, wird die Pose weiterhin dahingehend optimiert, dass der Reprojektionsfehler nicht-linear minimiert wird.

Im zweiten Schritt werden mittels des Harris-Eckendetektors [HS88] Ecken bzw. interessante Punkte extrahiert. Dazu wurde die Implementation des Harris-Operators aus der PUMA-Bildverarbeitungsbibliothek [PH03] genutzt. Die Anwendung eines Schwellwertverfahrens auf die gelieferten Antworten führt zu einer sehr schlechten Verteilung der extrahierten Punkte. Es wird daher nach lokalen Maxima der Antwort des Harris-Operators gesucht. Das Eingabebild wird dazu in Regionen gleicher Größe eingeteilt. Der Pixel einer Region, für den die Antwort maximal wird, wird als interessanter Punkt vorgemerkt. Von diesen vorgemerkten Punkten werden wiederum nur die mit den höchsten Werten als interessante Punkte extrahiert. Es zeigte sich, dass man gute Ergebnisse erhält, wenn man 50% der vorgemerkten Punkte extrahiert. Abbildung 3.4 zeigt das Ergebnis der Punktextraktion an einem Beispielbild.

Im nächsten Schritt werden den extrahierten Bildpunkten 3-D-Punkte auf dem Modell zugeordnet. Zur Rückprojektion eines Bildpunktes auf das Modell wird der Projektionsstrahl

Bild 3.4: Die roten Punkte zeigen die mittels Harris-Operator extrahierten Interessenspunkte für ein Beispielbild.

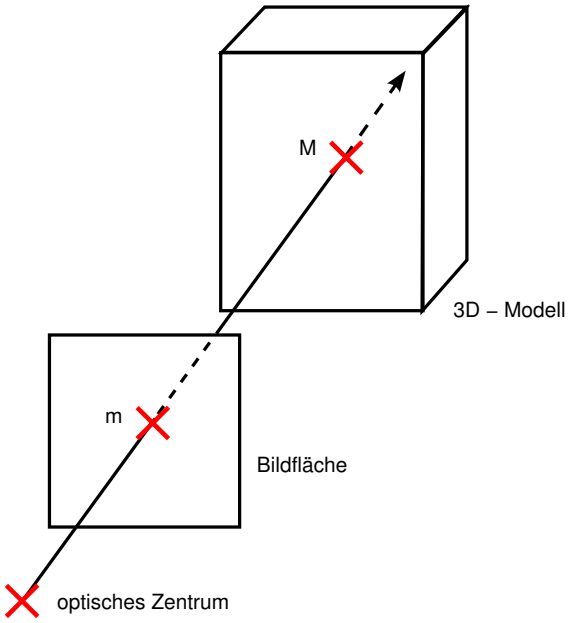

Bild 3.5: Der Strahl durch das optische Zentrum $\boldsymbol{C}$ und den Bildpunkt $\boldsymbol{m}$ schneidet das Modell im Punkt $\boldsymbol{M}$.

dieses Punktes mit allen Dreiecken geschnitten, aus denen das Modell besteht. Der Projektionsstrahl geht vom optischen Zentrum der Kamera aus und schneidet die Bildebene in dem betrachteten Bildpunkt (vgl. Abbildung 3.5). Die folgende Herleitung der Gleichung, die den Projektionsstrahl beschreibt, basiert auf den Ausführungen in [HZ03].

Der gesuchte Strahl geht durch das Kamerazentrum $\boldsymbol{C}$ mit $\boldsymbol{PC} = \boldsymbol{0}$.

Sei $\boldsymbol{P} = [\boldsymbol{P_{3\times3}}|\boldsymbol{p_4}]$, so gilt:

$$\boldsymbol{C} = \begin{pmatrix} -\boldsymbol{P_{3\times3}^{-1}}\,\boldsymbol{p_4} \\ 1 \end{pmatrix} \tag{3.4}$$

Der Projektionsstrahl zu einem Punkt im Pixelkoordinatensystem $\boldsymbol{m}$ geht weiterhin durch den Punkt im Weltkoordinatensystem $\boldsymbol{P^{+}m}$. $\boldsymbol{P^{+}}$ steht dabei für die Pseudoinverse der Projektionsmatrix $\boldsymbol{P}$. Für sie gilt $\boldsymbol{PP^{+}} = \boldsymbol{I}$ und somit $\boldsymbol{P}(\boldsymbol{P^{+}m}) = \boldsymbol{m}$. $\boldsymbol{P^{+}m}$ wird also

auf $\boldsymbol{m}$ projiziert.

Die Berechnung einer Pseudoinversen ist jedoch nicht nötig, wenn man zur Bestimmung des Projektionsstrahls als zweiten Punkt den entsprechenden Punkt auf der uneigentlichen Ebene wählt. Der Grund dafür ist, dass für die Rückprojektion von Punkten auf der uneigentlichen Ebene nur die Teilmatrix $\boldsymbol{P_{3\times3}}$ benötigt wird. Sei $\boldsymbol{D} = (\boldsymbol{d_0}, \boldsymbol{d_1}, \boldsymbol{d_2}, \boldsymbol{0})^T$ ein Punkt auf der uneigentlichen Ebene, so gilt:

$$\boldsymbol{PD} = [\boldsymbol{P_{3\times3}}|\boldsymbol{p_4}]\boldsymbol{D} = \boldsymbol{P_{3\times3}}(\boldsymbol{d_0}, \boldsymbol{d_1}, \boldsymbol{d_2})^T \tag{3.5}$$

Die Matrix $\boldsymbol{P_{3\times3}}$ ist im Gegensatz zu $\boldsymbol{P}$ invertierbar. Der Strahl ist durch zwei Punkte im Weltkoordinatensystem eindeutig bestimmt. Sei $\boldsymbol{M}$ der Punkt auf dem Modell, der durch die normalisierte Projektionsmatrix $\boldsymbol{P}$ auf den Bildpunkt $\boldsymbol{m}$ abgebildet wird, dann wird die Tatsache, dass $\boldsymbol{M}$ auf dem Projektionsstrahl liegt, durch die folgende Gleichung beschrieben:

$$\boldsymbol{X} = \begin{pmatrix} -\boldsymbol{P_{3\times3}^{-1}}\,\boldsymbol{p_4} \\ 1 \end{pmatrix} + \zeta \begin{pmatrix} \boldsymbol{P_{3\times3}^{-1}}\,\boldsymbol{m} \\ 0 \end{pmatrix} \quad , \quad \zeta \in \mathbb{R} \tag{3.6}$$

Extrahierte Punkte, für die der Projektionsstrahl das zu verfolgende Modell nicht schneidet, werden verworfen.

Im vierten Schritt werden lokale Deskriptoren der Punktnachbarschaften der extrahierten Punkte bestimmt. Der im Rahmen dieser Diplomarbeit implementierte Algorithmus betrachtet Punktnachbarschaften der Größe 11×11 Pixel. Die Bewertung verschiedener lokaler Deskriptoren in [Ble04] zeigt, dass sich Eigen-Image-basierte Deskriptoren sehr gut zum modellbasierten Tracking eignen. Da sich auch Lepetit in [LVTF03] bei der Wahl der Deskriptoren für Eigen-Image-basierte Deskriptoren entscheidet, wird der Ansatz auch in dieser Diplomarbeit verwendet.

Zur Abbildung der online extrahierten Patches müssen der bereits in Abschnitt 2.5.2 erwähnte Vektor μ und die Matrix Φ bekannt sein. Die Datenstruktur, die die Referenzdaten kapselt, muss somit eine Beschreibung des mittleren Patches und die Eigenvektor-Matrix enthalten. Abbildung 3.6 zeigt das Klassendiagramm der Referenzdaten-Klasse “RefData”.

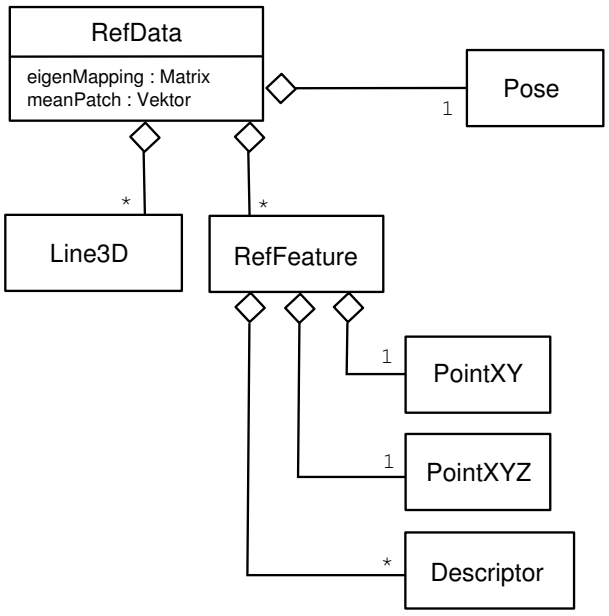

Bild 3.6: Klassendiagramm der Datenstruktur “RefData”.

Schließlich bewertet das Referenzdaten-Modul noch, welche der Modelllinien vom Tracking-Modul verfolgt werden sollten, wenn das aktuelle Referenzbild verwendet wird. Da man von einer großen Ähnlichkeit zwischen Kamera- und Referenzbild ausgehen kann, kann im Allgemeinen schon offline entschieden werden, wie stabil die Extraktion einzelner Linien online sein wird. Dies ist eine sinnvolle Erweiterung bestehender Ansätze, da so recht einfach die Anzahl von Ausreißern verringert werden kann.

Zur Bewertung der Qualität der Linien werden die einzelnen Modelllinien auf das Referenzbild projiziert. Zur Projektion wird die im ersten Schritt geschätzte Projektionsmatrix verwendet. Dann werden äquidistante Punkte auf der projizierten Linie betrachtet. Orthogonal zur projizierten Linie wird an den betrachteten Punkten auf die gleiche Art und Weise eine Antwort berechnet, wie dies das Tracking-Modul realisiert. Ist die berechnete Antwort genau auf der projizierten Linie maximal, so ist die Linie an dieser Stelle potentiell gut zu extrahieren. Gilt dies für mehrere Punkte, so wird die Modelllinie in die Referenzdaten aufgenommen. Das Tracking-Modul wird diese Linie dann verfolgen.

Einen abschließenden Überblick über die einzelnen Schritte des Algorithmus gibt das Aktivitätsdiagramm aus Abbildung 3.7.

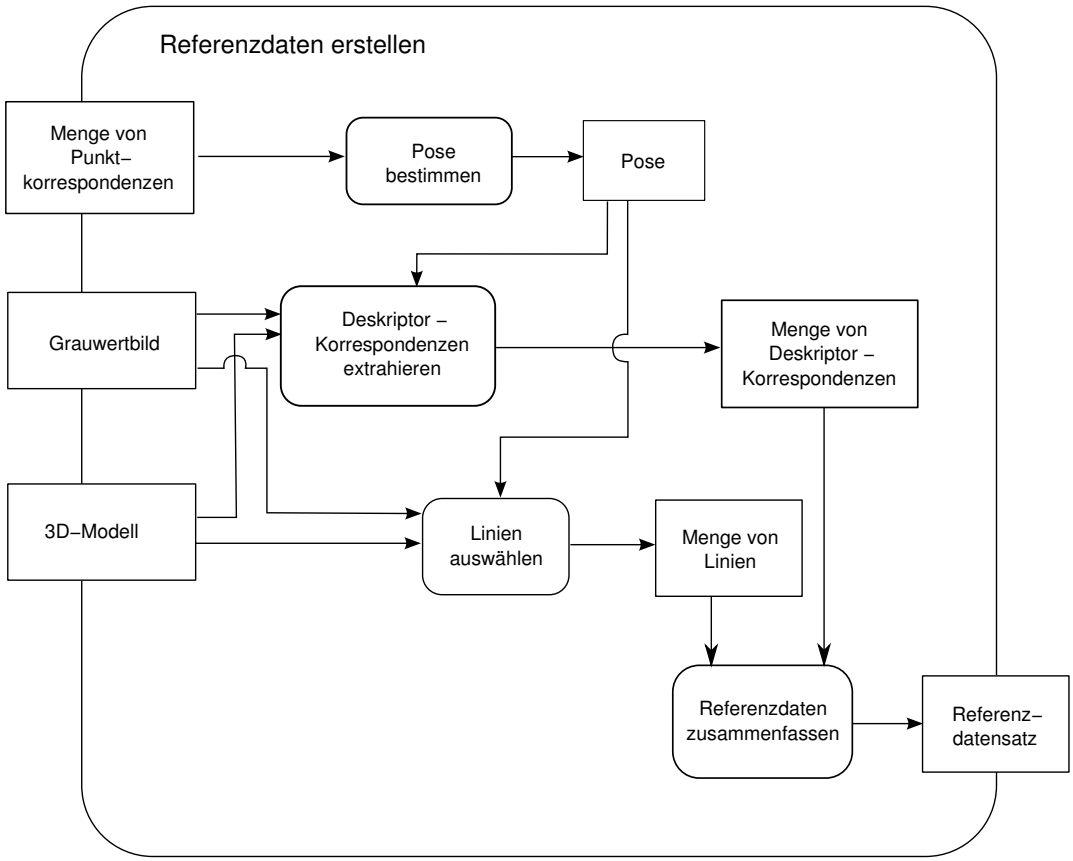

Bild 3.7: Übersicht über den Datenfluss des Referenzdaten-Moduls.

3.3 Tracking-Modul

Das Tracking-Modul erhält als Eingabe einen Strom von Bilddaten, deren Pose der Reihe nach zu bestimmen ist. Die Pose wird auf der Basis von Linien- und Punktkorrespondenzen bestimmt. Dabei ist es insbesondere interessant zu analysieren, inwieweit es sinn-

voll ist, ein punktbasiertes Tracking um ein linienbasiertes Tracking zu erweitern. Während beim punktbasierten Tracking Korrespondenzen zwischen 2-D- und 3-D-Punkten bestimmt werden, sind beim linienbasierten Tracking verschiedene Ansätze denkbar. So könnte man Korrespondenzen zwischen 3-D-Linien und 2-D-Punkten herstellen, wie dies z. B. Wuest in [WVS05] vorschlägt. Es ist aber auch denkbar, Korrespondenzen zwischen 3-D- und 2-D-Linien herzustellen.

Die Bestimmung von Punkt- und Linien-Korrespondenzen ist dabei weitgehend unabhängig voneinander und potentiell parallelisierbar. Auch deswegen ist es interessant, zu prüfen, inwieweit ein linienbasiertes Tracking ein punktbasiertes Tracking aufwerten kann. Die Parallelität der Berechnungen wird gerade bei der Betrachtung des Kontrollflusses des Tracking-Moduls deutlich. Abbildung 3.8 gibt einen Überblick über die einzelnen Schritte des Algorithmus.

Im weiteren Verlauf wird die Extraktion von Punkt- und Linienkorrespondenzen jeweils in einem einzelnen Abschnitt erläutert.

3.3.1 Bestimmung von Punktkorrespondenzen

Der im Rahmen dieser Diplomarbeit implementierte Algorithmus geht davon aus, dass zwei Punkte korrespondieren, wenn der Abstand der Deskriptoren dieser Punkte kleiner als ein Schwellwert ist. Dazu müssen genau wie bei der Erstellung der Referenzdaten mit Hilfe des Harris-Operators Punkte extrahiert werden. Die Intensitäten in der lokalen Umgebung der Punkte werden auf Eigen-Image-basierte Deskriptoren abgebildet.

Daraufhin erfolgt ein Vergleich der Deskriptoren des Referenzdatensatzes mit den online extrahierten Deskriptoren. In [Pau01] wird angegeben, dass es sinnvoll ist, als Abstandsmaß zwischen Eigen-Image-basierten Deskriptoren den euklidischen Abstand zu verwenden. Auch der hier vorgestellte Algorithmus berechnet den euklidischen Abstand zwischen den Deskriptoren. Die Entscheidung, ob es sich um korrespondierende Punkte handelt, erfolgt über einen Vergleich des Abstandes mit einem Schwellwert. Ein fest gewählter Schwellwert würde dazu führen, dass die Anzahl der Punktkorrespondenzen sehr stark variiert. Ein zu groß gewählter Schwellwert würde dazu führen, dass die Anzahl der Korrespondenzen für eine gute Poseschätzung zu klein ist. Ein zu klein gewählter Schwellwert

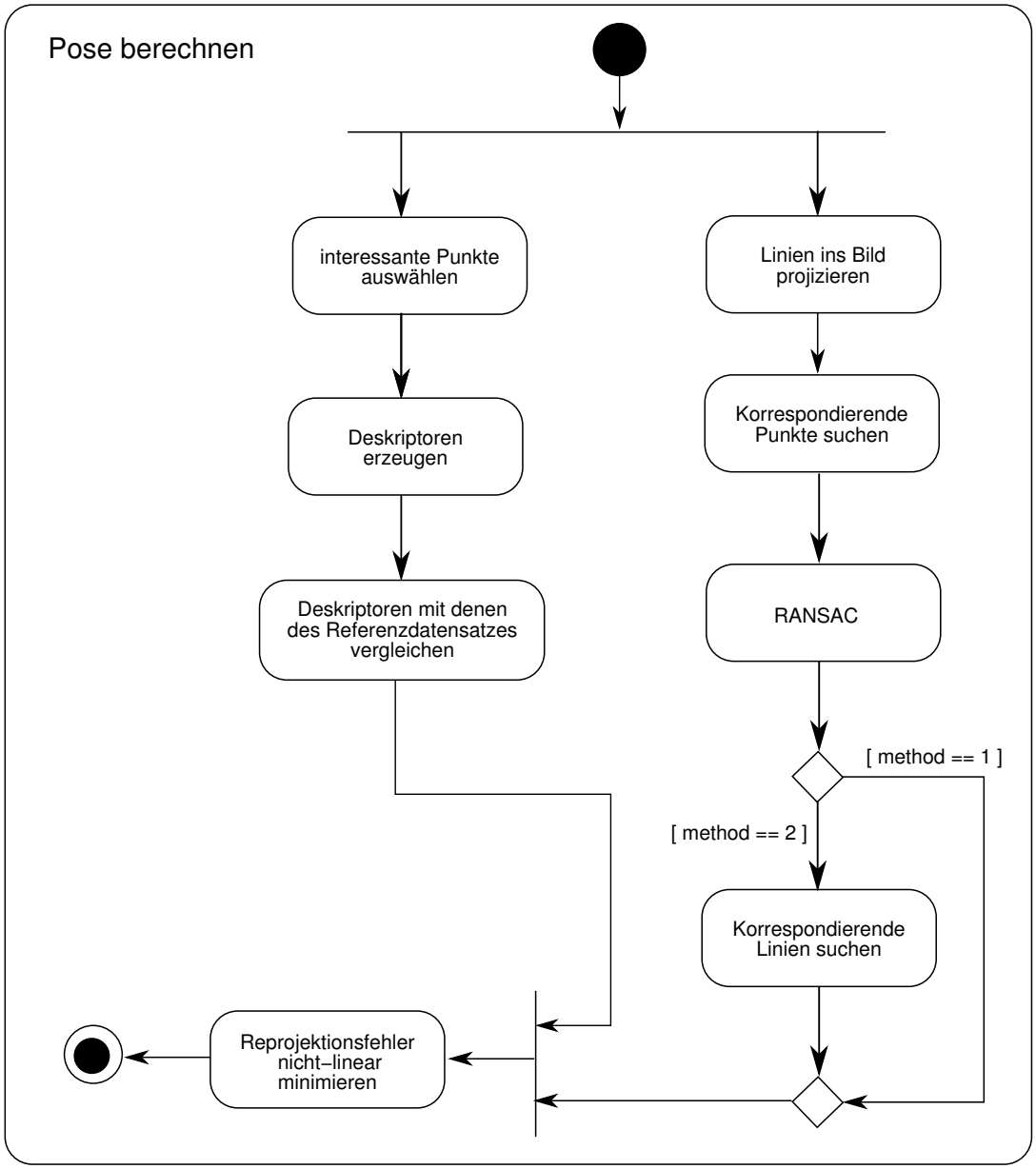

Bild 3.8: Überblick über den Kontrollfluss des Tracking-Moduls.

würde zu einer großen Anzahl von falschen Punktkorrespondenzen führen. Der vorgestellte Algorithmus verwendet aus diesem Grund einen variablen Schwellwert, der verringert wird, wenn die Anzahl der extrahierten Korrespondenzen zu gering ist. Durch gut gewählte Schwellwerte kann vermieden werden, dass die Extraktion der Korrespondenzen oft wiederholt werden muss.

Als Ergebnis des bisherigen Vorgehens können einzelne Punkte des Kamerabildes mit mehreren Punkten des Referenzbildes (bzw. mehreren Modellpunkten) korrespondieren. Da diese Korrespondenzen mit sehr hoher Wahrscheinlichkeit falsch sind, ignoriert der implementierte Algorithmus sie im weiteren Verlauf.

Auf eine Unterdrückung von Ausreißern mit dem RANSAC-Algorithmus wurde aufgrund des hohen Rechenaufwands verzichtet. Da Lepetit in [LF05] angibt, dass die Verwendung von M-Schätzern in der Regel schneller als RANSAC ist, wird dieses Vorgehen auch in dieser Diplomarbeit gewählt. Da der vorgestellte Algorithmus für jeden Interessenspunkt nur einen Deskriptor berechnet, ist weiterhin damit zu rechnen, dass die Anzahl fehlerhafter Punktkorrespondenzen groß ist. Hierdurch würde RANSAC in vielen Fällen erst nach einer großen Anzahl von rechenintensiven Iterationen terminieren.

3.3.2 Bestimmung von Linienkorrespondenzen

Es existieren bereits eine Vielzahl von Tracking-Ansätzen, die zur Schätzung der Pose Kanten verfolgen. Diese beschränken sich in der Regel darauf, Punkte mit hohen Gradienten zu betrachten, anstatt komplette Linien zu extrahieren. Auch der in dieser Diplomarbeit vorgestellte Algorithmus basiert auf dem von Harris vorgestellten Vorgehen, Punkte auf kontrastreichen Kanten zu extrahieren. Das Verfahren wird in vielen aktuelle Ansätzen aufgegriffen (z. B. [WVS05] [DC02]) und ist somit erfolgversprechend.

Zu jeder 3-D-Linie des aktuell betrachteten Referenzdatensatzes werden korrespondierende Punkte extrahiert, die mit hoher Wahrscheinlichkeit zu dieser Linie korrespondieren. Dazu wird jede Modelllinie auf der Basis der letzten Pose in das Bild projiziert. In der Umgebung der projizierten Linien wird nach Punkten gesucht, die mit den jeweiligen Modelllinien korrespondieren. Dies ist mit hoher Wahrscheinlichkeit der Fall, wenn ein Punkt auf einer kontrastreichen Kante liegt, die in einem ähnlichen Winkel wie die projizierte

0	0	1	2	1	0	0
0	2	18	35	18	2	0
0	9	66	128	66	9	0
0	0	0	0	0	0	0
0	−9	−66	−128	−66	−9	0
0	−2	−18	−35	−18	−2	0
0	0	−1	−2	−1	0	0

Bild 3.9: Beispiel für eine orientierte Gradientenmaske.

Linie verläuft. Zur Bewertung, ob dies der Fall ist, werden wie in [MC05] vorberechnete orientierte Gradientenmasken verwendet. Abbildung 3.9 zeigt ein Beispiel für eine in der Implementierung dieser Diplomarbeit verwendete Maske. In Anlehnung an den RAPiD-Ansatz beschränkt sich der Suchraum jeweils auf einzelne äquidistante Linien, die orthogonal zu der projizierten Linie verlaufen. Für jede Suchlinie wird der Punkt mit der maximalen Antwort bestimmt. Bereits in Kapitel 2 wurde das Ergebnis der Punktextraktion an einem Beispiel gezeigt (siehe Abbildung 2.5 auf Seite 27). Da auf eine Betrachtung mehrerer Hypothesen verzichtet wird, liefert dieses Verfahren in einigen Fällen falsche Korrespondenzen. Zur Steigerung der Qualität der Poseschätzung werden diese mit einem RANSAC-basierten Verfahren eliminiert. Dazu werden aus den extrahierte Punkten zwei zufällige Punkte $\boldsymbol{m_1}$ und $\boldsymbol{m_2}$ ausgewählt. Für die übrigen Punkte wird bewertet, ob sie auf der direkten Verbindung (d. h. Linie) zwischen $\boldsymbol{m_1}$ und $\boldsymbol{m_2}$ liegen bzw. der Abstand zur Linie unterhalb eines Schwellwertes liegt. Ist dies für eine ausreichende Anzahl von Punkten der Fall, so werden alle Punkte, die nicht auf dieser Linie liegen verworfen, und der Algorithmus terminiert. Ist dies nicht der Fall, werden erneut zwei zufällige Punkte ausgewählt und das Vorgehen wiederholt sich. Schlägt RANSAC bei den Korrespondenzen einer Modelllinie fehl, so wird diese Linie bei der Minimierung des Reprojektionsfehlers

nicht berücksichtigt. Da das Modell (d. h. Linie) einfach ist, ist das RANSAC-basierte Eliminieren von Ausreißern an dieser Stelle nicht besonders rechenintensiv.

Die in Kapitel 2 vorgestellte Fehlerfunktion 2.7 kann so verallgemeinert werden, dass die Berechnung Korrespondenzen beliebiger Primitive bzw. Merkmalen berücksichtigt. Sei $\boldsymbol{m_i}$ das 2-D-Merkmal, das zum 3-D-Merkmal $\boldsymbol{M_i}$ korrespondiert, ρ die Tukey-Funktion und d ein Abstandsmaß, so basiert die Poseschätzung in dieser Diplomarbeit auf der Minimierung der folgenden Fehlerfunktion:

$$r = \sum_i \rho(\, d(\, \boldsymbol{m_i}, \Phi(\boldsymbol{P}, \boldsymbol{M_i}))\,) \tag{3.7}$$

In die Minimierung des Rückprojektionsfehlers können somit Korrespondenzen zwischen Modelllinien und auf dem Kamerabild extrahierten Linien mit einbezogen werden. Eine vom Modell unabhängige Linienextraktion ist jedoch wenig erfolgversprechend (vgl. Abschnitt 2.6). Aus diesem Grunde extrahiert das im Rahmen dieser Diplomarbeit implementierte Programm Linien auf der Basis der extrahierten Korrespondenzen zwischen Bildpunkten und Modelllinien. Die direkte Verbindung zwischen dem ersten und dem letzten dieser Bildpunkte wird als Ausgangspunkt für die Linienextraktion verwendet. In Abbildung 3.10 wurden diese Punkte mit blauen Kreuzen markiert. Die direkte Verbindung wurde in rot eingezeichnet. Ausgehend von dieser Linie wird zu beiden Seiten geprüft, wo sich im Bild der genaue Start- und Endpunkt der Linie befindet. Zur Bestimmung der Linienendpunkte im Bild wird für die Punkte, die sich in direkten Verlängerung der extrahierten Linie befinden, eine Antwort berechnet. Zur Berechnung der Antwort werden erneut orientierte Gradientenmasken genutzt. Ist die Antwort von n aufeinanderfolgenden Pixeln kleiner als m, so wird angenommen, dass es sich um das Ende der Linie handelt. Die getestete Implementierung verwendet $n = 5$ und wählt m in Abhängigkeit von der durchschnittlichen Antwort der Pixel des mittleren Linienabschnittes.

Die Kamerapose kann für ein gegebenes Bild also auf der Basis von

- Punktkorrespondenzen
- Korrespondenzen zwischen Modelllinien und Bildpunkten
- Korrespondenzen zwischen Modelllinien und Linien im Bild

Bild 3.10: Extraktion von zwei Linien. Es wurden jeweils die zwei äußeren der mit dem RAPiD-Ansatz bestimmten Punkte mit blauen Kreuzen eingezeichnet. Die direkte Verbindung wurde als grüne Linie ausgegeben. Die abschließend geschätzten Linien beinhalten zusätzlich die in rot angegebenen Strecken.

berechnet werden.

Wenn man davon ausgeht, dass ein rein linienbasiertes Tracking nur in sehr wenigen Umgebungen erfolgreich sein kann, ergibt sich als Fragestellung für diese Diplomarbeit: Ist es sinnvoll, bei der Minimierung des Reprojektionsfehlers neben Punktkorrespondenzen auch die auf Linien basierenden Korrespondenzen zu berücksichtigen?

Abbildung 3.11 zeigt, dass sich die beschriebenen Zusammenhänge auch in der Architektur des Programms widerspiegeln. Die Strategien zur Bestimmung von Korrespondenzen wurden in Klassen gekapselt, mit denen die Berechnung der Korrespondenzen parametriert werden kann. Wird auf eine Extraktion von Linien verzichtet, so wird die "PointExtractionStrategy" zur Berechnung der Korrespondenzen genutzt - ansonsten die "LineExtractionStrategy".

3.3.3 Abstandsmaße zwischen Liniensegmenten

Zur Poseberechnung werden die extrinsischen Kameraparameter nicht-linear optimiert. Mit der sich in jedem Schritt als $\boldsymbol{P} = \boldsymbol{K}[\boldsymbol{R}|\boldsymbol{t}]$ ergebenden Projektionsmatrix werden die Modelllinien ins Bild projiziert. Eine so projizierte Linie sollte nun möglichst exakt der aus dem Grauwertbild extrahierten Linie entsprechen. Zur Bewertung, inwieweit dies der Fall ist, benötigt man Metriken, die die Ähnlichkeit von Linien bewerten. Im Folgenden werden einige solcher Abstandsmaße vorgestellt; drei von ihnen wurden in die Implementation des Tracking-Moduls aufgenommen.

Winzen verwendet in [Win94] als Distanzmaß für Linien und Linienzüge unter anderem die sogenannte Hausdorff-Distanz. In Anlehnung an die Definition von Winzen beschreibt $\mathrm{im}(l_i)$ die Menge von Vektoren bzw. Punkten auf der Linie l_i. Die Hausdorff-Distanz $d_{\mathrm{Hausdorff}}$ der Linien l_1 und l_2 ist definiert als:

$$\Delta_{l_1 \to l_2} = \max_{\boldsymbol{x} \in \mathrm{im}(l_1)} \min_{\boldsymbol{y} \in \mathrm{im}(l_2)} \|\boldsymbol{x} - \boldsymbol{y}\| \tag{3.8}$$

$$\Delta_{l_2 \to l_1} = \max_{\boldsymbol{x} \in \mathrm{im}(l_2)} \min_{\boldsymbol{y} \in \mathrm{im}(l_1)} \|\boldsymbol{x} - \boldsymbol{y}\| \tag{3.9}$$

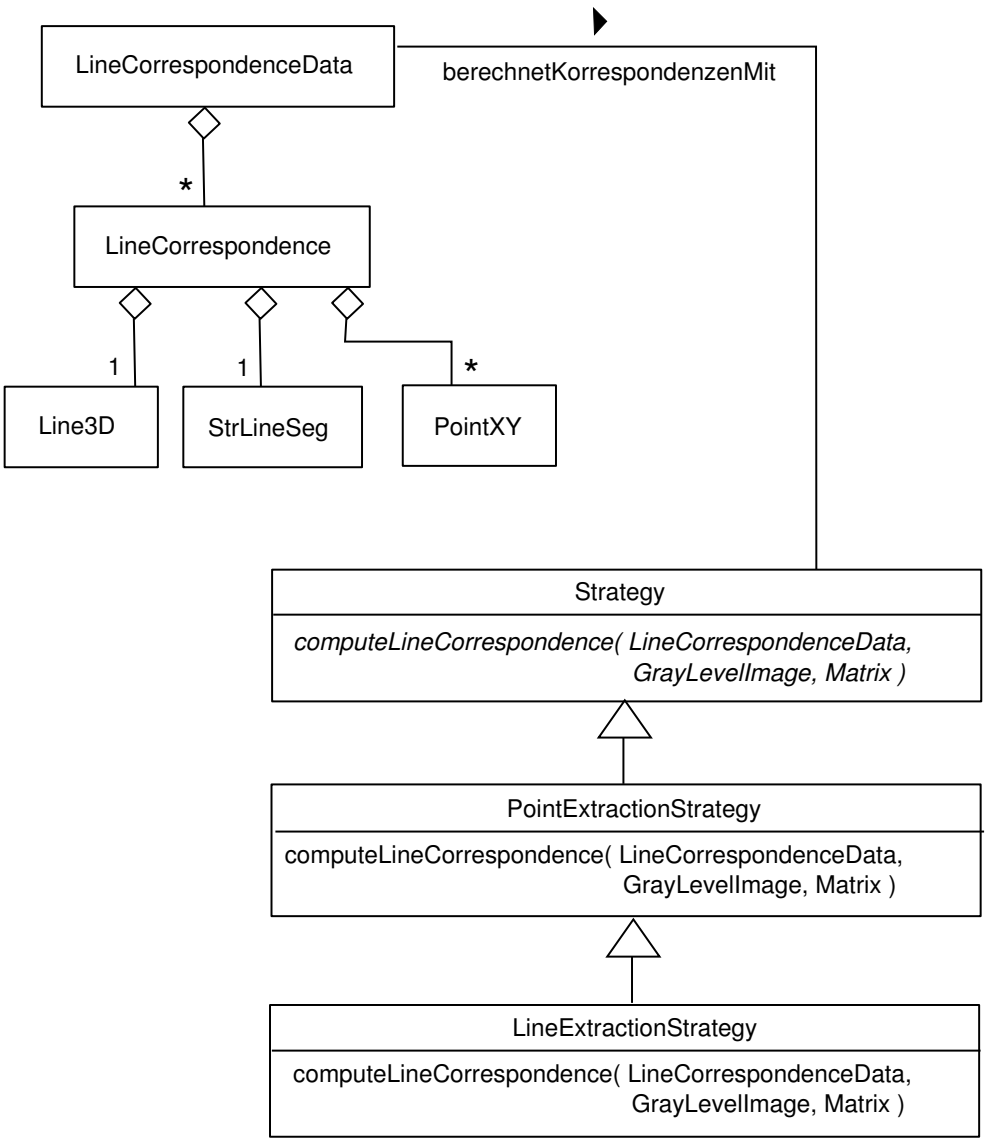

Bild 3.11: Klassendiagramm “LineCorrespondenceData”.

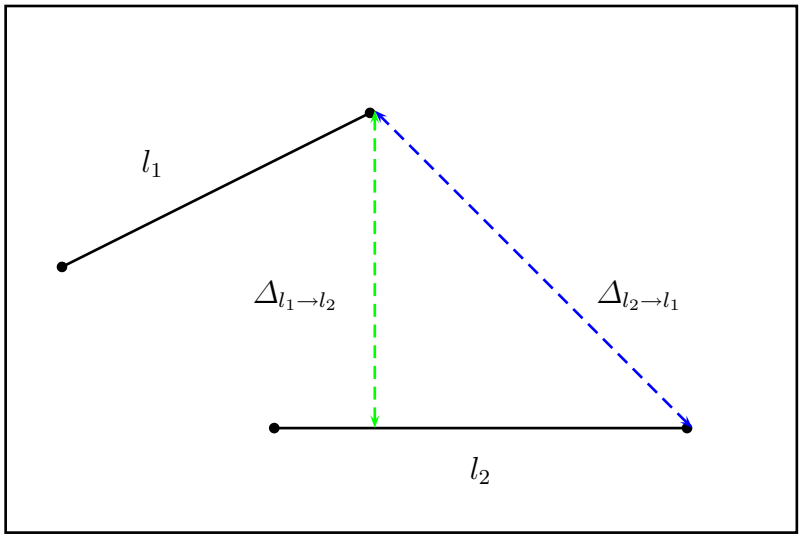

Bild 3.12: Visualisierung der wichtigsten Größen für die Berechnung der Hausdorff-Distanz. Da $\Delta_{l_2 \to l_1}$ in diesem Fall größer als $\Delta_{l_1 \to l_2}$ ist, beschreibt hier der in blau eingezeichnete Abstand die Hausdorff-Distanz.

$$d_{\text{Hausdorff}}(l_1, l_2) = \max\left(\Delta_{l_1 \to l_2}, \Delta_{l_2 \to l_1}\right) \tag{3.10}$$

Es wird also jedem Punkt der Linie l_1 ein Punkt der Linie l_2 mit minimalem Abstand zugeordnet. Das Maximum dieser Punktdistanzen wird mit $\Delta_{l_1 \to l_2}$ bezeichnet. Weiterhin wird jedem Punkt der Linie l_2 ein Punkt der Linie l_1 mit minimalem Abstand zugeordnet. Das Maximum dieser Punktdistanzen wird mit $\Delta_{l_2 \to l_1}$ bezeichnet. Die Hausdorff-Distanz ist das Maximum von $\Delta_{l_1 \to l_2}$ und $\Delta_{l_2 \to l_1}$. Abbildung 3.12 macht dies mit einer Skizze deutlich.

Da das Tracking-Modul nur gerade Liniensegmente verfolgt, kann man sich auf die Betrachtung der Start- und Endpunkte der Linien beschränken. Somit handelt es sich in diesem Kontext um ein einfaches und sehr schnell zu berechnendes Abstandsmaß.

Zur Beurteilung der Segmentierungsqualität von Linien verwendet Harbeck in [Har96] das mittlere Abstandsquadrat aller Punkte einer Linie zu einer anderen Linie. Er erwähnt weiterhin, dass auch die Fläche zwischen zwei Linien als Abstandsmaß verwendet werden

kann. Beide Ansätze wurden jedoch im Rahmen dieser Diplomarbeit nicht implementiert, da sie sehr aufwendig zu berechnen sind.

Trucco definiert in [TV98] ein Maß μ_{pro} [1], das die Nähe von Bildlinien bewertet. Dabei berücksichtigt er, dass es sich bei den Bildlinien um Projektionen von Modelllinien handelt. Gibt es einen großen Unterschied zwischen den Längen der Linien im Bild, so ist es wahrscheinlich, dass auch der Abstand vom Projektionszentrum sehr unterschiedlich ist. Die Ähnlichkeit zweier Linien ist in dem Fall als sehr gering zu bewerten. Abbildung 3.13 verdeutlicht dies anhand einer Skizze.

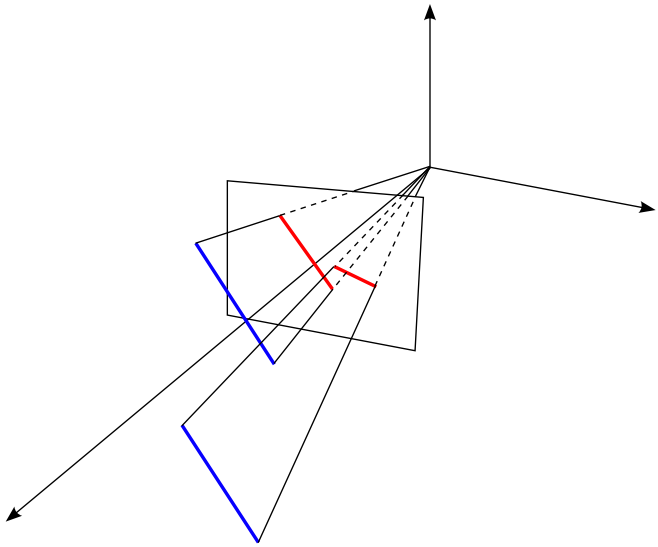

Bild 3.13: Die Längen von 3-D-Linien, die weit voneinander entfernt sind, unterscheiden sich stark.

Sei l_s die kürzere der beiden Linien und g die kürzeste Verbindung zweier Endpunkte der beiden Linien, dann ist μ_{pro} definiert als:

$$\mu_{pro} = \left(\frac{l_s}{g}\right)^2 \tag{3.11}$$

[1] "pro" steht bei Trucco für "proximity", engl. für "Nähe"

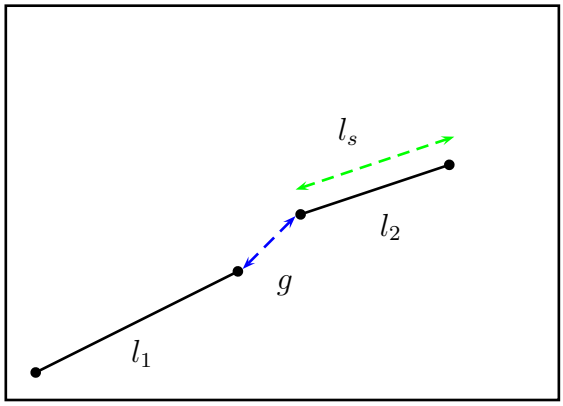

Bild 3.14: Veranschaulichung der zur Bestimmung von μ_{pro} (nach [TV98]) benötigten Größen .

Abbildung 3.14 illustriert die relevanten Größen anhand eines Beispiels. Beschreibt μ_{pro} die Nähe von Bildlinien, so kann der Kehrwert der Funktion als Maß d_{Trucco} für den Abstand zweier Linien betrachtet werden:

$$d_{\text{Trucco}}(l_1, l_2) = \left(\frac{g}{l_s}\right)^2 \tag{3.12}$$

Dass die betrachteten Maße Abstände recht unterschiedlich bewerten, wird an einem kleinen Beispiel deutlich. Die Skizze aus Abbildung 3.15 zeigt drei Linien l_1, l_2 und l_3. Bei der Berechnung der Linienabstände ergeben sich folgende Werte:

$d_{\text{Trucco}}(l_1, l_2) = \frac{1}{3}$
$d_{\text{Trucco}}(l_1, l_3) = \frac{1}{4}$
$d_{\text{Hausdorff}}(l_1, l_2) = \sqrt{2}$
$d_{\text{Hausdorff}}(l_1, l_3) = 2$

Die Hausdorff-Distanz der Linien l_1 und l_2 ist geringer als die Hausdorff-Distanz der Linien l_1 und l_3. Bei Verwendung des Abstandsmaßes d_{Trucco} würde wiederum der Abstand der Linien l_1 und l_3 als geringer bewertet werden.

Um zu bewerten, bei der Verwendung welches der Abstandsmaße der Tracking-Algorithmus die besten Ergebnisse liefert, wurden drei verschiedene Verfahren implementiert. Ne-

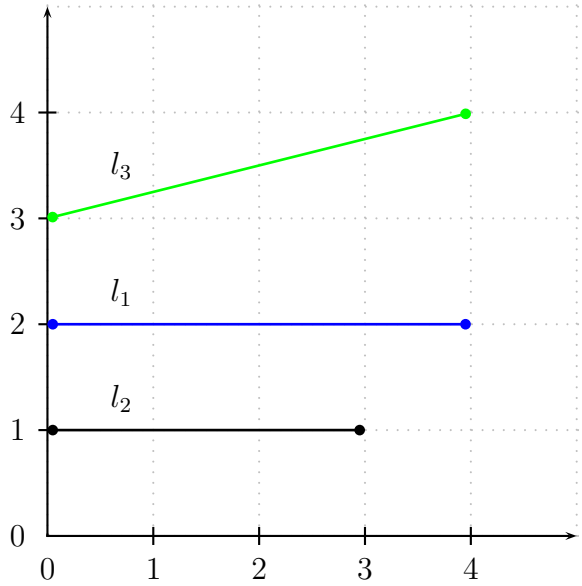

Bild 3.15: Die Hausdorff-Distanz von l_1 und l_2 ist kleiner als die der Linien l_1 und l_3. Demgegenüber ist $d_{\text{Trucco}}(l_1, l_2)$ größer als $d_{\text{Trucco}}(l_1, l_3)$.

ben der in [TV98] eingeführten Metrik und der Hausdorff-Distanz wurde noch ein weiteres Abstandsmaß implementiert. Dieses betrachtet die Mittel- und Endpunkte zweier Linien und kann somit als starke Vereinfachung der von Winzen verwendeten Metrik betrachtet werden. Da die Extraktion der Mittelpunkte im Allgemeinen stabiler als die der Eckpunkte ist, ist es sinnvoll, eine Gewichtung vorzunehmen. Sei $\boldsymbol{s_{l_i}}$ der Startpunkt, $\boldsymbol{e_{l_i}}$ der Endpunkt und $\boldsymbol{m_{l_i}}$ der Mittelpunkt der Linie l_i, so ist der Abstand $d_{\text{MultiplePoints}}$ definiert als:

$$d_{\text{MultiplePoints}}(l_1, l_2) = \|\boldsymbol{s_{l_1}} - \boldsymbol{s_{l_2}}\| + \|\boldsymbol{e_{l_1}} - \boldsymbol{e_{l_2}}\| + 3 \cdot \|\boldsymbol{m_{l_1}} - \boldsymbol{m_{l_2}}\| \tag{3.13}$$

Die relevanten Größen werden in Abbildung 3.16 visualisiert.

3.3.4 Poseschätzung

Die Implementation dieser Diplomarbeit gibt als Ergebnis der Poseschätzung die Pose zurück, für die der Reprojektionsfehler minimal ist. Die zu minimierende Größe ist eine Summe von Abständen. Basiert die Schätzung ausschließlich auf Punktkorrespondenzen,

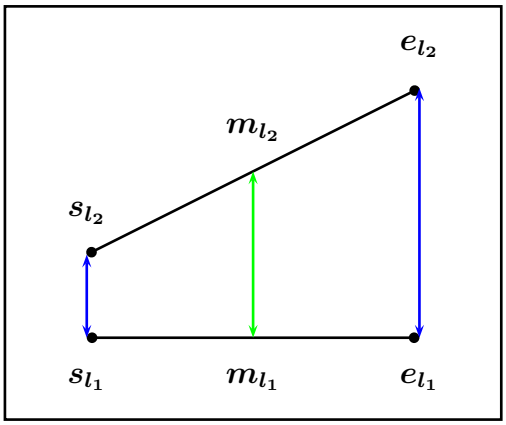

Bild 3.16: Der Abstand $d_{\mathrm{MultiplePoints}}$ der Linien l_1 und l_2 ist die Summe der Längen der zwei blauen Linien und der mit Faktor 3 gewichteten Länge der grünen Linie.

so wird die bereits in Unterabschnitt 2.3.5 vorgestellte Fehlerfunktion 2.7 verwendet.

$$r = \sum_i (\boldsymbol{m_i} - \Phi(\boldsymbol{P}, \boldsymbol{M_i}))^2 \tag{3.14}$$

Werden zusätzlich Korrespondenzen zwischen Modelllinien und Bildpunkten in die Poseschätzung mit einbezogen, so wird die Fehlerfunktion um einen Summanden erweitert. Dieser Summand ist als Summe von quadrierten Abständen zwischen Punkten und Linien definiert.

Für den Abstand zwischen einem Punkt und einer Linie wird im Folgenden die Bezeichnung $d^{\mathrm{P,L}}$ verwendet. $d^{\mathrm{L,L}} = \{d_{\mathrm{Hausdorff}}, d_{\mathrm{Trucco}}, d_{\mathrm{MultiplePoints}}\}$ bezeichnet den Abstand zweier Linien.

Sei i die Anzahl der Punktkorrespondenzen, j die Anzahl der betrachteten Linien und k die Anzahl der zu der betrachteten Linie korrespondierenden Punkte. Sei weiterhin $\boldsymbol{m_i}$ der i-te Bildpunkt und l_j die j-te Linie, dann ergibt sich der Fehler r wie folgt:

$$r = \sum_i (\boldsymbol{m_i} - \Phi(\boldsymbol{P}, \boldsymbol{M_i}))^2 + \sum_j \sum_k (d^{\mathrm{P,L}}(\boldsymbol{m_k}, \Phi(\boldsymbol{P}, l_j)))^2 \tag{3.15}$$

Der Abstand d zwischen einem Punkt und einer Linie wird als Abstand zwischen dem Punkt und dessen Fußpunkt auf der Linie berechnet.

Basiert die Schätzung der Pose auf Punkt- und Linienkorrespondenzen, so ist der zweite Summand der Fehlerfunktion eine Summe von Linienabständen. Sei l_j die j-te Modelllinie und l_j^{estimate} die korrespondierende Linie im Bild, dann ist der Fehler r definiert als:

$$r = \sum_i (\boldsymbol{m_i} - \Phi(\boldsymbol{P}, \boldsymbol{M_i}))^2 + \sum_j (d^{\mathrm{L,L}}(l_j^{\text{estimate}}, \Phi(\boldsymbol{P}, l_j)))^2 \tag{3.16}$$

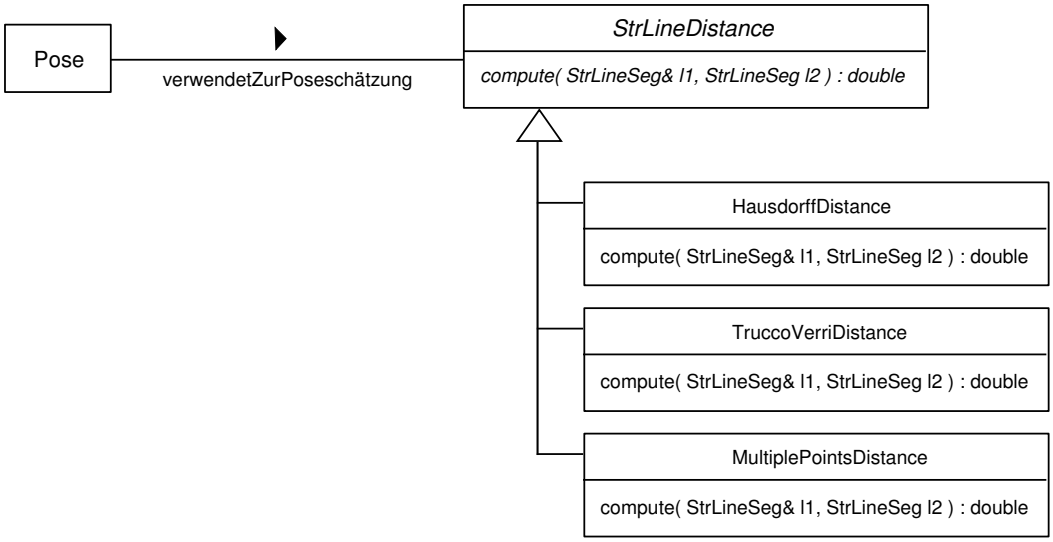

Bild 3.17: Integration der verschiedenen Abstandsmaße.

Die Funktion $d^{\mathrm{L,L}}$ ist je nach Abstandsmaß unterschiedlich definiert. In der Implementation wird die Berechnung des Fehlers mit einer Instanz einer abgeleiteten Klasse von “StrLineDistance” parametriert. Abbildung 3.17 verdeutlicht dies durch ein Klassendiagramm. Würde man zur Berechnung des Fehlers die Formel 3.16 zusammen mit den in Abschnitt 3.3.3 vorgestellten Abstandsmaßen verwenden, so würden die Linienabstände nur wenig Einfluss auf den gesamten Fehler haben. Aus dem Grund werden die einzelnen quadrierten Abstände mit heuristisch ermittelten Gewichtungsfaktoren multipliziert. Um

eine Vergleichbarkeit der Verfahren herzustellen, macht es Sinn, die Definition der Gewichtungsfaktoren abhängig von der Anzahl der Punkte, die pro Linie extrahiert werden, zu machen. Sei NR_TEST_PTS (in Anlehnung an die in der Implementierung verwendete Konstante) die Anzahl der Punkte, die pro Linie extrahiert werden, dann werden folgende Faktoren verwendet:

- HausdorffDistance : $\texttt{NR\_TEST\_PTS} \cdot 0.2$
- MultiplePointsDistance : $\texttt{NR\_TEST\_PTS} \cdot 0.1$
- TruccoVerriDistance : 1000000

Da die Abstandsberechnung der Klasse “TruccoVerriDistance” lediglich ein abstraktes Verhältnis darstellt, wurde auf eine Abhängigkeit von der Konstante NR_TEST_PTS verzichtet.

Da eine globale Minimierung der Fehlerfunktion zu rechenintensiv wäre, wird der Reprojektionsfehler lokal minimiert. Zahlreiche Tracking-Verfahren [LVTF03] [DC02] zeigen, dass eine gute Poseschätzung auch ohne globale Minimierung gelingen kann. Zur Minimierung wird die Implementierung des Downhill-Simplex-Verfahrens der PUMA-Bildverarbeitungsbibliothek verwendet. Das Verfahren gilt als recht robust und liefert z. B. in [Fel05] und [MP00] gute Ergebnisse.

Die zu minimierende Fehlerfunktion erhält als Eingabe sechs Parameter. Jeweils drei Parameter beschreiben die Translation und die Rotation der Kamera. Zur Parametrierung der Rotation wurden Euler-Winkel verwendet. Als initiale Parameter für die Optimierung wurde jeweils die Pose des letzten Bildes verwendet, für das die Kamerapose geschätzt wurde. Die Schätzung der Kamerapose des ersten Bildes eines Bilddatenstroms bildet hier eine Ausnahme. Für sie wird die Pose des Referenzbildes als Ausgangspunkt der Minimierung genutzt. Die Implementierung des Downhill-Simplex-Verfahrens erhält als weiteren Eingabeparameter die Eckpunkte des Startsimplex. Es zeigte sich, dass die Qualität der Poseschätzung sehr abhängig von der Wahl dieses Parameters ist. Eine ungünstige Wahl führt häufig zu einem “Hängenbleiben” in einem lokalen Minimum, das nicht dem globalen Minimum entspricht. Um dies zu verhindern, wird die Minimierung des Reprojektionsfehlers mehrmals mit unterschiedlichen Simplizes durchgeführt.

Kapitel 4

Experimente und Ergebnisse

4.1 Versuchsaufbau der ersten Testsequenz

In Kapitel 3 wurden verschiedene Varianten vorgestellt, wie das Verfolgen eines Objektes realisieren werden kann. Um diese bewerten zu können, wurden der in Kapitel 3 vorgestellte Algorithmus in C++ implementiert. Dabei ermöglicht das entwickelte Programm die Schätzung der Kamerapose auf der Basis von Punkt- und Linieninformationen. Um zu überprüfen, welche der Varianten am erfolgversprechendsten ist, wurde ein Objekt über eine Sequenz von 50 Bildern verfolgt. Eine der Hauptfragestellungen ist es, zu prüfen, ob es in dem gestellten Kontext sinnvoll ist, das punktbasierte Verfolgen von Objekten um ein linienbasiertes Tracking zu erweitern. Weiterhin wird geprüft, welcher der in Abschnitt 3.3.3 vorgestellten Ansätze dafür am besten geeignet ist. Insbesondere wird getestet, welches der Abstandsmaße in der Antwortfunktion verwendet werden sollte.

Da der Übergang zwischen Referenzbildern nicht im Mittelpunkt dieser Diplomarbeit steht, wurde für die Testsequenz nur ein Referenzbild genutzt. Abbildung 4.1 zeigt das Referenzbild, das dem in Kapitel 3 vorgestellten Programm “computeRefData” als Eingabe diente, um die Referenzdaten zu erzeugen. Das zu verfolgende Objekt ist der Karton eines Brettspiels. Der Karton lässt sich durch ein einfaches 3-D-Modell modellieren, dessen Kanten lang genug sind, um ein Verfolgen von Linien sinnvoll erscheinen zu lassen. Aufgrund der starken Texturierung des Kartons ist die Extraktion der Kanten als schwie-

riger einzustufen als das Verfolgen von kaum texturierten Objekten wie z. B. in [DC02]. Die starke Texturierung des Objektes ermöglicht gleichzeitig eine Verfolgung auf der Basis von Punktkorrespondenzen. Das Modell erscheint somit geeignet, den vorgestellten Algorithmus zu testen. Weiterhin weist das Objekt durch die rechteckige Form und die starke Texturierung eine gewisse Ähnlichkeit zu Automaten bzw. Maschinen auf. Bleser verfolgt in [Ble04] z. B. eine Maschine vor dem Hintergrund, die geschätzte Pose im Rahmen eines Augmented Reality-Systems zu verwenden.

Bild 4.1: Das zur Evaluation genutzte Referenzbild.

Das 3-D-Modell, das dem Referenzdaten-Modul als Eingabe dient, ist ein einfacher Quader, der aus acht Linien und zwölf Dreiecken besteht. Abbildung 4.2 zeigt eine Projektion des Drahtgittermodells. Nach Auswertung des Referenzbildes erzeugte das Referenzdaten-Modul einen Referenzdatensatz, der aus zwei Linien besteht. Die zur Evaluation genutzten Bilder wurden einzeln mit einer Kamera des Modells “Fuji Finepix A204” bei einer Auflösung von 960×1280 Pixel aufgenommen. Die intrinsischen Kameraparameter wurden vorab mit Hilfe der *Open Computer Vision Library* (OpenCV) [Int] bestimmt.

Die betrachteten Punktumgebungen haben eine Größe von 11×11 Pixeln; die daraus berechneten Eigen-Image-basierten Deskriptoren bestehen aus 30 Gewichten. Zur Linienextraktion werden orientierte Gradientenmasken der Größe 7×7 verwendet. Wie bereits in

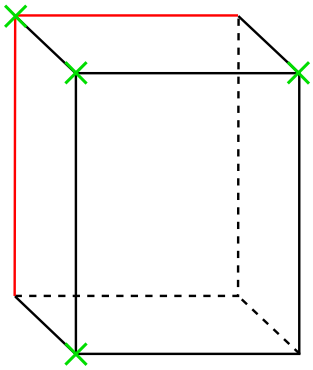

Bild 4.2: Das 3-D-Modell des verfolgten Brettspiels. Die grünen Kreuze markieren die Punkte, für die der Reprojektionsfehler zur Bewertung der Poseschätzung genutzt wird. Die Modelllinien, die bei der Poseschätzung berücksichtigt werden, sind in roter Farbe dargestellt.

Kapitel 3 erwähnt, wird die Minimierung des Reprojektionsfehlers mehrmals mit unterschiedlichen Simplizes durchgeführt. Tests ergaben, dass eine sechsfache Wiederholung mit heuristisch ermittelten Startsimplizes einen guten Kompromiss zwischen Geschwindigkeit und Qualität der Poseschätzung darstellt. Die Bewertung der Poseschätzung sollte weitgehend unabhängig von der Qualität des Referenzdaten-Moduls sein. Aus dem Grund wurden die zu den Bildpunkten korrespondierenden Modellpunkte explizit vorgegeben und nicht durch eine Rückprojektion auf das Modell bestimmt.

4.2 Ergebnisse der ersten Testsequenz

4.2.1 Erste Testsequenz

Die Bewertung der geschätzten Pose für die einzelnen Bilder der getesteten Sequenz erfolgt auf der Basis von vier Modellpunkten. Es handelt sich dabei um die Punkte, die in Abbildung 4.2 mit einem grünen Kreuz markiert wurden. Die geschätzte Projektionsma-

Tracking - Ansatz	Durchschnittlicher Fehler
punktbasiert	89.9076
punktbasiert & RAPiD	68.2578
punkt- & linienbasiert (HausdorffDistance)	110.286
punkt- & linienbasiert (TruccoVerriDistance)	119.015
punkt- & linienbasiert (MultiplePointsDistance)	132.536

Tabelle 4.1: Vergleich des durchschnittlichen Fehlers der fünf Ansätze zur Poseschätzung. Die Berechnung des Fehlers basiert auf Formel 4.1 und wird in Pixel angegeben.

trix ist umso besser, je genauer sie die Modellpunkte auf die korrespondierenden Bildpunkte abbildet. Die genauen Bildpositionen der vier Punkte wurden für jedes Bild manuell bestimmt. Aufgrund von Rauschen kann die Position meist nur auf $\pm$ 3 Pixel genau bestimmt werden. Sei $\boldsymbol{P}$ die zur geschätzten Pose korrespondierende Projektionsmatrix. Sei weiterhin $\boldsymbol{m_i}$ der zu Modellpunkt $\boldsymbol{M_i}$ korrespondierende Bildpunkt, so berechnet sich der Fehler als folgende Summe:

$$r = \sum_{i=1}^{4} \|\boldsymbol{m_i} - \boldsymbol{P}\boldsymbol{M_i}\| \tag{4.1}$$

Eine Gegenüberstellung der fünf getesteten Ansätze zeigt Abbildung 4.3. Für jedes Bild wurde für jedes der Verfahren eine Bewertung der Poseschätzung vorgenommen. Eine Gegenüberstellung des durchschnittlichen Fehlers pro Bild zeigt Tabelle 4.1.

Die Graphik zeigt, dass sich die Qualität des punktbasierten Tracking-Ansatzes verbessert, wenn Informationen über Kanten im Bild berücksichtigt werden. Die Poseschätzung des RAPiD-basierten Ansatzes ist fast über die gesamte Bildsequenz hinweg genauer als die der punktbasierten Objektverfolgung. Eine Erweiterung des punktbasierten Ansatzes um eine Verfolgung von Linien führt jedoch zu einer Verschlechterung der Poseschätzung.

Zur besseren Einordnung der Qualität der Poseschätzung zeigt Abbildung 4.4 vier Bilder der Sequenz. Das Modell wurde jeweils entsprechend der geschätzten Pose auf das Bild projiziert. Um die Werte in Abbildung 4.3 besser einordnen zu können, folgt eine Auswertung der Korrespondenzen, auf denen die Schätzung der Pose basiert. Abbildung 4.5 stellt

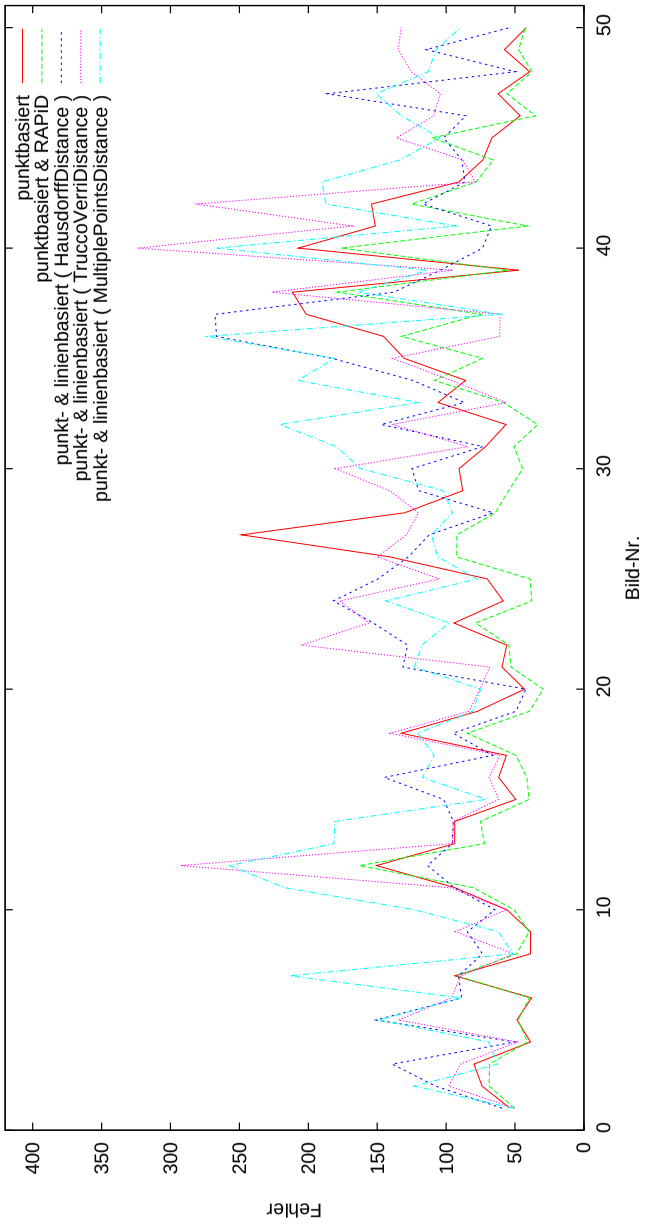

Bild 4.3: Vergleich der fünf Ansätze zur Schätzung der Pose. Die Berechnung des Fehlers basiert auf Formel 4.1. Der Fehler wird in Pixel angegeben.

Bild 4.4: Visualisierung der geschätzten Pose für vier Beispielbilder der Testsequenz.

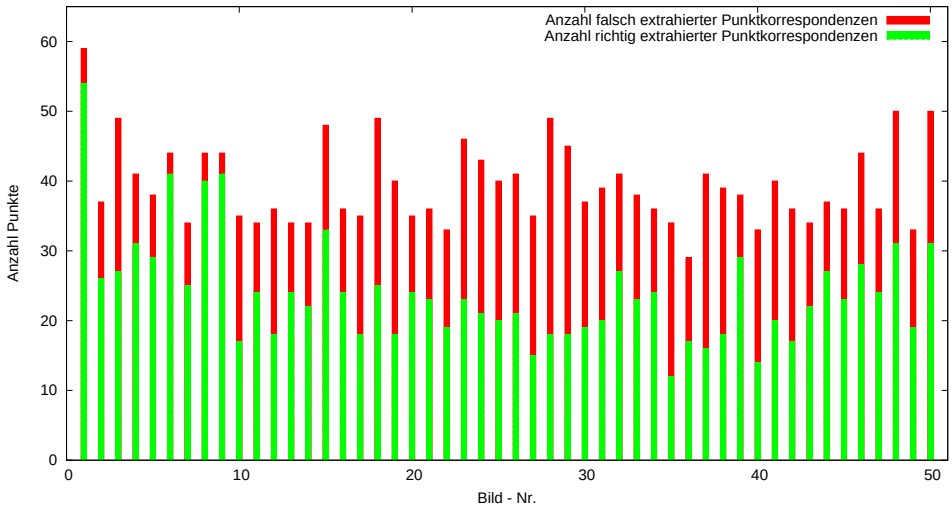

Bild 4.5: Gegenüberstellung der Anzahl von falsch und richtig extrahierten Punktkorrespondenzen für jedes Bild der Testsequenz.

für die 50 Kamerabilder die richtigen und falschen Punktkorrespondenzen gegenüber. Es wird deutlich, dass die punktbasierte Poseschätzung trotz der hohen Anzahl falscher Korrespondenzen gute Ergebnisse liefert. Die Graphiken lassen weiterhin keine durchgängige Proportionalität zwischen Qualität der Poseschätzung und Anzahl der richtigen Punktkorrespondenzen erkennen. Dies zeigt, dass auch die lokale Optimierung einen nicht unerheblichen Einfluss auf die geschätzte Pose hat.

Eine Auswertung der Korrespondenzen zwischen Modelllinien und Bildpunkten zeigt Abbildung 4.6. Für jedes der 50 Testbilder werden die richtigen und falschen Korrespondenzen gegenübergestellt. Offensichtlich liefert die Extraktion von Punkten auf kontrastreichen Linien eine sehr hohe Anzahl richtiger Korrespondenzen. Dies ist in erster Linie darauf zurückzuführen, dass Ausreißer mit dem verwendeten RANSAC-Ansatz unterdrückt wurden. Weiterhin wurden bereits offline die Linien aussortiert, für die der Algorithmus mit hoher Wahrscheinlichkeit falsche Korrespondenzen extrahiert. Die gezeigten Ergebnisse bestätigen, dass dieses Vorgehen sinnvoll ist.

Das Diagramm aus Abbildung 4.7 macht deutlich, dass ein Grund für die schlechten Er-

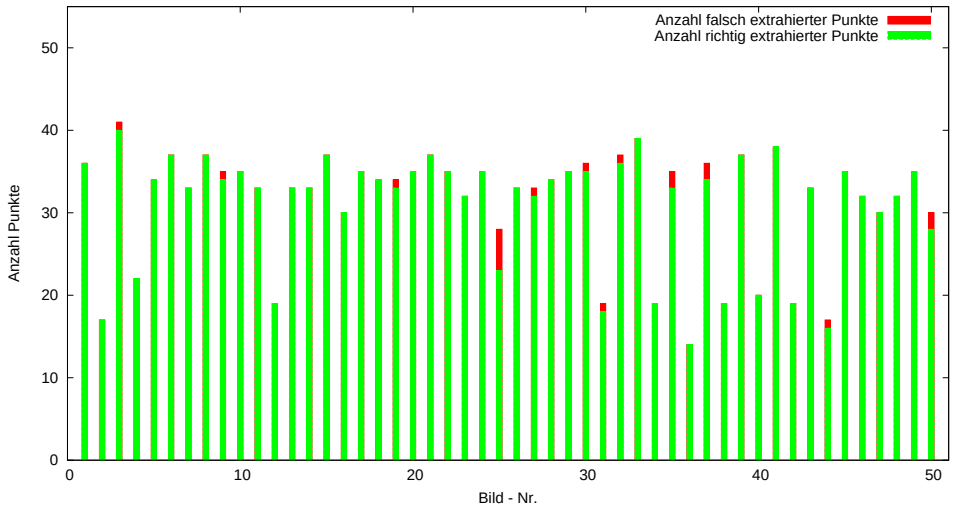

Bild 4.6: Gegenüberstellung der Anzahl von falsch und richtig extrahierten Punkten, die zu Modelllinien korrespondieren.

gebnisse der linienbasierten Objektverfolgung fehlerhafte Linienkorrespondenzen sind. Für die extrahierten Linien wurde die Hausdorff-Distanz zwischen extrahierter und manuell bestimmter Linie berechnet. Das Diagramm ordnet jedem Bild die Anzahl der betrachteten Linien und den durchschnittlichen Linienabstand zu. Aufgrund von Rauschen ist die manuelle Bestimmung der Linienendpunkte auch hier nur auf $\pm$ 3 Pixel genau. Es wird schnell deutlich, dass das Vorgehen nur in wenigen Fällen brauchbare Ergebnisse liefert. Offensichtlich muss zur Extraktion der Linienendpunkte ein höherer Aufwand betrieben werden.

Dass es sinnvoll sein kann, einen punktbasierten Ansatz um einen linienbasierten zu erweitern, zeigt das Liniendiagramm aus Abbildung 4.8. Das Diagramm ordnet jedem Kamerabild einen auf der Basis von Gleichung 4.1 berechneten Fehler zu. Die blaue Linie zeigt den Fehler der Poseschätzung, die auf manuell bestimmten Linienkorrespondenzen und der Hausdorff-Distanz basiert.

Erwartungsgemäß führen exakt bestimmte Linienendpunkte zu einer besseren Poseschätzung. Die Qualität der Poseschätzung ist jedoch auch eindeutig besser als die des RAPiD-

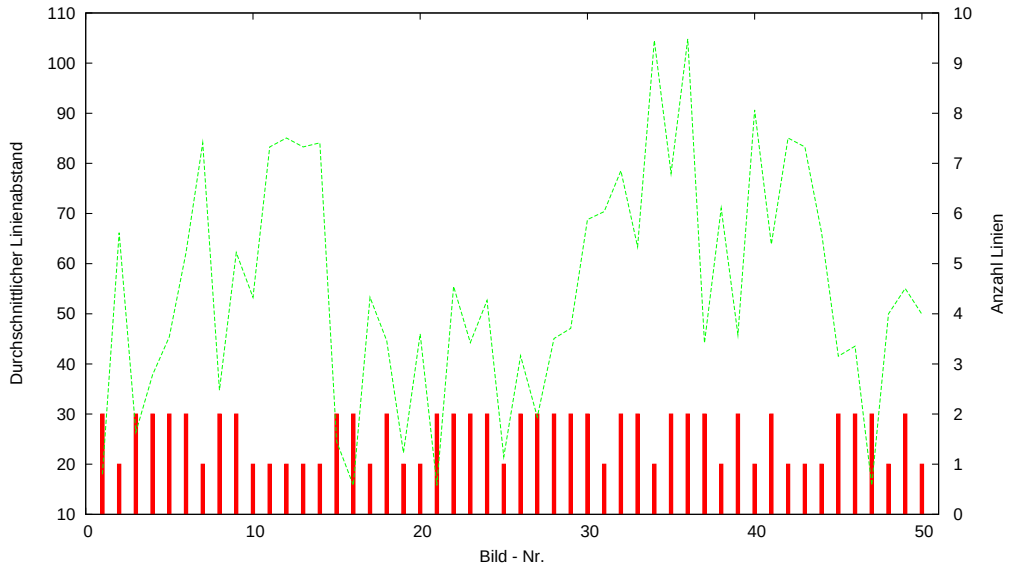

Bild 4.7: Das Liniendiagramm gibt für jedes der 50 Testbilder den durchschnittlichen Abstand zwischen extrahierten und manuell bestimmten Linien an. Das Säulendiagramm zeigt für jedes Bild die Anzahl der betrachteten Linien an.

Ansatzes, der auf der Basis des Diagramms aus Abbildung 4.3 als bester Ansatz hervorging. Es ist also durchaus erfolgversprechend, den punktbasierten Tracking-Ansatz um eine Verfolgung von Linien zu erweitern. Es ist jedoch zu überprüfen, ob eine robuste Bestimmung von Linienendpunkten mit angemessenem Aufwand geleistet werden kann.

Es ist weiterhin zu überprüfen, wie gut sich die vorgestellten Abstandsmaße für Linien für die Objektverfolgung eignen. Die Werte in Tabelle 4.1 geben für die Hausdorff-Distanz den kleinsten mittleren Fehler an. Eine abschließende Bewertung ist auf der Basis dieser Daten jedoch nicht möglich. Daher ist es interessant zu betrachten, ob sich signifikante Unterschiede ergeben, wenn die Schätzung auf korrekt extrahierten Linienendpunkten basiert. Das Diagramm aus Abbildung 4.9 bewertet die Poseschätzung für jedes Abstandsmaß und jedes der 50 Testbilder. Zur Bewertung wird wiederum das in Gleichung 4.1 definierte Fehlermaß verwendet. Eine Auflistung des durchschnittlichen Fehlers für die drei getesteten Abstandsmaße liefert die Tabelle 4.2.

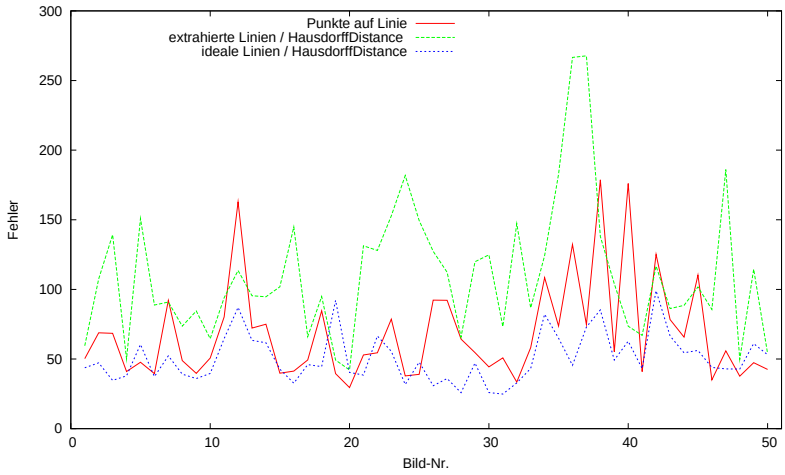

Bild 4.8: Für jedes Bild und jedes Verfahren wurde der bei der Poseschätzung gemachte Fehler nach Formel 4.1 bestimmt. Basiert die Poseschätzung auf manuell extrahierten Linien (blaue Linie), so ist sie meist exakter als die Schätzung des RAPiD-Ansatzes (rote Linie).

Augenscheinlich wird in diesem Fall die Abstandsberechnung der Klasse “MultiplePoints-Distance” als beste bewertet. Dieses Abstandsmaß scheint abhängiger als die anderen beiden von der korrekten Extraktion der Linienendpunkte zu sein. Da die Definition (siehe Gleichung 3.13) die Endpunkte explizit berücksichtigt, ist dies jedoch wenig überraschend. Letztlich muss das Abstandsmaß also abhängig von der Qualität der Linienextraktion gewählt werden.

4.2.2 Performanz

Zur Bewertung, wie performant das implementierte Programm Objekte verfolgt, wurde es an zwei Sequenzen getestet. Die Sequenzen bestehen jeweils aus 50 Grauwertbildern der Größe 960×1280 bzw. 480×640 Pixel. Um die Performanz einzelner Programmteile beurteilen zu können, wurden für diese einzelne Zeitmessungen vorgenommen. Bei den in Tabelle 4.3 zusammengestellten Zeiten handelt es sich um Mittelwerte der 50 getesteten

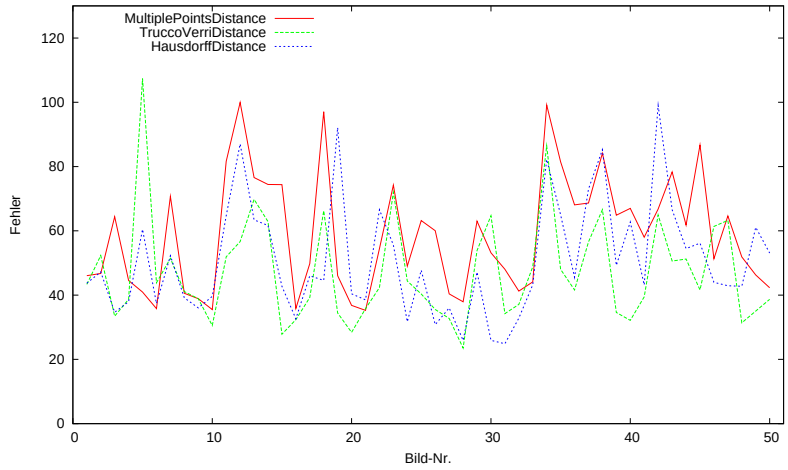

Bild 4.9: Vergleich des Fehlers der geschätzten Pose nach Formel 4.1 (in Pixel). Es wird die Qualität der linienbasierten Poseschätzung bei der Nutzung verschiedener Ähnlichkeitsmaße verglichen. Die Linienendpunkte wurden manell extrahiert.

Bilder. Die Zeiten wurden auf einem Mobilrechner (Intel Centrino 1,4 GHz) gemessen.

Es wird deutlich, dass der Aufwand für die Punktdetektion gerade bei großen Bildern sehr groß ist. Wenn es das Ziel ist, Objekte auf Grauwertbildern großer Auflösungen zu verfolgen, so sollte man hier ansetzten, um das Verfahren zu beschleunigen. Es wäre zu testen, inwieweit die Wahl eines anderen Punktdetektors den Algorithmus beschleunigen kann. Abhängig vom Vorwissen über die Position der interessanten Punkte könnte auch eine Beschränkung der Antwortberechnung auf eine Interessenregion das Verfahren beschleu-

Tracking - Ansatz	Durchschnittlicher Fehler
punkt- & linienbasiert (HausdorffDistance)	50.7369
punkt- & linienbasiert (TruccoVerriDistance)	58.8447
punkt- & linienbasiert (MultiplePointsDistance)	47.1967

Tabelle 4.2: Vergleich des durchschnittlichen Fehlers (in Pixel) bei Verwendung unterschiedlicher Ähnlichkeitsmaße und manuell extrahierten Linienendpunkten.

Auflösung	960×1280 Pixel	480×640 Pixel
Punktdetektion & Deskriptorberechnung	1.2102	0.3196
Deskriptor Matching	0.245	0.1816
Bestimmung der Punktkorrespondenzen	0.0636	0.0338
Bestimmung der Linienkorrespondenzen	0.1086	0.047
Optimierung (punktbasiert)	0.106	0.0892
Optimierung (punktbasiert & RAPiD)	0.169	0.1608
Optimierung (punkt- und linienbasiert)	0.0956	0.0716

Tabelle 4.3: Durchschnittlich benötigte Zeit (in Sekunden) für einzelne Schritte der Poseschätzung für Bilder verschiedener Auflösungen.

nigen.

Die für die Deskriptorberechnung benötigte Zeit ist proportional zur Größe der extrahierten Patches. Durch eine Verbesserung der offline durchgeführten Deskriptorberechnung (vgl. z. B. [LLF05]) könnte die Bestimmung der Punktkorrespondenzen bessere Ergebnisse liefern. Dies würde eine Reduktion der verwendeten Patchgröße von 11×11 Pixel erlauben und die Berechnung der Deskriptoren beschleunigen.

Die für die nicht-lineare Optimierung benötigte Zeit könnte beschleunigt werden, indem die Minimierung des Reprojektionsfehlers nicht mit sechs verschiedenen Startsimplizes wiederholt wird. Dies wäre sinnvoll, wenn es gelingen würde, die Anzahl der Ausreißer unter den Punktkorrespondenzen zu verringern. In dem Fall würde der Optimierungsalgorithmus nicht so häufig in lokalen Minima “hängenbleiben” und wäre somit unabhängiger von den Startparametern.

4.3 Zweite Testsequenz

Ziel der bisherigen Tests war es zu testen, ob es sinnvoll ist, eine punktbasierte Objektverfolgung um eine linienbasierte Objektverfolgung zu erweitern. Grundsätzlich ist es jedoch auch möglich, ein Objekt nur auf der Basis von Linien zu verfolgen. Um zu testen, wie gut

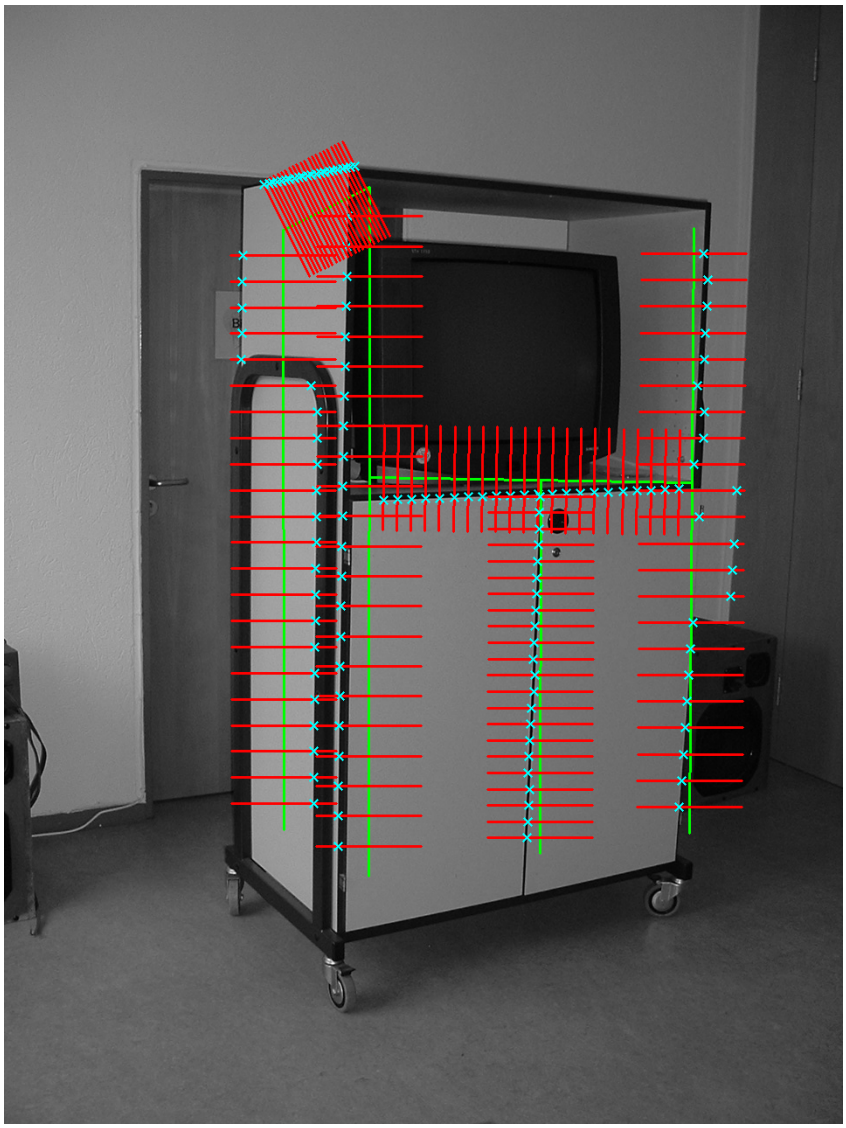

Bild 4.10: Linienextraktion für ein Bild der zweiten Sequenz. Die sechs grünen Linien wurden auf der Basis der letzten Pose in das Bild projiziert. Die Maxima auf den orthogonal verlaufenden, roten Suchlinien sind mit blauen Kreuzen gekennzeichnet. Von den extrahierten Punkten, die zu der im Bild am weitesten links liegenden Linie korrespondieren, liegen nur fünf auf der richtigen Kante.

dies mit dem implementierten Algorithmus gelingt, wurde ein Fernsehschrank über eine kurze Sequenz von Bildern verfolgt. Abbildung 4.10 zeigt, dass der Schrank eine Reihe von kontrastreichen geraden Kanten aufweist und sich somit als Modell für ein rein linienbasiertes Tracking eignet. Das Drahtgittermodell besteht aus 14 Linien, von denen 6 verfolgt werden. Da der RAPiD-basierte Ansatz zur Poseschätzung die besten Ergebnisse lieferte, wird er auch zur Verfolgung des Schranks genutzt. Die Bewertung der Poseschätzung erfolgt analog zu der aus Abschnitt 4.2.1 als Summe des Projektionsfehlers von vier Eckpunkten.

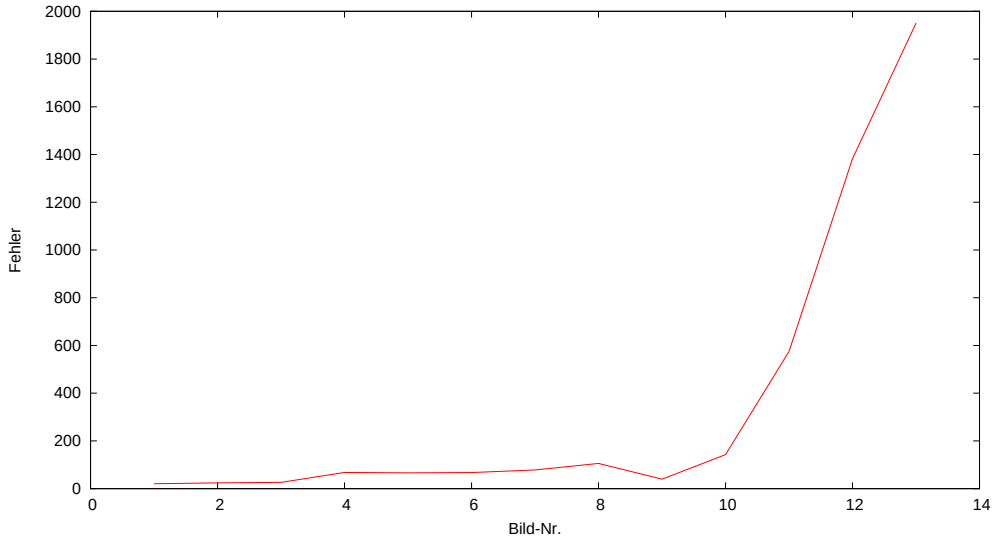

Bild 4.11: Entwicklung des Fehlers der geschätzten Pose für die zweite Testsequenz. Der Fehler ist in Pixel angegeben und wurde basierend auf der Formel 4.1 berechnet.

Das Liniendiagramm in Abbildung 4.11 zeigt die Entwicklung des Fehlers über eine kurze Sequenz von Bildern. Es wird deutlich, dass es über eine Reihe von Bildern sehr gut gelingt, das Objekt zu verfolgen. Sobald jedoch beim Schätzen der Lage einer Kante ein großer Fehler gemacht wird, ist die Qualität der Poseschätzung schlecht. Abbildung 4.10 zeigt, dass die Extraktion von Punkten, die zur linken Modelllinie korrespondieren, eine große Anzahl von falschen Korrespondenzen liefert. Durch die Anwendung des RANSAC-Ansatzes für diese Linie werden selbst die wenigen richtig extrahierten Punkte unter-

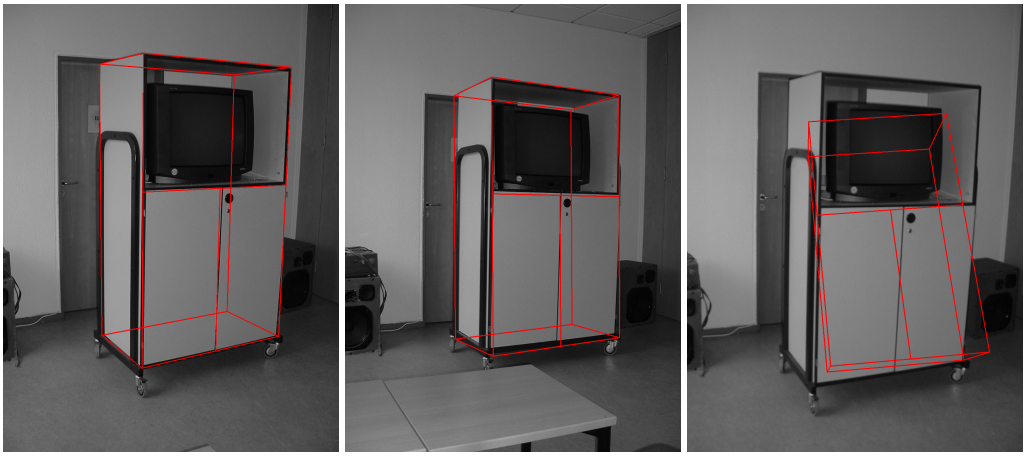

Bild 4.12: Drei Beispiele für die Poseschätzung für Bilder der zweiten Testsequenz.

drückt.

Verfolgt man ein Modell RAPiD-basiert auf der Basis von Linien, so ist die Qualität der Poseschätzung sehr abhängigig von der initialen Schätzung. Der implementierte Ansatz verwendet die Pose des letzten Kamerabildes als initiale Schätzung. Dies führt in der Regel dazu, dass der Algorithmus nach einer fehlerhaften Poseschätzung nicht mehr wie gewünscht funktioniert. Die schlechte Schätzung der Pose von Bild 10 (siehe Abbildung 4.11) führt dazu, dass auch die Schätzungen für die folgenden Bilder falsch sind. Abbildung 4.12 zeigt das Ergebnis der Poseberechnung für drei Beispielbilder der Sequenz. Die sehr gute Poseschätzung für das linke und mittlere Bild macht deutlich, dass eine rein linienbasierte Objektverfolgung funktionieren kann. Sie ist jedoch sehr fehleranfällig und für viele Modelle unbrauchbar.

Kapitel 5

Zusammenfassung & Ausblick

Im Rahmen dieser Diplomarbeit wurde eine Architektur zur markerlosen, modellbasierten Objektverfolgung entwickelt. Der vorgestellte Algorithmus basiert auf Referenzbildern und schätzt die Pose durch nicht-lineare Minimierung des Reprojektionsfehlers. Die Berechnung des zu minimierenden Fehlers basiert auf Punktkorrespondenzen, Korrespondenzen zwischen Linien und Bildpunkten oder Linienkorrespondenzen. Es wurden drei Ansätze gegenübergestellt, implementiert und evaluiert.

Es stellte sich heraus, dass es sinnvoll ist, den punktbasierten Ansatz zur Poseberechnung zu erweitern. Eine Poseschätzung, die neben Punktkorrespondenzen auch zu Modelllinien korrespondierende Punkte berücksichtigt, stellte sich als beste Lösung heraus. Eine Poseschätzung auf der Basis von Linienkorrespondenzen ist erfolgversprechend, aber stark von der Linienextraktion abhängig. Werden Linienkorrespondenzen zur Berechnung des Fehlers verwendet, so benötigt man ein Abstandsmaß zwischen Linien. Es wurden drei Abstandsmaße vorgestellt und in die Implementierung dieser Diplomarbeit integriert. Es zeigte sich, dass sich zur linienbasierten Objektverfolgung auch sehr einfach und schnell zu berechnende Abstandsmaße eignen.

Die im Rahmen dieser Diplomarbeit entwickelte Architektur wurde so konzipiert, dass sie leicht um zusätzliche Funktionalitäten erweitert werden kann. Die folgende Liste soll einen Überblick über mögliche Erweiterungen des implementierten Ansatzes geben:

1. Um das Feature Matching zu verbessern, könnten die Modellpunkte eines Referenzdatensatzes zusätzlich auf mehrere synthetisch erzeugte Deskriptoren abgebildet werden. In Abschnitt 2.5.2 wurde eine mögliche Implementierung von Lepetit [LLF05] vorgestellt.

2. Basiert die Poseschätzung nur auf Korrespondenzen zwischen dem aktuellen Kamerabild und einem Referenzbild, so kann es bei aufeinanderfolgenden Bildern zu sprunghaften Änderungen der geschätzten Pose kommen. Es wäre interessant zu überprüfen, welche Methoden sich zur Reduktion von Jitter eignen und wie diese sich auf die Geschwindigkeit des Verfahrens auswirken. Eine interessante Lösung des Problems wird z. B. in [LVTF03] vorgestellt.

3. Die in Kapitel 4 vorgestellten Testergebnisse haben gezeigt, dass die Qualität der Linienextraktion nicht gut genug ist, um eine gute Poseschätzung zu ermöglichen. Es wäre sinnvoll zu überprüfen, welche alternativen Verfahren zur Extraktion der Linienendpunkte existieren. Würde es gelingen, einen guten Kompromiss zwischen Qualität und Geschwindigkeit zu finden, würde dies den vorgestellten Algorithmus sicher aufwerten.

4. Im Rahmen dieser Diplomarbeit wurden stets gerade Linien betrachtet. Da viele 3-D-Modelle aus parametrischen Kurven bestehen, wäre es interessant, den vorgestellten Ansatz um eine Verfolgung von Splines zu erweitern. Würde man sich auf die Verfolgung von wenigen gut detektierbaren Kurven beschränken, so könnte dies auch performant gelöst werden.

Die entwickelte Architektur bietet somit noch viele Möglichkeiten einer Erweiterung des implementierten Ansatzes. Es wird interessant sein, zu beobachten, inwieweit es in Zukunft gelingen wird, die genannten Erweiterungen umzusetzen.

Anhang A

Kommandozeilenoptionen

Im Rahmen dieser Diplomarbeit wurden drei Programm entwickelt:

1. computeRefData
2. computePose
3. trackModel

Mit dem Programm “computeRefData” können Referenzdatensätze erstellt werden, die das Programm “trackModel” zum Verfolgen eines Objektes nutzt. Soll nur die Pose für ein einziges Bild berechnet werden, so kann dafür das Programm “computePose” genutzt werden. Die Programme werden über die Kommandozeile aufgerufen. Einen ersten Überblick über die wichtigsten Parameter geben die Skizzen aus Abbildung A.1 und Abbildung A.2.

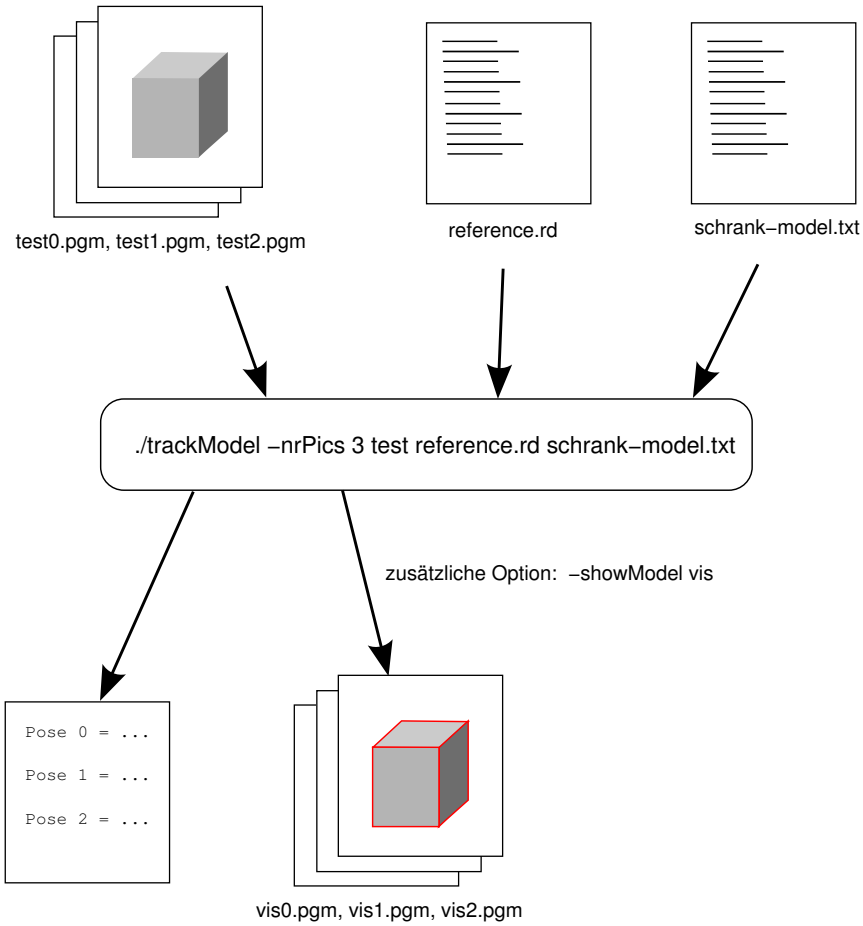

Bild A.1: Benutzung des Programms “trackModel”

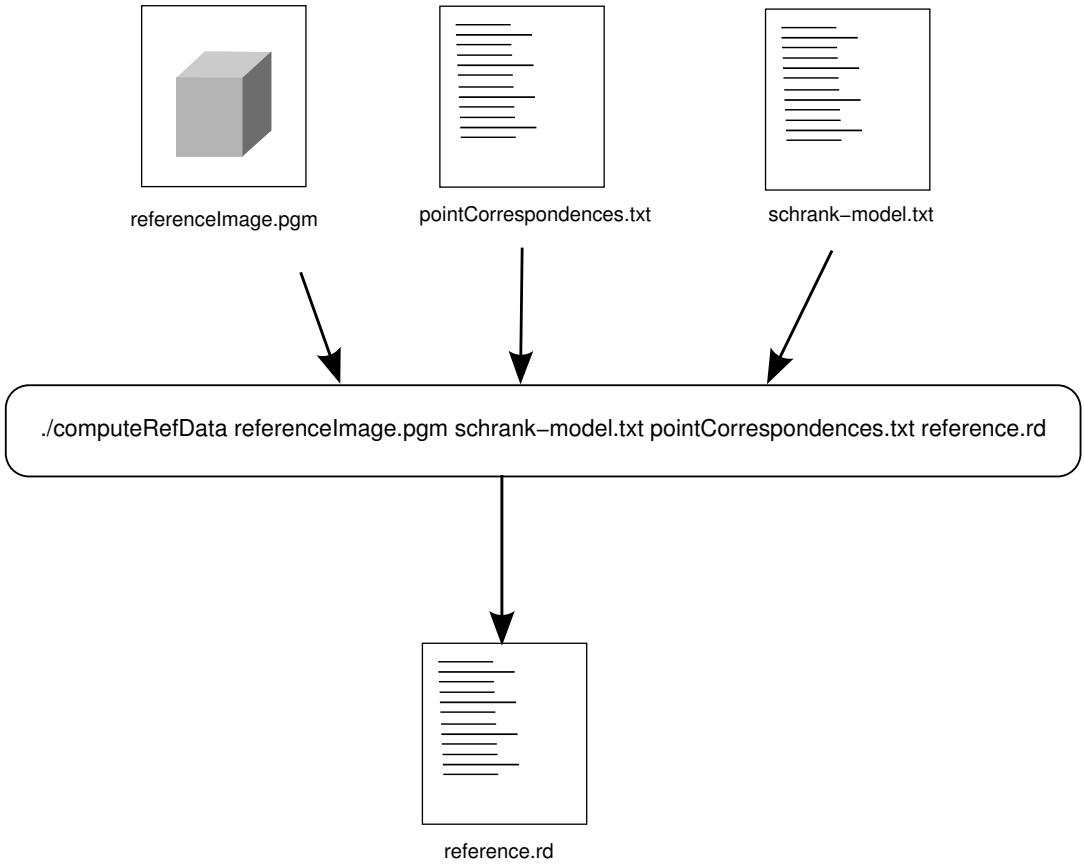

Bild A.2: Benutzung des Programms “computeRefData”

Um einen vollständigen Überblick über die Kommandozeilenoptionen und -parameter zu geben, folgt an dieser Stelle eine komplette Auflistung für die drei implementierten Programme. Man erhält sie ebenfalls beim Aufruf der Programme mit der “help”-Option.

Listing A.1: Kommandozeilenoptionen & -parameter “computeRefData”

```
Computes the reference data for a given gray level reference image.
In addition to the image the program needs a model of the object,
that has to be tracked. The reference data will be written to a file.
The filename has to be provided by the user.

If you don’t want to compute the point correspondences by a
backprojection on the model, you can provide them by a file.
Use the option ’explicitCorrs’ and a file that is formatted like this:

<pc1 x>;<pc1 x>;<wc1 x>;<wc1 y>;<wc1 z>
<pc2 x>;<pc2 x>;<wc2 x>;<wc2 y>;<wc2 z>
...
( pc: pixel coordinate / wc: world coordinate )
The ’printCorrs’ option prints all point correspondences to stdout.


Options:
   explicitCorrs(opt)(1)(string): read explicit point correspondences
             Default:
   showModel (opt)(1)(string): filename for the visualization of the pose
             Default:
   printCorrs(opt)(0)(bool)  : print point correspondences
   fatLines  (opt)(0)(bool)  : use fat lines to visualize the pose
   verbose   (opt)(0)(bool)  : output of runtime information
   help      (opt)(0)(bool)  : Help-Info

Arguments:
   argument 1   (obl)(argument): input gray level image file
   argument 2   (obl)(argument): text file containing the 3d model
   argument 3   (obl)(argument): text file containing the point correspondences
   argument 4   (obl)(argument): output reference data file
```

Listing A.2: Kommandozeilenoptionen & -parameter “computePose”

```
Given a reference data file and a gray level image the program
computes the pose and prints the result to ’stdout’. A reference data
file can be built by the program ’computeRefData’.

Possible line extraction strategies:
 0 = point-based
```

```
 1 = point-based & RAPiD
 2 = line-based

Possible line distance measures:
 0 = HausdorffDistance
 1 = TruccoVerriDistance
 2 = MultiplePointsDistance

The point correspondences can be visualized by the option
'showPointCorrs'. The reference image has to be provided:
'./computePose -showPointCorrs <reference image filename>'



Options:
   strategy  (opt)(1)(int)   : strategy for line extraction
             Range: (0,2)
             Default: 0
   distance  (opt)(1)(int)   : line distance measure
             Range: (0,2)
             Default: 1
   showPointCorrs(opt)(1)(string): show point correspondences
             Default:
   showLineExtraction(opt)(1)(string): filename for the visualization of the
             line extraction
             Default:
   showModel (opt)(1)(string): filename for the visualization of the pose
             Default:
   showPoints(opt)(1)(string): filename for the visualization of the interest points
             Default:
   timestat  (opt)(0)(bool)  : print time statistics
   fatLines  (opt)(0)(bool)  : use fat lines to visualize the pose
   verbose   (opt)(0)(bool)  : output of runtime information
   help      (opt)(0)(bool)  : Help-Info

Arguments:
   argument 1   (obl)(argument): name of the input gray level image file
   argument 2   (obl)(argument): corresponding reference data file
   argument 3   (obl)(argument): text file containing the 3d model
```

Listing A.3: Kommandozeilenoptionen & -parameter “trackModel”

```
Given a reference data file and an array of gray level images
the program computes the pose for each of these images and prints the
result to 'stdout'. A reference data file can be built by the program
'generateRefData'.

If the first argument is 'pic' the program computes the pose for
```

```
the images 'pic0.pgm', 'pic1.pgm', ...'pic<nrPics-1>.pgm'

Possible line extraction strategies:
 0 = point-based
 1 = point-based & RAPiD
 2 = line-based

Possible line distance measures:
 0 = HausdorffDistance
 1 = TruccoVerriDistance
 2 = MultiplePointsDistance

The point correspondences can be visualized by the option
'showPointCorrs'. The reference image has to be provided:
'./trackModel -showPointCorrs <reference image filename>'



Options:
   nrPics    (opt)(1)(int)   : number of input pictures
             Range: (1,1000)
             Default: 18
   strategy  (opt)(1)(int)   : strategy for line extraction
             Range: (0,2)
             Default: 0
   distance  (opt)(1)(int)   : line distance measure
             Range: (0,2)
             Default: 0
   showPointCorrs(opt)(1)(string): show point correspondences
             Default:
   showModel (opt)(1)(string): filename for the visualization of the pose
             Default:
   showLineExtraction(opt)(1)(string): filename for the visualization of the
             line extraction
             Default:
   showPoints(opt)(1)(string): filename for the visualization of the interest points
             Default:
   verbose   (opt)(0)(bool)  : output of runtime information
   fatLines  (opt)(0)(bool)  : use fat lines to visualize the pose
   evaluation(opt)(0)(bool)  : evaluation of the pose computation
   help      (opt)(0)(bool)  : Help-Info

Arguments:
   argument 1   (obl)(argument): name of the input gray level image files
   argument 2   (obl)(argument): corresponding reference data file
   argument 3   (obl)(argument): text file containing the 3d model
```

Literaturverzeichnis

[Ble04] BLESER, GABI: *Entwicklung eines 3D-markerlosen Kamera-Trackingverfahrens zur Echtzeit-Augmented-Reality-Bildsynthese*. Diplomarbeit, Universität Koblenz-Landau, www.uni-koblenz.de, 10 2004.

[Can86] CANNY, JOHN FRANCIS: *A computational approach to edge detection*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 8(6):679–698, 1986.

[DC02] DRUMMOND, TOM und ROBERTO CIPOLLA: *Real-Time Visual Tracking of Complex Structures*. IEEE Trans. Pattern Anal. Mach. Intell., 24(7):932 946, 2002.

[DD92] DEMENTHON, DANIEL und LARRY S. DAVIS: *Model-Based Object Pose in 25 Lines of Code*. In: *ECCV '92: Proceedings of the Second European Conference on Computer Vision*, Seite 335 343, London, UK, 1992. Springer-Verlag.

[FB81] FISCHLER, MARTIN A. und ROBERT C. BOLLES: *Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography.* Commun. ACM, 24(6):381–395, 1981.

[Fel05] FELDMANN, TOBIAS: *Kombination verschiedener Ähnlichkeitsmaße für die 2-D/3-D Registrierung von Röntgenbildern*. Diplomarbeit, Universität Koblenz-Landau, UniKO, www.uni-koblenz.de, 2005. Diplomarbeit teilweise extern betreut von Dr. Frank Deinzer bei Siemens Medical Solutions, Forchheim.

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

[Har96] HARBECK, MICHAEL: *Objektorientierte linienbasierte Segmentierung von Bildern*. Doktorarbeit, Friedrich-Alexander-Universität Erlangen-Nürnberg, Univ. Erlangen-Nürnberg, 12 1996.

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

[HZ03] HARTLEY, RICHARD I. und ANDREW ZISSERMAN: *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003.

[Int] INTEL: *Open Source Computer Vision Library (OpenCV)*.

[KB99] KATO, HIROKAZU und MARK BILLINGHURST: *Marker Tracking and HMD Calibration for a Video- based AR Conferencing System*. In: *2nd Int. Workshop on AR*, Seiten 85–94, 1999.

[LF05] LEPETIT, V. und P. FUA: *Monocular Model-Based 3D Tracking of Rigid Objects: A Survey*. Foundations and Trends in Computer Graphics and Vision, 1(1):1–89, 10 2005.

[LLF05] LEPETIT, VINCENT, PASCAL LAGGER und PASCAL FUA: *Randomized Trees for Real-Time Keypoint Recognition*. In: *Conference on Computer Vision and Pattern Recognition, San Diego, CA*, Seiten 775–781, 6 2005.

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

[LPF04] LEPETIT, VINCENT, JULIEN PILET und PASCAL FUA: *Point Matching as a Classification Problem for Fast and Robust Object Pose Estimation*. In: *Conference on Computer Vision and Pattern Recognition, Washington, DC*, 6 2004.

[LVTF03] LEPETIT, V., L. VACCHETTI, D. THALMANN und P. FUA: *Fully Automated and Stable Registration for Augmented Reality Applications*. In: *International Symposium on Mixed and Augmented Reality, Tokyo, Japan*, 2003.

[MC05] MARCHAND, ERIC und FRANCOIS CHAUMETTE: *Feature tracking for visual servoing purposes*. Robotics and Autonomous Systems, 52(1):53–70, 7 2005.

[MN65] MEAD, R. und J. A. NELDER: *A Simplex Method for Function Minimization*. The Computer Journal, TCJ, (7):308–313, 1965.

[MP00] MALCIU, MARIUS und FRANÇOISE J. PRÊTEUX: *A Robust Model-Based Approach for 3D Head Tracking in Video Sequences*. In: *FG '00: Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition 2000*, Seiten 169–175, 2000.

[MS05] MIKOLAJCZYK, KRYSTIAN und CORDELIA SCHMID: *A performance evaluation of local descriptors*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 27(10):1615–1630, 2005.

[Pau01] PAULUS, DIETRICH: *Aktives Bildverstehen*. Der Andere Verlag, Osnabrück, 2001.

[PH03] PAULUS, DIETRICH und JOACHIM HORNEGGER: *Applied pattern recognition: A practical introduction to image and speech processing in C++*. Advanced Studies in Computer Science. Vieweg, Braunschweig, 4 Auflage, 2003.

[RD06] REITMAYR, GERHARD und TOM DRUMMOND: *Going Out: Robust Model-based Tracking for Outdoor Augmented Reality*. In: *IEEE ISMAR'06*, 2006.

[SR00] SPIES, HAGEN und IAN RICKETTS: *Face Recognition in Fourier Space*. Vision Interface, Seiten 38–44, 2000.

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

[TP91] TURK, M. und A. PENTLAND: *Eigenfaces for Recognition*. J. Cognitive Neuroscience, J.Cog., 3(1), 1991.

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

[VLF04] VACCHETTI, LUCA, VINCENT LEPETIT und PASCAL FUA: *Combining Edge and Texture Information for Real-Time Accurate 3D Camera Tracking*. In: *International Symposium on Mixed and Augmented Reality, Arlington, VA*, 11 2004.

[Win94] WINZEN, ANDREAS: *Automatische Erzeugung dreidimensionaler Modelle für Bildanalysesysteme*. Doktorarbeit, Friedrich-Alexander-Universität Erlangen-Nürnberg, Univ. Erlangen-Nürnberg, Erlangen, 1994.

[WVS05] WUEST, HARALD, FLORENT VIAL und DIDIER STRICKER: *Adaptive Line Tracking with Multiple Hypotheses for Augmented Reality.* In: *ISMAR*, Seiten 62–69, Vienna, Austria, 10 2005. IEEE Computer Society.