UNIVERSITÄT
KOBLENZ · LANDAU
Fachbereich 4: Informatik

Markerloses Tracking unter Verwendung von Analyse durch Synthese auf Basis von Featuredetektoren

Diplomarbeit

zur Erlangung des Grades einer Diplom-Informatikerin
im Studiengang Computervisualistik

vorgelegt von
Sabine Achilles

Erstgutachter:	Prof. Dr. Stefan Müller Institut für Computervisualistik, Universität Koblenz-Landau
Zweitgutachter:	Dipl.-Inf. Timo Dickscheid Institut für Photogrammetrie, Universität Bonn

Koblenz, im Juli 2008

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

	Ja	Nein
Mit der Einstellung der Arbeit in die Bibliothek bin ich einverstanden.	☐	☐
Der Veröffentlichung dieser Arbeit im Internet stimme ich zu.	☐	☐

. . . . . .

(Ort, Datum) (Unterschrift)

Danksagung

Ich bedanke mich bei allen, die mir bei der Diplomarbeit und über das gesamte Studium hinweg geholfen haben. Zuerst einen herzlichen Dank an Prof. Müller für das interessante Thema sowie die Ratschläge und Denkanstöße während der Anfertigung dieser Diplomarbeit. Des Weiteren möchte ich mich sehr bei Steffi und Rosa, den besten Korrekturleserinnen der Welt bedanken. Auch wenn es für euch schwer war, eine Diplomarbeit im Bereich der Informatik zu lesen, waren eure Tipps Gold wert. Vielen Dank an Leif für die langen Diskussionen über Tracking, C++ und BV. Auch wenn du selber gerade an deiner Diplomarbeit saßt, konnte ich dich immer mit Fragen löchern. Danke an Heinz, der mir ein weiteres Semester ermöglicht hat. Zu guter Letzt natürlich danke, danke, danke an meine Eltern für die Unterstützung während des Studiums. Ohne euch hätte ich das alles nicht geschafft.

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Motivation und Zielsetzung

Augmented Reality (AR) ist ein Anwendungsbereich der Computergraphik, in dem eine reale Szene mit virtuellen, computergenerierten Daten kombiniert wird. Sie findet unter anderem Anwendung in der Konstruktion, der Montage und Wartung und der Medizin. Im Gegensatz zur virtuellen Realität (VR), kann sich der Anwender bzw. die Anwenderin hierbei in der realen Welt bewegen und wird durch die Einbindung/Überlagerung virtueller Informationen (z. B. Text, Graphiken oder Baupläne) in seiner bzw. ihrer Arbeit unterstützt. Um diese Informationen auf die richtige Position in der realen Szene zu projizieren, ist es wichtig die extrinsischen Kameraparameter (Kameraposition und -orientierung) jederzeit genau zu kennen. Diese Aufgabe übernimmt ein Trackingsystem, das anhand bestimmter Merkmale in der Szene die Kamerapose zurückrechnen kann.

Es existieren viele verschiedene Trackingverfahren, die sich in zwei Gruppen unterteilen lassen: *Grobtrackingsysteme* (GPS, Infrarot-Baken, W-Lan) und *Feintrackingsysteme* (Ultraschall, elektromagnetische und vor allem optische Systeme). Grobtrackingsysteme können die Position der Kamera bis auf wenige Meter genau bestimmen und sind damit ausreichend präzise für beispielsweise die Standortermittlung von Fahrzeugen oder Personen zur Navigationsunterstützung. Für Anwendungen in der AR sind jedoch eine weitaus höhere Genauigkeit und Angaben zur Orientierung der Kamera von Nöten, weshalb hierfür Trackingverfahren verwendet werden, die der Gruppe der Feintrackingsysteme zuzuordnen sind. Oft wird zur Initialisierung eines Trackingsystems ein Grobtracking durchgeführt, mit dem die un-

gefähre Position ermittelt werden kann, um die Pose danach mit Hilfe von Feintrackingverfahren zu verfeinern.

Vor allem optische Trackingsysteme haben sich beim Feintracking aufgrund ihrer hohen Genauigkeit hervorgetan. Hierbei wird unterschieden zwischen „inside-out“ und „outside-in“ sowie „markerbasiert“ und „markerlos“. Das „inside-out“-Tracking zeichnet sich dadurch aus, dass die Kamera am Körper des Anwenders befestigt ist und so die Umgebung getrackt wird, wohingegen die Kamera beim „outside-in“-Tracking in der Umgebung positioniert ist und der Anwender getrackt wird. Diese Arten können beide jeweils mit oder ohne Marker arbeiten. Während für markerbasierte Verfahren schon sehr ausgereifte Systeme existieren (z. B. AR-Toolkit [ART]), wird das markerlose Tracking noch immer als ungelöstes Problem bezeichnet.

Marker sind künstlich in die Umgebung integrierte, unverwechselbare Symbole, die durch einfache Bildverarbeitungstechniken identifiziert werden können. Es muss hierfür also ein Eingriff in die Umgebung vorgenommen werden, der einen hohen Aufwand vor dem tatsächlichen Tracking zur Folge hat. Ein weiterer Nachteil ist, dass die Marker leicht von anderen Objekten verdeckt werden können, womit eine Identifizierung unmöglich gemacht wird. Anders ist dies beim markerlosen Tracking, da dort lediglich natürliche Merkmale der Umgebung benutzt werden, um die extrinsischen Kameraparameter zu bestimmen. Diese natürlichen Merkmale können entweder einzelne, auffällige Merkmale (Features) oder aber auch der gesamte Bildinhalt sein.

Diese Diplomarbeit versucht die Frage zu beantworten, wie die Methoden der Computergraphik im Kontext eines optischen Trackingsystems die Methoden der Bildverarbeitung verbessern bzw. unterstützen können. Die meisten bisher vorgestellten Trackingverfahren arbeiten auf einer Frame-to-Frame-Basis oder verwenden lediglich Wireframemodelle zur Unterstützung der Poseberechnung. In dieser Diplomarbeit werden dagegen zur Bestimmung der Kamerapose bestimmte Features einerseits aus dem Kamerabild und andererseits aus einer mit OpenGL gerenderten 3D-Szene verwendet. Die Verwendung eines 3D-Modells zur Bestimmung der Kamerapose wird *Analyse durch Synthese* (siehe Kapitel 2.1) genannt und ist im stark computergraphikgestützten Trackingkontext ein relativ neuer Ansatz, der in dieser Diplomarbeit weiter untersucht werden soll.

Ziel dieser Diplomarbeit ist somit, zu untersuchen, welche Featuredetektoren geeignet sind möglichst dieselben Merkmale im gerenderten und im Referenzbild der Kamera zu detektieren und welche Deskriptoren in diesem Zusammenhang die besten Ergebnisse erzielen. Außerdem ist zu testen, wie detailliert die Szene zu rendern ist,

wie genau die Texturen dargestellt sein sollten und wie wichtig der Lichteinfall für ein genaues Tracking ist.

1.2 Aufbau der Arbeit

Die Diplomarbeit ist wie folgt gegliedert: In Kapitel 2 werden die Grundlagen für ein markerloses Trackingsystem beschrieben. Daraufhin wird in Kapitel 3 zunächst ein Überblick über bestehende Trackingverfahren gegeben, gefolgt von einer Einführung in bestehende Trackingsysteme und der Beschreibung der praktischen Umsetzung des im Rahmen dieser Diplomarbeit erstellten Trackingsystems in Kapitel 3. Testergebnisse im Hinblick auf die Präzision der einzelnen Featuredetektoren unter sich ändernden Bedingungen werden in Kapitel 4 vorgestellt. Abschließend werden in Kapitel 5 die Ergebnisse der vorliegenden Diplomarbeit zusammengefasst, sowie einen Ausblick über Verbesserungen und Schwerpunkte für zukünftige Forschungsarbeiten gegeben.

Kapitel 2

Grundlagen

In diesem Kapitel sollen die Grundlagen des Trackingsystems vorgestellt werden. Das in dieser Diplomarbeit entwickelte System basiert auf der Erkennung von *interessanten Merkmalen* (Features) im Kamerabild sowie in einem auf die Bildebene projizierten 3D-Modell derselben Szene (siehe Abschnitt „Punktdetektoren“ 2.2). Die unterstützende Verwendung eines 3D-Modells wird als *Analyse durch Synthese* bezeichnet. Eine detaillierte Erklärung des Begriffs folgt in Abschnitt 2.1. Die in beiden Bildern gefundenen Merkmale werden in einem weiteren Schritt miteinander verglichen und zu Merkmalspaaren zusammengefasst (siehe Abschnitt „Korrespondenzproblem“ 2.4). Abschließend wird aus diesen Merkmalspaaren die Pose der Kamera geschätzt (siehe Abschnitt „Poseschätzung“ 2.7).

2.1 Analyse durch Synthese

Die *Analyse durch Synthese* hat ihren Ursprung in der Spracherkennung und kann als ein Prozess beschrieben werden, in dem Hypothesen aufgestellt und mit bestimmten Eingangsdaten verglichen werden bis eine Übereinstimmung gefunden ist. Eine der ersten Beschreibungen dieses Begriffs wurde 1961 durch Bell et al vorgenommen, die in [BFH+61] die Vorteile der Verwendung von synthetischen akustischen Signalen zur Analyse von Sprache beschrieben.

Nach dem „Analyse-durch-Synthese-Modell“ analysieren Menschen Sprache, indem sie die wahrgenommenen Worte immer wieder intern mit Lautfolgen vergleichen, bis eine wahrscheinliche Übereinstimmung auftritt. Nun wird die Erkenntnis darüber, dass bestimmte Parameter existieren, die die Entstehung von Lauten ein-

deutig beschreiben, zur Analyse in der Spracherkennung eingesetzt. Das Analyse-durch-Synthese-Prinzip wird in der Sprachverarbeitung umgesetzt, indem aus diesen Parametern eine Syntheseeinheit erstellt wird, die im Analyseschritt mit dem realen Sprachsignal verglichen wird. Der berechnete Fehler zwischen synthetischem und realem Signal wird minimiert, bis ein gewisser Schwellwert erreicht ist und somit die wahrscheinlichste Übereinstimmung festgestellt wurde.

Gegeben sei also ein synthetisches Modell $M(x)$ mit dem Parametervektor x, der den Zustand des Modells steuern kann. Liegt nun ein Eingangssignal $S(y)$ mit dem zu bestimmenden Vektor y vor, dann kann der Parametervektor x so geschätzt werden, dass gilt:

$$M(x) = S(y) \tag{2.1}$$

Die Schätzung für x entspricht somit dem Ergebnis für y.

Dieses Prinzip kann auch auf visuelle Probleme übertragen werden. Erste Ansätze wurden 1978 in [MN78] und 1979 in [BS79] verfolgt, indem Modelle benutzt wurden, um Menschen in Bildern zu erkennen bzw. deren Bewegungen in Videosequenzen zu verfolgen. Das benutzte Modell beschränkte sich jedoch größtenteils auf statische Zylinderdarstellungen der einzelnen Gliedmaßen eines Körpers. In [Moe99] wird angegeben, dass sich diese Art der Analyse durch Synthese als problematisch herausgestellt hat, da die Unbeweglichkeit der Modelle eine zu geringe Genauigkeit liefert. Aus diesem Grund wurden in nachfolgenden Arbeiten erfolgsversprechendere, dynamische Modelle verwendet.

Im Zusammenhang mit einer Schätzung der Kamerapose in einem Trackingsystem stellt ein gerendertes 3D-Modell der Szene die Syntheseeinheit dar. Die Eingangssignale sind die einzelnen Frames einer Videosequenz, und die Parametervektoren entsprechen der Pose der Kamera. Es wird nun versucht die Kamerapose des gerenderten 3D-Modells so zu schätzen, dass sie mit der Pose des realen Frames möglichst genau übereinstimmt. Eine genauere Beschreibung der praktischen Umsetzung der *Analyse durch Synthese* in dem zu dieser Diplomarbeit entwickelten Trackingsystem folgt in Kapitel 3.2.3.

2.2 Punktdetektoren

Interessante Merkmale in einem Bild können z. B. Punkte (Ecken), Linien oder komplexere geometrische Formen sein. In der Bildverarbeitung haben sich vor allem *Punktdetektoren* durchgesetzt, da diese keine Segmentierung erfordern und trotzdem ein robustes Ergebnis erzielen. Punktdetektoren (engl. „Corner Detector“) finden interessante Merkmale vor allem in Regionen in denen sich zwei Kanten schneiden. Prinzipiell finden sie jedoch Merkmal in Bereichen, in denen eine große Intensitätsveränderung auftritt, also auch an Geradenenden oder stark gekrümmten Kurven.

Um die Qualität eines Punktdetektors zu beschreiben, existieren vier Bewertungskriterien:

- Die *Detektionsrate* ist das Maß für die Anzahl der richtig gefundenen Merkmale.
- Die *Wiederholbarkeitsrate* beschreibt die Stabilität der Ergebnisse unter variierenden Transformationsbedingungen und Rauschen.
- Die *Lokalisationsgenauigkeit* ist das Maß für die Genauigkeit der gefundenen Merkmale, also ob der gefundene Punkt auch genau auf dem tatsächlich interessanten Merkmal liegt.
- Der *Rechenaufwand* beschreibt die Zeit die der Algorithmus braucht um die Merkmale im Bild zu finden.

2.2.1 Moravec-Operator

Der 1977 von Hans Moravec in [Mor77] beschriebene „Moravec-Operator“ findet Merkmale an Stellen, an denen das Minimum der Summe der quadratischen Differenzen zwischen benachbarten Pixeln in den vier Hauptrichtungen ein lokales Maximum darstellen. Es werden also Punkte mit geringer Selbstähnlichkeit gefunden, indem getestet wird wie ähnlich ein auf ein Pixel gelegtes Patch einem anderen, etwas verschobenen Patch ist. Die resultierende Intensitätsveränderung V wird dabei wie folgt berechnet:

$$V_1 = \sum_{\mu=0}^{k} \sum_{\nu=0}^{k} (I_{\mu,\nu} - I_{\mu,\nu+1}) \tag{2.2}$$

$$V_2 = \sum_{\mu=0}^{k} \sum_{\nu=0}^{k} (I_{\mu,\nu} - I_{\mu+1,\nu}) \tag{2.3}$$

$$V_3 = \sum_{\mu=0}^{k} \sum_{\nu=0}^{k} (I_{\mu,\nu} - I_{\mu+1,\nu+1}) \tag{2.4}$$

$$V_4 = \sum_{\mu=0}^{k} \sum_{\nu=0}^{k} (I_{\mu+1,\nu} - I_{\mu+1,\nu+1}) \tag{2.5}$$

$$V = min\{V_1, V_2, V_3, V_4\} \tag{2.6}$$

Der Moravec-Operator war einer der ersten Featuredetektoren die entwickelt wurden und dient als Grundlage für viele später entstandene Detektoren (Harris, Förstner, Kanade-Lucas-Tomasi). Seine eher durchschnittliche Detektions- und Wiederholbarkeitsrate sowie die hohe Rechenzeit führten dazu, dass er nur noch selten zur Merkmalsextraktion benutzt wird.

2.2.2 Harris Corner Detector

Der 1988 von Chris Harris und Mike Stephens entwickelte „Harris Corner Detector“ [HS88] (oft auch als „Plessley-Punkt-Detektor“ bezeichnet) ist eine Verbesserung des Moravec-Operators, indem die Verschiebungen in die Hauptrichtungen durch eine Autokorrelationsmatrix berechnet wird. Eine Autokorrelation ist im Allgemeinen die Kreuzkorrelation eines Signals mit sich selbst. Im Fall der hier betrachteten Punktdetektoren sind diese Signale die Ableitungen eines Bildes in x- und y-Richtung. Sei also:

$$v = \begin{pmatrix} g_x(i,j) \\ g_y(i,j) \end{pmatrix} \tag{2.7}$$

der Vektor mit den Ableitungen an der Stelle i, j im Bild, dann ist

$$A = vv^T = \begin{bmatrix} g_x^2 & g_x g_y \\ g_x g_y & g_y^2 \end{bmatrix} \tag{2.8}$$

die zu dem Punkt gehörende Autokorrelationsmatrix.

Da jedoch nicht nur ein Punkt sondern eine Punktnachbarschaft untersucht werden soll, wird die Autokorrelationsmatrix A aus den Summen der Ableitungen über eine $n \times n$ Nachbarschaft des Bildes in x- und y-Richtung berechnet:

$$A = \begin{bmatrix} \sum_{i,j \in \Omega} g_x(i,j)^2 & \sum_{i,j \in \Omega} g_x(i,j) \cdot g_y(i,j) \\ \sum_{i,j \in \Omega} g_x(i,j) \cdot g_y(i,j) & \sum_{i,j \in \Omega} g_y(i,j)^2 \end{bmatrix} \tag{2.9}$$

Je nach Begebenheit der Nachbarschaft des zu untersuchenden Punktes unterscheidet sich der Rang von A:

- Rang 2: interessanter Punkt
- Rang 1: gerade Kante
- Rang 0: homogene Region

Die Stärke („cornerness measure“) des gefundenen Merkmals wird durch die Eigenwerte von A ausgedrückt. Da die Berechnung der Eigenwerte jedoch sehr rechenaufwändig ist, wird Wert für die Merkmalsstärke V approximiert (der Koeffizient κ ist ein empirisch ermittelter Wert und liegt zwischen 0,04 und 0,15):

$$V = det(A) - \kappa \cdot spur(A)^2 \tag{2.10}$$

Durch die Verwendung der Autokorrelationsmatrix konnte die Lokalisationsgenauigkeit der Merkmalsextraktion gegenüber dem Moravec-Operator gesteigert werden und bewegt sich in einem Bereich von 1 - 2 Pixeln. Wird außerdem eine isotropische Gradientenberechnung benutzt, also eine Berechnung der Bildableitungen in viele Richtungen, ist auch eine Verbesserung der Wiederholbarkeitsrate für affine Transformationen zu beobachten, da somit auch Kanten erkannt werden, die nicht entlang einer der Hauptrichtungen verlaufen. Der Rechenaufwand ist jedoch, aufgrund

der umfangreicheren Berechnungen für V, um einiges höher als der des Moravec-Operators.

2.2.3 Förstner-Operator

Der 1987 von Wolfgang Förstner und Eberhard Gülch entwickelte „Förstner-Operator“ [FG87] verwendet, wie der Harris Corner Detector, die Autokorrelationsmatrix zur Berechnung der Intensitätsunterschiede bei einer Verschiebung in die Hauptrichtungen. Der Unterschied ist nun, dass der Förstner-Operator die um das Merkmal liegende Fehlerellipse untersucht und daraus Schlüsse über die Interessantheit des Punktes zieht. Um den Grad der Interessantheit zu berechnen, werden zwei Kriterien angewandt:

1. Die Fehlerellipse sollte möglichst rund sein

2. Die Fehlerellipse sollte möglichst klein sein

Anhand der Eigenwerte μ_1 und μ_2 (mit $\mu_1 > \mu_2$) der invertierten Autokorrelationsmatrix A^{-1} kann die Form und Größe der Ellipse bestimmt werden. Da diese Berechnung jedoch sehr rechenintensiv ist, werden zur Bestimmung der Rundheit q und der Größe w vereinfachte Formeln benutzt:

$$w = \frac{1}{\mu_1 + \mu_2} = \frac{1}{spur(A^{-1})} = \frac{det(A)}{spur(A)} \tag{2.11}$$

$$q = 1 - \left(\frac{\mu_1 - \mu_2}{\mu_1 + \mu_2}\right)^2 = \frac{4 \cdot det(A)}{spur(A)^2} \tag{2.12}$$

Das Maß für die Rundheit q liegt dabei zwischen 0 und 1. Ist $q = 1$, dann ist die Fehlerellipse ein Kreis, denn es liegen identische Eigenwerte vor. Ist $q = 0$, so muss mindestens einer der Eigenwerte gleich 0 sein und es liegt eine gerade Kante vor. Von den Autoren wird ein Schwellwert q_{lim} zwischen 0,5 und 0,75 vorgeschlagen.

Der Schwellwert w_{lim} wird benutzt, um homogene Regionen auszusortieren und sollte abhängig vom gesamten Inhalt des Bildes berechnet werden. Dazu gibt es zwei Möglichkeiten:

1. Der Schwellwert w_{lim} wird in Beziehung zum durchschnittlichen Gewicht w_{mean} gesetzt:

   - $w_{lim} = f \cdot w_{mean}$, mit $0,5 \leq f \leq 1,5$
   - Der Nachteil hierbei ist die Abhängigkeit des Gewichts von der Kantenschärfe, dem Kanteninhalt und dem Rauschlevel im Bild.

2. Der Schwellwert w_{lim} wird in Beziehung zum durchschnittlichen Gewicht in den homogenen Regionen gesetzt. Somit wird eine bessere Berechnung des mittleren Gewichts der Median w_{med} aller Gewichte w des Bildes erzielt:

   - $w_{lim} = c \cdot w_{med}$, mit $c = 5$ als gutem Erfahrungswert
   - Dieser Ansatz ist verlässlicher, jedoch auch durch die Berechnung des Medians etwas rechenaufwändiger.

Um eine Häufung von potentiellen Merkmalen zu vermeiden, wird von den Autoren in [FG87] vorgeschlagen diese nach der Berechnung jeweils in einem ersten Schritt spiralförmig in einer 3×3 - Nachbarschaft mit jedem Pixel zu vergleichen. Wird ein Pixel gefunden, das einen höheren Wert besitzt, wird das Merkmalspixel auf 0 gesetzt und das Fenster auf das nächste Merkmal verschoben. In einem zweiten Schritt wird die Berechnung auf den verbleibenden Merkmalen mittels eines 5×5 Pixel großen Fensters wiederholt. Die Unterteilung in zwei Schritte reduziert laut Förstner und Gülch den Rechenaufwand dieser Aufgabe.

Im Gegensatz zum Harris Corner Detector konnte die Lokalisationsgenauigkeit noch weiter gesteigert werden. Die Wiederholbarkeitsrate ist für affine Transformationen sehr gut und der Detektor ist wesentlich robuster gegenüber Rauschen. Der Rechenaufwand ist jedoch etwas höher als der des Harris Corner Detectors.

2.2.4 Kanade-Lucas-Tomasi-Detector

Der 1994 von Jianbo Shi und Carlo Tomasi entwickelte „Kanade-Lucas-Tomasi Detector“ (KLT) [ST94] basiert auf dem Harris Corner Detector (siehe 2.2.2). Der Unterschied zwischen den beiden Verfahren besteht darin, dass nicht die Formel 2.10 zur Berechnung der „Cornerness“ benutzt wird, sondern die Stärke eines Merkmals durch das Minimum der Eigenwerte $min(\lambda_1, \lambda_2)$ der Autokorrelationsmatrix A (siehe 2.9) ausdrückt wird.

2.2.5 Wang and Brady Corner Detector

Der 1995 von Han Wang und Michael Brady entwickelte „Wang and Brady Corner Detector“ [WB95a] sieht ein Bild nicht als einzelne Intensitätswerte, sondern als eine Fläche an. Ein interessanter Punkt ist dann gegeben, wenn eine große Krümmung an einer Kante festzustellen ist, die Kante also eine schnelle Richtungsänderung vorweist. Die Autoren haben festgestellt, dass sich die totale Krümmung eines Grauwertbildes proportional zur zweiten Richtungsableitung entlang der Kantennormalen und inversproportional zur Kantenstärke verhält. Die „Cornerness“ C wird definiert durch:

$$C = \nabla^2 I - c|\nabla I|^2 \tag{2.13}$$

Die Detektions- und Lokalisationsrate dieses Detektors ist vergleichbar mit der von Harris und Förstner, wohingegen die Wiederholbarkeitsrate zwischen denen von Harris und Moravec liegt. Der Rechenaufwand ist dem von Moravec ähnlich.

2.2.6 SUSAN

Der 1997 von Stephen Smith und Michael Brady entwickelte Detektor „SUSAN“ (Smallest Univalue Segment Assimilating Nucleus) [SB97] verwendet eine Kreismaske R, die auf den zu untersuchenden Pixel (Nukleus) gelegt wird. Diese Maske hat meistens einen Radius von 3,4 Pixeln, was einer Gesamtanzahl von 37 Pixeln in der Maske entspricht. Die Intensität jedes Pixels $r \in R$ innerhalb dieser Maske wurde mit dem Nukleus r_0 anhand eines Schwellwerts für die Differenz der Intensitäten t ursprünglich wie folgt verglichen:

$$c(r, r_0) = \begin{cases} 1, \text{ wenn } |I(r) - I(r_0)| \leq t \\ 0, \text{ wenn } |I(r) - I(r_0)| > t \end{cases} \tag{2.14}$$

Mit Hilfe dieses Vergleichs entsteht ein Bereich innerhalb der Maske, der Pixel beinhaltet, die eine dem Nukleus ähnliche Intensität besitzen. Diese Region wird USAN (Univalue Segment Assimilating Nucleus) genannt (siehe Abbildung 2.1). Sie hat einen großen Wert, wenn der Bereich um den Nukleus homogen ist und ist am kleins-

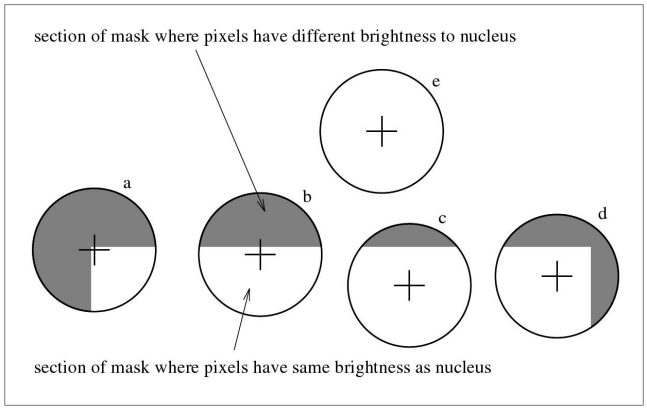


Abbildung 2.1: Vier Kreismasken des SUSAN-Detektors, wobei die weißen Regionen die USANs darstellen (aus [SB97]).

ten wenn dort eine Ecke vorliegt. Die Ergebnisse für jedes in der Maske liegende Pixel werden zu einem ersten „cornerness“ Wert aufsummiert:

$$n(r_0) = \sum_r c(r, r_0) \tag{2.15}$$

Die Summe n ist hierbei die Gesamtanzahl der Pixel in der USAN, die später minimiert wird.

Da die Vergleichsfunktion 2.14 eine zu „harte“ Entscheidung über die Ähnlichkeit der Pixel trifft, haben die Autoren die Funktion so abgeändert, dass sie „weicher“ ist und damit auch kleine Intensitätsunterschiede akzeptiert, ohne das Ergebnis für c zu beeinflussen (der Exponent der Größe 6 ist hierbei empirisch ermittelt worden [SB97]):

$$c(r) = e^{\frac{(I(r)-I(r_0))^6}{t}} \tag{2.16}$$

Der nächste Schritt ist der Vergleich von $n(r_0)$ mit dem geometrischen Schwellwert $g = \frac{n_{max}}{2}$, wobei n_{max} der höchstmögliche Wert von n ist. Berechnet wird dieses erste Ergebnis wie folgt:

$$R(r_0) = \begin{cases} g - n(r_0), & \text{wenn } n(r_0) < g \\ 0, & \text{sonst} \end{cases} \tag{2.17}$$

Je kleiner also die USAN ist, desto größer ist der Wert für $R(r_0)$. Aufgrund von falsch-positiven Ergebnissen, die bei SUSAN unter bestimmten Umständen auftreten können (z. B. bei verwaschenen Kanten direkt neben der tatsächlichen Kante), wird das initial gewonnene Ergebnis weiter überarbeitet. Um diese „false positives“ zu eliminieren, wird der Abstand zwischen dem Nukleus und dem Gravitationszentrum der USAN berechnet. Ist dieser Abstand klein, dann wird der Treffer verworfen, denn eine zu einem Eckpunkt korrespondierende USAN hat ihren Mittelpunkt relativ weit vom Nukleus entfernt. Als letzter Schritt wird noch eine Non-Maximum Unterdrückung durchgeführt, die eine Anhäufung von Merkmalen vermeidet (vgl. Kapitel 2.2.3).

Diese Art der Punktdetektion unterscheidet sich stark von den vorherigen Detektoren, da keine Bildableitungen berechnet werden und auch Rauschen im Bild keinen Einfluss auf die Stabilität der Ergebnisse hat. Außerdem kann der Algorithmus (unter Verwendung eines modifizierten geometrischen Schwellwerts von $g = \frac{3 \cdot n_{max}}{4}$) auch für die einfache Detektion von Kanten benutzt werden. In diesem Fall ist der Schwellwert nur nötig, um Kanten in verrauschen Regionen zu detektieren.

Die Detektionsrate für SUSAN ist mit allen bereits vorgestellten Detektoren vergleichbar. Die Lokalisationsgenauigkeit hingegen ist aufgrund der Rauschunabhängigkeit durchschnittlich besser (nur bei verwaschenen Bildern ist sie viel schlechter). Die Wiederholbarkeitsrate ist bei Skalierung besser als bei Moravec, jedoch schlechter als bei den anderen Detektoren. Für affine Transformationen jedoch ist die Rate schlechter als bei allen anderen. Der Rechenaufwand ist mit dem von Moravec und Wang/Brady zu vergleichen.

2.2.7 FAST

Der 2005 von Edward Rosten und Tom Drummond vorgestellte Detektor „FAST“ (Features from Accelerated Segment Test) [RD05] [RD06b] betrachtet einen Bresenham Kreis mit einem Radius von 3 und einem Kreisumfang von 16 Pixeln um das zu untersuchende potentielle Merkmal p. Der Segmenttest besteht darin, zu überprüfen, ob der Intensitätsunterschied zwischen p und mindestens 12 zusammenhängenden

Pixeln in diesem Kreis größer oder kleiner als ein bestimmter Schwellwert t ist. Ist dies der Fall, so liegt ein Feature vor. Interessant an dieser Unterscheidung ist, dass die Merkmale in positive und negative eingeteilt werden können, so dass später beim Matching kein Vergleich von positiv-negativ Paaren stattfinden muss.

Um den Detektor zu beschleunigen, wird der Segmenttest mit Hilfe einer Methode des maschinellen Lernens verändert. Zum Aufbau eines Entscheidungsbaums wird der ID3 (Iterative Dichotomiser 3) Algorithmus [Mit97] benutzt, durch den alle Ecken klassifiziert werden können und somit die Reihenfolge der zu testenden Pixel optimiert wird. Anzumerken ist, dass dieser optimierte FAST Detektor nicht zu exakt dem Ergebnis des Segmenttests führt, da nicht alle möglichen Ecken durch den Algorithmus abgedeckt werden. Die Autoren geben in [RD05] jedoch an, dass zum einen die Entscheidungsfindung so manipuliert werden kann, dass die beiden Algorithmen das exakt gleiche Ergebnis erzielen, und zum anderen, dass die meisten Feature Detektoren in gewisser Weise heuristisch angelegt sind (Kompromiss zwischen Güte der Lösung und dem Rechenaufwand) und die beiden FAST Varianten sich nur geringfügig in ihrer Heuristik unterscheiden.

Auch beim FAST Detektor besteht der abschließende Schritt darin, eine Non-Maximum Unterdrückung durchzuführen (vgl. Kapitel 2.2.3). Da der Segmenttest jedoch keinen Wert für die „cornerness“ eines Merkmals zurückliefert, kann keine direkte Berechnung durchgeführt werden. Deshalb wird für jedes gefundene Merkmal eine „Score-Funktion“ V aus der Summe der absoluten Differenzen des Merkmalpunktes und jedem betrachteten Punkt im Kreis berechnet. Mit diesem Wert für V wird nun die „Non-Maximum Unterdrückung“ durchgeführt.

Schon im einfachen Segmenttest übertrifft der FAST Detektor in Bezug auf den Rechenaufwand alle anderen Featuredetektoren um Weiten. Von den Autoren durchgeführte Tests (auf einem Opteron mit 2.6 GHz) zeigen, dass SUSAN 7x, Harris 18x und der SIFT Detektor (DoG) sogar 45x länger zur Merkmalsdetektion brauchen. Die Wiederholbarkeitsrate hängt sehr stark vom Rauschlevel des Bildes ab. Während bei wenig verrauschten Bildern bessere Ergebnisse als bei Harris, SUSAN und SIFT festzustellen sind, kann der FAST Detektor bei verrauschten Bildern aufgrund seines Designs keine stabilen Ergebnisse mehr liefern. Obwohl zur Lokalisationsgenauigkeit und Detektionsrate leider keine Vergleichswerte vorliegen, wird in [TM08] angegeben, dass der FAST Detektor auch in diesem Bereich mit den rechenaufwändigeren Detektoren qualitativ vergleichbar arbeitet und sogar für manche Anwendungen bessere Ergebnisse erzielt.

2.2.8 Difference of Gaussians (SIFT Detector)

Der 1999 von Davd G. Lowe vorgestellte „SIFT Operator“ (Scale Invariant Feature Transform) [Low99] [Low04] ist ein skalierungsinvariantes Verfahren zum Detektieren und Beschreiben von lokalen Merkmalen in Bildern. Es ist also ein kombinierter Detektor und Deskriptor. In diesem Abschnitt wird der Detektor vorgestellt, für eine Beschreibung des SIFT-Deskripors siehe Kapitel 2.3.1.

Der Detektor arbeitet im Skalenraum [Lin93] eines Bildes, um Features unabhängig von der Skalierung der Bilder zu finden. Dieser Skalenraum ist als dreidimensionale Funktion

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{2.18}$$

mit dem Gaußkern $G(x, y, \sigma)$ und dem Eingangsbild $I(x, y)$ definiert. Die Repräsentation des Skalenraums ist eine „überabgetastete Pyramide“, eine Kombination aus zwei Techniken: den „Kaskaden“ und den „Pyramiden“ [Dic05]. Die Pyramidenstruktur skaliert das Bild in verschiedene Auflösungen, und die Kaskadenstruktur wendet auf jeder dieser Auflösungsstufen („Oktaven“) inkrementell, mit dem Faktor k gewichtet, eine Glättung mit einem Gaußkern an.

Die Featuredetektion wird jedoch nicht direkt auf diesen Skalenraum angewandt, sondern aus Gründen der Stabilität und Effizienz auf den „Difference of Gaussians“ (DoG), also den Differenzen zweier benachbarten Ebenen in einer Oktave (hier „Laplace-Ebenen“ genannt):

$$\begin{aligned} D(x, y, \sigma) &= (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \\ &= L(x, y, k\sigma) - L(x, y, \sigma) \end{aligned} \tag{2.19}$$

Die Merkmalsdetektion wird durchgeführt, indem jedes Pixel auf jeder Laplace-Ebene (außer den Randebenen) mit seinen acht Nachbarpixeln derselben Skalierungsstufe und den jeweils neun Pixeln in den zwei benachbarten Laplace-Ebenen verglichen wird. Als Merkmalskandidat wird jenes Pixel akzeptiert, das entweder größer oder kleiner als alle untersuchten Nachbarpixel ist.

Diese Kandidaten werden in einem weiteren Schritt durch eine *Subpixel-Lokalisier-*

ung und die darauf folgende *Entfernung von Kantenpixeln* verfeinert. Die *Subpixel-Lokalisierung* wird durchgeführt, indem versucht wird dem Merkmal mit Hilfe einer Taylorreihe eine dreidimensionale quadratische Funktion anzupassen, um damit dessen Subpixelposition zu interpolieren. Für die *Entfernung von Kantenpixeln* wird die 2×2 Hesse-Matrix des Merkmals untersucht. Anhand der Eigenwerte dieser Matrix werden Kantenpixel aus der Menge der Merkmalskandidaten verworfen.

In einem abschließenden Schritt wird jedem detektierten Merkmal eine Orientierung zugewiesen, um eine Rotationsinvarianz zu erzielen. Dazu werden aus der Summe der quadrierten horizontalen und vertikalen Differenzen in der Nachbarschaft eines Merkmals die Orientierungen $\theta(x, y)$ und Stärken der Steigungen $m(x, y)$ dieser Pixel berechnet. Die Orientierungen und die mit einer Gaußmaske gewichteten Steigungen werden für jedes Merkmal in einem 36 Urnen (engl. „bins“) großen Histogramm gespeichert. Die tatsächliche Orientierung eines Merkmals wird durch das Ermitteln der Urne mit dem größten Wert bestimmt.

Der Vorteil dieser Art der Merkmalsdetektion ist, dass durch die skalierungsinvarianz Korrespondenzen gefunden werden können, die bei anderen Detektoren wie z. B. Harris (siehe 2.2.2) nicht auftauchen würden. Jedoch wird dieser Vorteil auf Kosten des Rechenaufwandes erzielt, was SIFT aufgrund des rechenintensiven Aufbaus der Pyramide zu einem der langsamsten Detektoren und damit auch laut [LLF05] für Echtzeit-Anwendungen schwer verwendbar macht.

Eine sehr ausführliche Beschreibung des SIFT-Operators kann in [Dic05] gefunden werden.

2.3 Deskriptoren

Da bei der Verwendung von Punktdetektoren das als interessant detektierte Pixel (außer des Intensitätswertes) keine Aussagekraft besitzt, muss für die eindeutige Identifizierung eine Beschreibung gefunden werden, die es ermöglicht Pixel voneinander unterscheiden zu können. Die sogenannten „Deskriptoren“ liefern eine solche eindeutige Merkmalscharakteristik und dienen somit ihrer Identifizierung und Unterscheidung.

Es existieren verschiedene Arten von Deskriptoren, von denen der einfachste aus einem Vektor besteht, der die Informationen der benachbarten Pixel beinhaltet. Mit Hilfe einer Kreuzkorrelation kann nun die Ähnlichkeit der zwei Deskriporen bestimmt werden, was jedoch aufgrund der hohen Dimensionalität dieses Deskriptors

zu einem sehr hohen Rechenaufwand führt. Deshalb wird diese Technik meistens nur in zeitunkritischeren Aufgaben eingesetzt. Andere Ansätze können laut Mikolajczyk und Schmid [MS05] grob in *Verteilungsbasierte Deskriptoren*, *Raumfrequenztechniken* und *Differentialdeskriptoren* unterteilt werden.

Die *verteilungsbasierten Deskriptoren* benutzen zur Darstellung der Charakteristik eines Merkmalspunktes Histogramme. Der einfachste Fall ist hierbei, die Pixelintensitäten lediglich durch ein unmodifiziertes Histogramm darzustellen. Ein weiterer Ansatz (Zabih und Woodfill [ZW94]) beruht auf der Anordnung und den gegenseitigen Beziehungen der Pixelintensitäten in einem Histogramm. Auch die von Lowe entwickelte *Scale Invariant Feature Transform (SIFT)* [Low04] benutzt ein 3D-Histogramm, das aus Gradientenpositionen und -orientierungen besteht.

Die *Raumfrequenztechniken* beschreiben den Inhalt der Pixelnachbarschaft meist mit Hilfe der „Gabor-Transformation“. Obwohl aus der Fouriertransformation laut [MS05] keine expliziten räumlichen Beziehungen zwischen Merkmalspunkten resultieren und ihre Basisfunktionen unendlich sind, kann diese trotzdem basierend auf einer speziellen, gefensterten Fourier-Transformation (der „Gabor-Transformation“) in einen lokalen Ansatz adaptiert werden. Ein Nachteil dieser Methode ist, dass eine große Anzahl von Gabor-Filtern erforderlich ist, um geringe Frequenz- und Orientierungsänderungen zu messen.

Die *Differentialdeskriptoren* benutzen zur Beschreibung eines Merkmalspunktes die Eigenschaften der lokalen Ableitungen der Bilder (*local jet*, die Menge aller räumlichen, partiellen Ableitungen an einem Punkt). In diese Gruppe von Deskriptoren fallen auch die von Freeman und Adelson [FA91] entwickelten „steuerbaren Filter“ (engl. steerable filters). Basierend auf den Komponenten des *local jet*, steuern diese Filter die Ableitungen in eine bestimmte Richtung. Werden sie z. B. in Richtung der Gradienten gesteuert, werden sie rotationsinvariant.

Im folgenden Abschnitt 2.3.1 wird der SIFT-Deskriptor detaillierter beschrieben.

2.3.1 SIFT (Deskriptor)

Aufbauend auf dem SIFT-Detektor (siehe 2.2.8) hat Davd G. Lowe in [Low99] und [Low04] einen skalierungsinvarianten Deskriptor vorgestellt. Dieser Deskriptor beschreibt jedes Merkmal eindeutig anhand eines hochdimensionalen Vektors. Zur Erstellung dieses Vektors wird die Nachbarschaft des zu beschreibenden Merkmals in acht 4×4 große Quadrate unterteilt. Hierfür wird die Ebene der Pyramide benutzt, die

dem Merkmal am nächsten liegt. In jedem dieser acht Teilfenster der insgesamt 128 Pixel großen Nachbarschaft wird nun ein Orientierungshistogramm (auch Gradientenhistogramm genannt) mit jeweils acht Urnen berechnet[1]. Diese Urnen beschreiben die vollen 360° einer Rotation in 45°-Schritten. Die hier eingetragenen Orientierungen verstehen sich relativ zur Orientierung des Merkmals. Der Deskriptor besteht aus den sequenziell nacheinander eingetragenen Werten der Histogramme in einen Spaltenvektor.

Die Deskriptoren sind zu diesem Zeitpunkt noch nicht invariant gegenüber Kontrastveränderungen. Um dies annäherungsweise zu erreichen, werden die Vektoren auf eine Einheitslänge normiert.

Für eine ausführliche Beschreibung des SIFT-Deskriptors wird an dieser Stelle auf [Dic05] verwiesen.

Erweiterungen

Mehrere später beschriebene Deskriptoren haben sich am SIFT-Deskriptor orientiert. So haben Krystian Mikolajczyk and Cordelia Schmid in [MS05] die SIFT Erweiterung „GLOH“ (Gradient Location and Orientation Histogram) vorgestellt, bei der eine größere Region zur Erstellung der Histogramme verwendet wird, der Deskriptor dabei jedoch mittels Hauptkomponentenanalyse (englisch: Principal Components Analysis, PCA) auf eine Größe von 64 Einträgen beschränkt wird. Ziel dieser Erweiterung ist die Robustheit und Unterscheidbarkeit von SIFT-Merkmalen zu verbessern.

Herbert Bay, Tinne Tuytelaars und Luc van Gool haben sich für ihre in [BTvG06] vorgestellte Detektor-Deskriptor-Kombination „SURF“ (Speeded Up Robust Features) zwar an SIFT orientiert, jedoch eine etwas andere Herangehensweise gewählt. Der Deskriptor basiert auf der Verteilung von Haar-Wavelet-Responses in der Nachbarschaft eines Merkmals. Um die Rechenzeit so gering wie möglich zu halten, wird auf Integralbilder zurückgegriffen. Auch hier wird die Größe des Deskriptors auf 64 Einträge beschränkt, was die Matching-Zeit gering hält. Die Autoren geben in [BTvG06] an, dass SURF alle bisherigen Deskriptoren in ihrer Leistung übertrifft.

[1] Die hier beschriebene Nachbarschaftsgröße ist von Lowe in [Low04] vorgeschlagen worden. Sie kann jedoch je nach Anwendungskontext variiert werden.

2.4 Korrespondenzproblem (Matching)

Aus der Merkmalsdetektion in zwei Bildern entstehen die zwei Mengen $\{m_i\}$ und $\{m'_j\}$ von 2D-Merkmalen. Ziel des Matchings ist nun, die Elemente dieser zwei Mengen mit Hilfe von Ähnlichkeits- bzw. Korrelationsmaßen zu Merkmalspaaren $m_i \leftrightarrow m'_i$ zusammenzufassen. In den folgenden Abschnitten 2.4.1 und 2.4.2 werden zwei der gebräuchlichsten Maße beschrieben.

2.4.1 Summe quadrierter/absoluter Differenzen (SSD, SAD)

Die Summe quadrierter/absoluter Differenzen sind die zwei einfachsten Arten, Merkmale miteinander zu vergleichen. Nehmen wir an, es wurden das Merkmal f in Bild I an der Position x, y, und das Merkmal f' in Bild I' an der Position $x + i, y + j$ gefunden. Sei also:

$$\Delta_{x,y} = I_{x,y} - I'_{x+i,y+j} \tag{2.20}$$

das Differenzbild von I und I', so ist

$$d_{SAD} = \sum_{y=y-\frac{m}{2}}^{\frac{m}{2}} \sum_{x=x-\frac{n}{2}}^{\frac{n}{2}} |\Delta_{x,y}| = \sum_{y=y-\frac{m}{2}}^{\frac{m}{2}} \sum_{x=x-\frac{n}{2}}^{\frac{n}{2}} |I_{x,y} - I'_{x+i,y+j}| \tag{2.21}$$

die *Summe absoluter Differenzen* zweier Bilder in einem Fenster der Größe $m \times n$. Analog ist:

$$d_{SSD} = \sum_{y=y-\frac{m}{2}}^{\frac{m}{2}} \sum_{x=x-\frac{n}{2}}^{\frac{n}{2}} \Delta^2_{x,y} = \sum_{y=y-\frac{m}{2}}^{\frac{m}{2}} \sum_{x=x-\frac{n}{2}}^{\frac{n}{2}} (I_{x,y} - I'_{x+i,y+j})^2 \tag{2.22}$$

die *Summe quadrierter Differenzen* zweier Bilder in einem Fenster der Größe $m \times n$.

Diese beiden Ähnlichkeitsmaße haben den Vorteil, dass sie sehr einfach und effizient zu implementieren sind. Negativ ist jedoch anzumerken, dass beispielsweise durch Bildrauschen entstandene Ausreißer genauso stark in das Ergebnis eingehen, wie

„echte“ Fehler, die auf unterschiedliche Merkmale zurückzuführen sind. Außerdem ist dieses Maß nicht invariant gegenüber Kontrast- und Helligkeitsunterschieden der Art: $I' = \alpha I + \beta$. Die Bilder müssen also vorher Farb- und Helligkeitsnormiert sein, damit sinnvolle Ergebnisse entstehen können.

Um eine Invarianz gegenüber Kontrast- und Helligkeitsunterschieden zu erlangen wird oft auch die *normalisierte Summe quadrierter Differenzen* (NSSD) benutzt:

$$\begin{aligned} d_{NSSD} &= \frac{1}{\sigma\sigma' N} \sum_{y=y-\frac{m}{2}}^{\frac{m}{2}} \sum_{x=x-\frac{n}{2}}^{\frac{n}{2}} \Delta_{x,y}^2 \\ &= \frac{1}{\sigma\sigma' N} \sum_{y=y-\frac{m}{2}}^{\frac{m}{2}} \sum_{x=x-\frac{n}{2}}^{\frac{n}{2}} \left(I_{x,y} - I'_{x+i,y+j}\right)^2 \text{, mit } N = m \cdot n \end{aligned} \tag{2.23}$$

mit

$$\sigma = \frac{1}{N} \sqrt{\sum_{(x,y)\in W} (I_{x,y} - \mu)^2} \tag{2.24}$$

$$\mu = \frac{1}{N} \sum_{(x,y)\in W} I_{x,y} \tag{2.25}$$

2.4.2 NCC - Normalisierte Kreuzkorrelation

Die normalisierte Kreuzkorrelation beruht auf dem aus der Wahrscheinlichkeitstheorie stammenden Begriff der Kovarianz:

$$Cov(X, Y) = E[X - E(X), Y - E(Y)] \tag{2.26}$$

wobei X und Y Zufallsvariablen sind. Sollen nun die Ergebnisse auf Werte zwischen -1 und 1 normiert werden, so werden sie durch das Produkt der beiden Standardabweichungen geteilt:

$$Corr(X,Y) = \frac{Cov(X,Y)}{\sigma_X \sigma_Y} = \frac{E[X - E(X), Y - E(Y)]}{\sigma_X \sigma_Y} \tag{2.27}$$

Die Korrelation zweier Zufallsvariablen beträgt 1, wenn ein linearer Zusammenhang besteht. 0 wird das Ergebnis, wenn die Variablen unkorreliert sind und -1, wenn ein negativer linearer Zusammenhang besteht. Auch non-integer Werte zwischen -1 und 1 sind möglich.

Werden nun die Bildintensitäten als Proben von Zufallsvariablen angesehen, so ist:

$$NCC = \frac{1}{\sigma\sigma' N} \sum_{(x,y)\in W} (I_{x,y} - \mu)(I'_{x+i,y+j} - \mu') \tag{2.28}$$

wobei die Fensterfunktion W die $N \times N$ Nachbarschaft beschreibt, die zur Berechnung des NCC mit einbezogen wird. Um einen besseren NCC-Wert zu erzielen, hat sich hier die Verwendung einer Gaußmaske als vorteilhaft erwiesen. Außerdem ist:

$$\mu = \frac{1}{N} \sum_{(x,y)\in W} I_{x,y} \tag{2.29}$$

$$\sigma = \frac{1}{N} \sqrt{\sum_{(x,y)\in W} (I_{x,y} - \mu)^2} \tag{2.30}$$

Analog sind die Berechnungen für μ' und σ'. Dieses Maß ist, im Gegensatz zu der SSD und der SAD, invariant gegenüber Kontrast- und Helligkeitsunterschieden.

Um die Rechenzeit der Korrelationen zwischen den Merkmalen zu verringern, sollte der Suchraum, in dem nach Merkmalen im zweiten Bild gesucht wird, nicht auf das ganze Bild ausgebreitet werden, sondern auf eine angemessene Größe beschränkt werden.

2.5 Liniendetektion und Matching

In diesem Abschnitt sollen verschiedene Methoden der Liniendetektion vorgestellt werden. Aufgrund von auftretenden Ungenauigkeiten beim Matching von Punktmerkmalen in Regionen die weiter von der Kamera entfernt sind, ist es von Vorteil die Schätzung der Kamerapose durch Linien zu unterstützen. Auch in bestehenden Arbeiten hat sich eine Liniendetektion als sinnvoll herausgestellt (siehe [VLF04], [Ewe06] und [RD06a]).

Prinzipiell kann die Liniendetektion direkt mit Hilfe von *Template Matching* durchgeführt werden, das heißt durch Kantendetektoren erster oder zweiter Ordnung. Das Problem hierbei ist jedoch, dass die gefundenen Kantenpixel unzusammenhängend und ohne Aussage über die unterliegende Kontur des Objektes sind. Außerdem ist die Anzahl der Filtermasken extrem hoch, wenn Linien präzise lokalisiert werden sollen. Stattdessen werden diese Kantenbilder als Grundlage für komplexere Liniendetektoren benutzt [TV98].

2.5.1 Hough-Transformation

Die *Hough-Transformation* ist ein auf schwarz-weiß Kantenbildern arbeitendes Verfahren zur Detektion von Linien, Kreisen und vielen anderen geometrischen Formen, die durch Parameter beschreibbar sind. In ihrer klassischen, von Paul V. C. Hough 1962 unter dem Namen „Method and Means for Recognizing Complex Patterns“ [Hou62] patentierten Form konnten lediglich Linien detektiert werden. 1972 erweiterten Richard Duda und Peter Hart das Verfahren um die Möglichkeit allgemeine geometrische Formen in Bildern zu erkennen [DH72]. Außerdem gaben sie an, dass die Parameterdarstellung einer Geraden durch die Hessesche Normalform den Rechenaufwand des Algorithmus verringert. In der klassischen Methode wurde eine Gerade durch ihre Steigung und den y-Achsen-Abschnitt beschrieben, was jedoch den Nachteil hat, dass beide Werte für vertikale Linien gegen Unendlich streben und dadurch in der Bildanalyse problematisch anzuwenden sind. In der Bildverarbeitung hat sich aus diesem Grund die von Duda und Hart vorgeschlagene Parameterdarstellung

$$d = x \sin\theta y \cos\theta, \text{ mit } \theta \in [0, \pi] \tag{2.31}$$

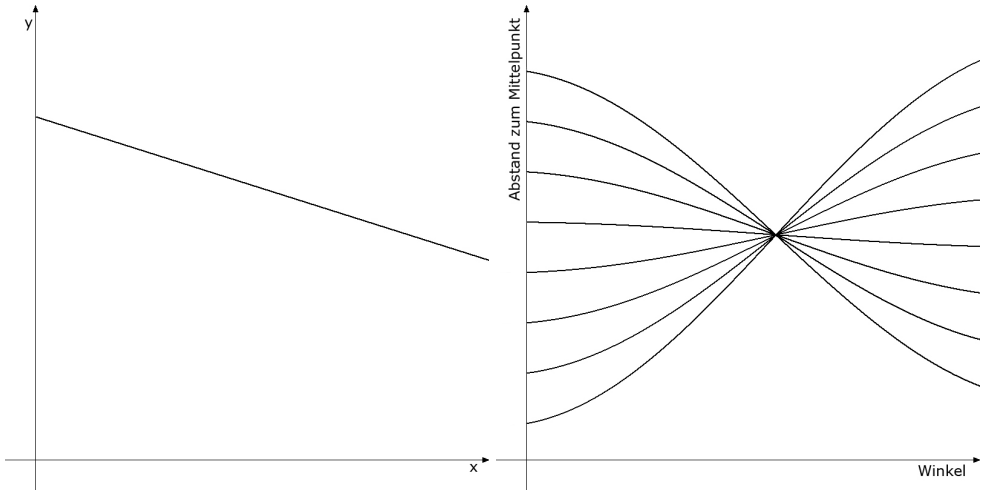


Abbildung 2.2: Schematische Darstellung der Hough-Transformation. Links: eine Gerade im (x, y)-Raum. Rechts: die der Gerade entsprechenden Sinusoide im (θ, d)-Raum. Der Schnittpunkt der Kurven entspricht den Parametern der Geraden.

einer Geraden durchgesetzt. Die folgenden Formeln beziehen sich aus Gründen der Leserlichkeit auf die Hough-Transformation für Geraden. Sie können ohne Probleme auch für jede beliebige Parameterdarstellung abgewandelt werden.

Als Vorverarbeitungsschritt wird eine Kantendetektion durchgeführt, die bestimmte auf den geometrischen Formen eines Bildes liegende Pixel erkennt. Diese Pixel sind jedoch unzusammenhängend und besitzen keine Aussage über die Art der unterliegenden geometrischen Form. Die Aufgabe der Hough-Transformation ist, diese Pixel zu gruppieren und in eine durch Parameter darstellbare Form zu bringen. Dazu wird jedes Pixel des Bildes in den (θ, d)-Raum (auch *Hough-Raum* genannt) transformiert, indem es nicht durch seine Koordinaten (x, y) sondern durch die Parameter der zugrundeliegenden Form (hier Geraden) dargestellt wird (siehe Gleichung 2.31).

Sei nun durch eine vorherige Kantendetektion eine Menge von n Kantenpunkten $\{(x_0, y_0), ..., (x_n, y_n)\}$ gegeben, so wird jeder Punkt (x_i, y_i) in den Hough-Raum transformiert:

$$d = x_i \sin\theta y_i \cos\theta \tag{2.32}$$

Jede Linie im (x, y)-Raum entspricht hierbei einem Punkt im (θ, d)-Raum und da-

mit jeder Punkt im (x, y)-Raum einem Sinusoid im (θ, d)-Raum. Die Menge von Punkten, die zu einer bestimmten Geraden gehört produziert Sinusoide, die sich an ihren (θ, d)-Parametern im Hough-Raum schneiden (siehe [DH72]). Somit müssen zur Detektion von Geraden lediglich die Schnittpunkte der Sinusoide im Parameterraum gefunden werden. Jeder Schnittpunkt definiert dann die Parameter θ und d einer Geraden (siehe Abbildung 2.2).

Eine analytische Berechnung der Hough-Transformation (z. B. durch eine Minimierungstechnik) führt dazu, dass der Rechenaufwand quadratisch zu der Gesamtanzahl der Pixel im Bild steht. Duda und Hart beschreiben deshalb in [DH72] die Quantisierung des (θ, d)-Raums in ein Gitter mit viereckigen Zellen („Framelets“ in [Hou62]). Dieses *Akkumulator* genannte Gitter wird durch ein n-dimensionales Array repräsentiert, wobei n die Anzahl der gesuchten Parameter darstellt (also im Fall von Geraden $n = 2$). Für jedes Pixel (x_i, y_i) im Bild werden die nach 2.32 entsprechenden Werte („Bins“) im Akkumulator-Array hochgezählt. Die Schnittpunktsuche der Sinusoiden beschränkt sich daher auf die Suche nach lokalen Maxima im Akkumulator-Array.

Da Kantenpixel unabhängig voneinander untersucht werden, kann die Hough-Transformation sehr gut mit Verdeckungen umgehen. Auch Rauschen beeinflusst die Liniendetektion kaum, denn die tatsächlich zu einer Geraden gehörenden Bins werden bis zu einem gewissen Rauschlevel nicht verändert. Der größte Nachteil ist der mit steigender Anzahl von unbekannten Parameter steigende Rechenaufwand und die Tatsache, dass Formen, die nicht gesucht werden, die Ergebnisse beeinflussen können. Zum Beispiel können sehr flache Ellipsen die Erkennung von Geraden in einem Bild verfälschen.

Detaillierte Beschreibungen der Hough-Transformation für Kreise und Ellipsen sowie der *Generalisierten Hough-Transformation* (für komplexere Formen) können in [NA02] und [TV98] nachgelesen werden.

2.5.2 Marcel and Cattoen Edge and Line Detector

Baptiste Marcel und Michel Cattoen stellen in [MC97] einen auf low-level Bildanalyse basierenden Algorithmus zur Kanten- und Liniendetektion vor. Er wurde im Zusammenhang mit einer Studie über das Erkennen von zweidimensionalen symbolischen Codes entworfen und besteht aus fünf Schritten: *Kantendetektion*, Berechnung der *Kantenrichtung*, *Linking* der Kanten, *Kettenbegradigung* und *Linienrestaurierung*.

Die *Detektion* der Kanten wird mit einem Laplace-Filter durchgeführt. Dieser Filter hat den Vorteil, dass sein Nulldurchgang (engl. „zero crossing“) dünne Kanten liefert, weshalb hier im Gegensatz zu anderen Methoden zur Liniendetektion eine anschließende Kantenausdünnung wegfällt. Mit Hilfe von zwei direktionalen Gradientenmasken wird im zweiten Schritt die *Richtung* für jede detektierte Kante berechnet. Marcel und Cattoen geben in [MC97] an, dass die Kantenrichtung nicht mit hoher Genauigkeit berechnet werden muss und acht grobe Richtungen mit einer maximalen Abweichung von 22,5° ausreichen. Der dritte Schritt besteht aus der *Zusammenführung* (engl. „Linking“) von Kanten zu zusammenhängenden Ketten, basierend auf ihrer Richtung und räumlichen Nähe zueinander. Da die zusammengeführten Kanten nach der Zusammenführung noch großwinklige Ecken beinhalten, werden die gelinkten Ketten im vierten Schritt auseinandergebrochen um sie zu *begradigen*. Dafür wird eine Verbindungslinie vom Start- zum Endpunkt gelegt und von jedem Punkt auf der Linienkette der Abstand zu dieser Linie berechnet. Am Punkt mit dem größten Abstand zur Linie wir die Kette aufgebrochen und der Algorithmus wird auf den beiden neu entstandenen Kettengliedern weitergeführt, bis der größte Punkt-Linien-Abstand unter einem gewissen Schwellwert liegt. Der abschließende Schritt besteht aus einer Zusammenführung von Linien, die durch nicht-detektierte Kantenstücken zwischen ihnen separiert sind.

2.5.3 Korrespondenzsuche

Da sich zwei in unterschiedlichen Bildern gefundene Liniensegmente meistens in ihrer Länge unterscheiden, ist es nicht möglich diese miteinander zu matchen. Deshalb wird in dieser Diplomarbeit die Liniendetektion nur auf dem Bild des gerenderten 3D-Modells durchgeführt. Um eine korrespondierende Linie im Kamerabild zu finden, können mehrere Strategien verfolg werden. Im folgenden werden zwei dieser Methoden vorgestellt.

Gradientenverfahren

In [Ewe06] wird ein Verfahren angewandt, das eine korrespondierende Linie im Kamerabild anhand von direktionalen Gradientenmasken sucht. Da die Referenzlinien des 3D-Modells hier, genauso wie das Modell selber, lediglich in einer Textdatei vorliegen, kann beim Matchen von Linen nicht auf a priori Wissen über die Semantik der Linien zurückgegriffen werden. Deshalb wird entlang orthogonaler Suchlinien

nach Kanten im Kamerabild gesucht. Dies geschieht mit Hilfe von vorberechneten direktionalen Gradientenmasken, die je nach Steigung der Suchlinie ausgesucht werden. Wird ein Maximum auf einer Suchlinie gefunden, so speichert der Algorithmus die jeweiligen Koordinaten. Aus der so gefundenen Menge von Punkten werden mit RANSAC (siehe Kapitel 2.6.1) Outlier entfernt und aus den übriggebliebenen Inliern eine Linie gebildet (vom räumlich ersten bis zum letzten Inlier).

NCC-Verfahren

Das in dieser Diplomarbeit benutze Verfahren basiert auf der normalisierten Kreuzkorrelation (NCC, siehe 2.4.2), da durch die Verwendung eines 3D-Modells Wissen über die unterliegende Struktur der Linien vorhanden ist. Zuerst werden im Modellbild Linien detektiert. Jede dieser Linien wird in eine Reihe von Punkten aufgebrochen, wonach ausgehend von jedem Punkt eine zum Liniensegment orthogonale Suchlinie berechnet wird. Auf jeder dieser Suchlinien wird schließlich mit Hilfe der normalisierten Kreuzkorrelation nach Korrespondenzen gesucht. Dazu wird auch diese Suchlinie in eine Reihe von Punkten unterteilt, so dass jeder dieser Punkte mit dem entsprechenden Punkt auf dem Liniensegment verglichen werden kann. Das hier eindimensionale Suchfenster der normalisierten Kreuzkorrelation muss dabei die gleiche Steigung wie die Suchlinie besitzen, also auf dieser „entlanggleiten“. Der Punkt mit dem maximalen Korrelationswert wird als mögliche Korrespondenz gespeichert. Nachdem das Maximum für jeden Linienpunkt gefunden wurde, wird die entstandene Punktmenge mit dem RANSAC Algorithmus von „Outliern“ befreit (siehe Kapitel 2.6.1). Als Modell wird hierbei eine Gerade vorgegeben, das Distanzmaß ist der Punkt-Linien-Abstand. Nachdem die Punkmenge nun möglichst ausreißerfrei ist, wird die tatsächliche Linie mit Hilfe einer linearen Regression berechnet.

2.6 Robuste Schätzung

In der Bildverarbeitung ist die *robuste Schätzung* (oft auch *robuste Regression* genannt) ein wichtiges Werkzeug, um durch Messfehler verrauschte Datensätze verlässlich benutzen zu können. Da z. B. bei Punktkorrespondenzen nicht von normalverteiltem Rauschen ausgegangen werden kann, ist eine Verwendung der „Methode der kleinsten Quadrate“ (engl. „least squares method“) nicht möglich. Die hierdurch berechneten Modellparameter würden ungenau und nicht verlässlich sein. Deshalb wird eine *robuste Regression* verwendet, die durch die Eliminierung von

„Outliern“ (Definition siehe 2.6.1) eine verlässlichere Lösung liefert.

In den folgenden Abschnitten werden zwei dieser robusten Methoden, RANSAC (2.6.1) und die M-Schätzer (2.6.2), vorgestellt.

2.6.1 RANSAC (RANdom SAmple Concensus)

Der 1981 von Fischler und Bolles in [FB81] vorgestellte RANSAC-Algorithmus wird in der Bildverarbeitung zur Eliminierung von Ausreißern und damit zur Verbesserung der Robustheit eines Algorithmus benutzt. Hierzu unterscheidet RANSAC zwischen *Inliern* (Messwerte, die zu den Modellparametern, die durch die Mehrheit der Daten vorgegeben wird, passen) und *Outliern* (Ausreißer, also Messwerte, die das zu untersuchende Modell verfälschen). Auf eine merkmalsbasierte Poseschätzung bezogen bedeutet dies, dass falsche und schlecht lokalisierte Punktkorrespondenzen als Outlier klassifiziert werden. Im Zusammenhang mit RANSAC wird auch von einer *robusten Schätzung* gesprochen, da dieser sehr gut auch mit einer großen Menge von Outliern umgehen kann.

RANSAC benutzt eine minimale Menge an zufällig gewählten Messwerten, um die Parameter des Modells zu berechnen. Diese s Proben werden zunächst als ausreißerfrei angenommen. Mit dieser minimalen Probe werden zuerst die Modellparameter berechnet und dann die Distanz jedes Messwertes zu diesem Modell bestimmt. Ist die Distanz eines Messwertes geringer als ein bestimmter Schwellwert, so wird dieser als „Inlier“ klassifiziert. Ist die Distanz größer, wird er als „Outlier“ eliminiert. Diese als *Inlierklassifizierung* bezeichneten Schritte werden so oft wiederholt, bis die Menge der „Inlier“ (der *Consensus Set*) maximal ist. Dazu wird dieser *Consensus Set* nach jeder Iteration gespeichert und seine Größe mit der Setgröße der nächsten Iteration verglichen. Denn steigt die Anzahl der „Inlier“ von einer Iteration zur anderen, so wurden Parameter gefunden, die das Modell besser unterstützen. Abschließend wird mit dem maximalen *Consensus Set*, der im Idealfall keine „Outlier“ mehr enthält, eine traditionelle Regression durchgeführt.

Ablauf des Algorithmus

1. Die Messwerte werden bestimmt.

2. Robuste Schätzung mit RANSAC: Wiederhole für N Proben (N wird nach 2.6.1 adaptiv berechnet)

(a) Eine zufällige Probe der Größe s wird gewählt, mit der die Parameter des Modells M berechnet werden.

(b) Die Distanz d von jedem Datum der Datenmenge zu dem berechneten Modell wird bestimmt.

(c) Die Menge der Inlier für die gilt: $d^2 < t^2$ wird bestimmt.

(d) Ist der Consensus Set größer als das bisherige Maximum, wird er gespeichert, sonst verworfen.

3. Schätze die finalen Parameter mit Hilfe von traditionellen Regressionsmethoden und dem maximalen Consensus Set.

Probenauswahl

Als Grundlage der *Inlierklassifizierung* sollten die zufällig gewählten Messwerte zu Beginn des Algorithmus auf ihre Tauglichkeit untersucht werden. Schlechte Proben sind:

- Messwerte, die linear abhängig sind, da somit keine zuverlässige Pose berechnet werden kann
- Messwerte, die zu nah beieinander liegen, da in diesem Fall das berechnete Modell zu ungenau werden kann, vor allem beim Berechnen der Rotation

Anzahl der Proben und Terminierung des Algorithmus

Um die Rechenzeit so gering wie möglich zu halten, ist es sinnvoll, nicht jede mögliche Probe zu testen. Die Anzahl der Proben N sollte so gewählt werden, dass mit der Wahrscheinlichkeit p wenigstens eine der Proben der Größe s keine Outlier besitzt. Für diese Wahrscheinlichkeit wird normalerweise $p = 0.99$ gewählt [HZ04]. Sei nun $\epsilon = 1 - w$ die Wahrscheinlichkeit, dass der gewählte Punkt ein Outlier ist, so werden mindestens N Proben der Größe s gebraucht, für die gilt:

$$N = \frac{log(1-p)}{log(1-(1-\epsilon)^s)} \tag{2.33}$$

Da in der Praxis die Outlierwahrscheinlichkeit ϵ oft nicht bekannt ist, sollte die Anzahl der zu benutzenden Proben adaptiv bestimmt werden. Initialisiert wird ϵ mit einer Schätzung des schlimmsten Falles (also der Fall in dem die meisten Proben

Outlier sind), sinnvollerweise z. B. 1.0. Damit würde dann der anfängliche Wert von N unendlich groß. Wird eine Probe entdeckt, die weniger Outlier besitzt als die zuvor gewählte, wird ϵ auf diesen Wert gesetzt und N damit neu berechnet, bis die Anzahl der bereits untersuchten Proben diese Zahl übersteigt.

Diese adaptive Berechnung verläuft laut [HZ04] folgendermaßen:

- Initialisierung: $\epsilon = 1.0$, $N = \infty$ und SampleCount = 0
- solange $N >$ SampleCount:
  - Wähle zufällige Probe und bestimme Anzahl der Inlier
  - Setze $\epsilon = 1 - \frac{\text{Anzahl der Inlier}}{\text{Gesamtanzahl Punkte}}$, falls ϵ kleiner als im vorherigen Schritt
  - Berechne N mit der neuen Inlierwahrscheinlichkeit ϵ und $p = 0.99$
  - Erhöhe SampleCount um 1
- Terminiere Algorithmus

2.6.2 M-Schätzer

Die *M-Schätzer* sind eine Klasse von „Maximum-Likelihood-Artigen“ Schätzfunktionen, die jedoch im Gegensatz zu Maximum-Likelihood-Schätzern robuster gegen nicht normalverteiltes Rauschen sind. Laut [HRRA86] gehören sie zu den meistverwendetsten robusten Regressionsmethoden. Sie werden auch als „intelligente Schätzfunktionen“ bezeichnet, weil sie z. B. im Vergleich zur „Methode der kleinsten Quadrate“ die Möglichkeit bieten eine Gewichtung der Vertrauenswürdigkeit von Messwerten in die Minimierung mit einzubeziehen. Je größer die Abweichungen werden, desto kleiner werden die Messwerte gewichtet.

Jedes Minimierungsproblem der Form

$$\min_{\theta} \sum_{i=1}^{n} \rho(x_i - \theta) \tag{2.34}$$

oder

$$\sum_{i=0}^{n} \psi(x_i - \theta) = 0 \text{ bzw. } \sum_{i=0}^{n} w_i(x_i - \theta) = 0 \tag{2.35}$$

wird *M-Schätzer* genannt. ρ ist hierbei eine beliebige Funktion, ψ ihre Differentialfunktion und w eine Gewichtungsfunktion:

$$\psi(x - \theta) = \frac{\partial \rho(x - \theta)}{\partial \theta} \tag{2.36}$$

außerdem ist:

$$w_i = \frac{\psi(x_i - \theta)}{x_i - \theta}, \text{ mit } i = 1, \ldots, n \tag{2.37}$$

Die Gewichtungsfunktion ist somit dafür zuständig, dass mögliche Ausreißer die Lösung des Minimierungsproblems nicht zu sehr verfälschen.

Es existieren viele Arten von Funktionen ρ und ψ. Die in dieser Diplomarbeit verwendete Schätzfunktion ist der *Tukey-Schätzer* (auch „Tukeys Biweight“ genannt) [HMT83], dessen Funktion ab einer bestimmten Fehlergröße, die durch die Konstante a definiert werden kann, nicht mehr steigt und deshalb sehr robust gegenüber Ausreißern ist. Die drei Funktionen für den Tukey-Schätzer lauten:

$$\rho(x) = \begin{cases} \frac{a^2}{6}\left[1 - \left(1 - \left(\frac{x}{a}\right)^2\right)^3\right], & \text{für } |x| \leq a \\ \frac{a^2}{6}, & \text{sonst} \end{cases} \tag{2.38}$$

$$\psi(x) = \begin{cases} x\left(1 - \left(\frac{x}{a}\right)^2\right)^2, & \text{für } |x| \leq a \\ 0, & \text{sonst} \end{cases} \tag{2.39}$$

$$w(x) = \begin{cases} \left(1 - \left(\frac{x}{a}\right)^2\right)^2, & \text{für } |x| \leq a \\ 0, & \text{sonst} \end{cases} \tag{2.40}$$

Eine andere sehr beliebte Schätzfunktion ist der *Huber-Schätzer* [Hub81]:

$$\rho(x) = \begin{cases} \frac{x^2}{2}, \text{ für } |x| \leq a \\ a|x| - \frac{a^2}{2}, \text{ sonst} \end{cases} \tag{2.41}$$

Die M-Schätzer werden „Maximum-Likelihood-Artig“ genannt, weil sie die Maximum-Likelihood-Schätzung verallgemeinern. Wird nämlich als Funktion

$$\rho(x - \theta) = -\log f(x - \theta)$$

gewählt, so erhält man eine einfache Maximim-Likelihood-Schätzung.

2.7 Poseberechnung

Sind, wie in Kapitel 2.4 beschrieben, 2D/2D Korrespondenzen gefunden, so gibt es verschiedene Möglichkeiten, die Pose der Kamera zu schätzen. Im Folgenden werden zwei dieser Methoden vorgestellt.

Die erste Methode funktioniert, indem mit Hilfe der *Epipolargeometrie* die Fundamentalmatrix, die die intrinsischen Kameraparameter K sowie die essentielle Matrix E beinhaltet, berechnet wird.

$$F = (K^{-1})^T E K^{-1} \tag{2.42}$$

Ist K bekannt, kann E einfach aus der Fundamentalmatrix berechnet werden. Durch eine Faktorisierung von E ergeben sich die relative Position P und die relative Rotation R der beiden Kameras zueinander. Unter Verwendung der Rotation und Translation kann schließlich mit Hilfe der Triangulierung zu jeder 2D/2D Korrespondenz dessen 3D-Koordinate berechnet werden (siehe [TV98] zur Vertiefung der Epipolargeometrie und Triangulierung).

Können die Merkmale des einen Bildes jedoch auf die Struktur eines in OpenGL gerenderten 3D-Modells zurückgeführt werden, erschließt sich ein direkter Weg, nämlich die 2D/2D-Korrespondenzen $m_i \leftrightarrow m_i'$ in 2D/3D Korrespondenzen $M_i \leftrightarrow m_i'$ umzurechnen, um daraus eine Pose zu schätzen. Dazu wird zuerst mit Hilfe des z-Buffers die Tiefeninformation eines 2D-Punktes ermittelt. Anschließend wird

die GLU-Funktion *gluUnProject* verwendet, die anhand der nun vorliegenden x, y und z Koordinaten und der aktuellen Modelview- und Projection-Matrizen die 3D-Koordinaten eines Punktes wiederherstellen kann. Eine detaillierte Beschreibung folgt in Kapitel 3.2.3.

Auf Basis von 2D/3D-Korrespondenzen kann die Kamerapose entweder mit linearen oder nichtlinearen Methoden geschätzt werden. Diese werden in der folgenden Abschnitten 2.7.1 und 2.7.2 beschrieben.

2.7.1 Lineare Schätzung

Direkte, lineare Methoden zur Berechnung der Pose existieren schon seit dem 19. Jahrhundert. Der erste Lösungsansatz für die Verwendung in der Geodäsie wurde 1841 vom Mathematiker Johann August Grunert in [Gru41] vorgestellt [HLON94]. Zwei der wichtigsten linearen Verfahren im heutigen Rechnersehen sind die *Direkte Lineare Transformation* (DLT) und *POSIT*.

Direkte Lineare Transformation (DLT)

Die Grundlage für die DLT lieferte der Informatiker Ivan Edward Sutherland in [Sut64] (damals noch für eine 2D-Anwendung), die in [Fau93] und dann in [HZ04] für Aufgaben im Bereich des 3D-Rechnersehens adaptiert wurde. Mit DLT ist es möglich die gesamte Projektionsmatrix $P = K[R|t]$ zu schätzen. Dazu werden basierend auf der perspektivischen Projektion $m_i = PM_i$, mit $m_i = (u_i, v_i, 1)^T$ und $M_i = (X_i, Y_i, Z_i, 1)^T$ folgende Gleichungen aufgestellt:

$$u_i = \frac{P_{11}X_i + P_{12}Y_i + P_{13}Z_i + P_{14}}{P_{31}X_i + P_{32}Y_i + P_{33}Z_i + P_{34}} \tag{2.43}$$

$$v_i = \frac{P_{21}X_i + P_{22}Y_i + P_{23}Z_i + P_{24}}{P_{31}X_i + P_{32}Y_i + P_{33}Z_i + P_{34}} \tag{2.44}$$

Diese Gleichungen könnten in die Form $Ap = 0$ gebracht werden (siehe hierzu [HZ04]), wobei der Vektor p mit den Elementen P_{ij} der Projektionsmatrix gefüllt ist. Mit Hilfe einer Sigulärwertzerlegung (SVD) kann der Nullraum der Matrix A und somit die Lösung für p bestimmt werden. Ist die intrinsische Kameramatrix K

bekannt, kann die extrinsische Kameramatrix bis auf einen Skalierungsfaktor bestimmt werden. Diese Vorgehensweise wäre im Trackingkontext zwar anwendbar, jedoch nicht erfolgsversprechend, weil einerseits die Pixelkoordinaten nicht immer akkurat genug gemessen werden können und andererseits die Punktkorrespondenzen fehlerhaft sein können. Die fehlerhaften Punktkorrespondenzen können zwar durch die Verwendung von robusten Schätzern (wie z. B. RANSAC, siehe Kapitel 2.6.1) teilweise vermieden werden, jedoch stören auch wenige Ausreißer das Ergebnis derart, dass die DLT keine ausreichend guten Ergebnisse erzielen würde.

POSIT

Der aus zwei Algorithmen zusammengesetzte POSIT-Algorithmus zur Berechnung der Objektpose wurde 1992 von Daniel DeMenthon und Larry Davis in [DD92] vorgestellt. Der erste Algorithmus *POS* (Pose from Orthography and Scaling) berechnet basierend auf der Annahme, dass eine skaliert orthographisch Projektion vorliegt eine ungefähre Lösung für die Pose. Das heißt es wird ein lineares Gleichungssystem gelöst. Der zweite Algorithmus *POSIT* (POS with iterations) wendet im ersten Schritt die durch POS berechnete Pose auf die 3D-Koordinaten des Modells an, um wiederum mit Hilfe von POS die auf den „korrigierten“ Pixelkoordinaten basierende neue Pose zu berechnen. Diese Schritte werden iteriert, bis die Veränderung der Pixelkoordinaten unter einen bestimmten Schwellwert fällt. Laut DeMenthon und Davis konvergiert der Algorithmus im Allgemeinen nach vier bis fünf Iterationen. Zur Schätzung der Pose benötigt der POSIT-Algorithmus mindestens vier linear unabhängige 2D/3D-Punktkorrespondenzen. Mehr als vier Korrespondenzen haben den Vorteil, dass dadurch die Stabilität der geschätzten Pose verbessert werden kann. Ein ähnlicher Ansatz wird in [ODD96] verfolgt, hier jedoch unter Verwendung von koplanaren Punkten.

Die linearen Methoden haben den Vorteil, dass sie keine initiale Schätzung benötigen, um eine Pose zu berechnen. Der Nachteil ist jedoch, dass sie eine große Instabilität unter verrauschten Bedingungen zeigen. Sind die Pixelkoordinaten schlecht lokalisiert oder die Korrespondenzen teilweise falsch, so wird die berechnete Pose sehr ungenau.

2.7.2 Nichtlineare Optimierung

Die nichtlineare Optimierung hat gegenüber den linearen Schätzern den Vorteil, dass sie auch mit Ausreißern in der Messwertreihe gute Ergebnisse erzielen kann. Dazu benötigt sie jedoch eine initiale Schätzung der Kamerapose, da numerische Verfahren zum Lösen eines Optimierungsproblems nur lokale Minima bestimmen können [Alt02]. Vor der Verwendung der nichtlinearen Optimierung wir deshalb oft eine Initialpose mit linearen Lösungsverfahren ermittelt, so dass möglichst sicher gestellt werden kann, dass das „richtige“ lokale Minimum geschätzt wird.

Um nichtlineare Lösungsverfahren benutzen zu können, muss die zu optimierende Kamerapose, im Speziellen die Rotationsmatrix, in so wenig Parameter wie möglich zerlegt werden. Möglichkeiten hierzu wären der Eulerwinkel, die Axis-Angle-Darstellung oder Quaternionen. Aufgrund des bei Eulerwinkeln auftretenden „Gimbal Locks“, ist hier die Verwendung von Quaternionen von Vorteil. Jedoch haben Quaternionen den Nachteil, dass sie eine Rotation mit 4 Parametern beschreiben, wohingegen Eulerwinkel nur 3 brauchen.

Das Ziel von nichtlinearen Lösungsverfahren ist es, das Minimum bzw. Maximum einer nichtlinearen Zielfunktion zu bestimmen:

$$\min_{x \in \mathbb{R}^n} f(x) \tag{2.45}$$

Dazu können, abhängig von der Art des Optimierungsproblems, zum einen unterschiedliche Optimierungsverfahren und zum anderen verschiedene Fehlerfunktionen verwendet werden. Mögliche Optimierungsverfahren unterteilen sich in *ableitungsfreie* Methoden (z. B. das Downhill-Simplex-Verfahren oder der Levenberg-Marquardt-Algorithmus), Methoden basierend auf der *ersten Ableitung* (z. B. das Quasi-Newton-Verfahren) sowie Methoden basierend auf der *zweiten Ableitung* (z. B. das Newton-Verfahren).

Im Trackingkontext ist es sinnvoll den *Rückprojektionsfehler* zu minimieren, indem jeder 3D-Punkt mit Hilfe der Projektionsmatrix $P = K[R|t]$ auf die Bildebene rückprojiziert und die Differenz zu seiner 2D-Korrespondenz des Kamerabildes berechnet wird. Die einfachste Zielfunktion bzw. Fehlerfunktion hierfür wäre die Summe der Fehlerquadrate. Da jedoch bei Punktkorrespondenzen mit einer großen Menge an nicht normalverteilten Ausreißern zu rechnen ist, ist die Verwendung eines *robusten Schätzers* (siehe Kapitel 2.6.2) verlässlicher.

Kapitel 3

Markerloses Tracking

In diesem Kapitel wird in Abschnitt 3.1 ein Überblick über bestehende markerlose Trackingsysteme gegeben. Anschließend erfolgt in Abschnitt 3.2 eine Beschreibung des dieser Diplomarbeit zugrunde liegenden Systems.

3.1 Überblick anderer Ansätze

In diesem Kapitel sollen drei interessante Trackingansätze vorgestellt werden, die genau wie diese Diplomarbeit auf der Einbindung von 3D-Modellen basieren: das auf Kantendetektion basierende System für Outdoor-Anwendungen von Reitmayr und Drummond (3.1.1), das auf einem Linienmodell basierende System für industrielle Anwendungen von Wuest und Stricker (3.1.2) und das auf einem Modell und Referenzbildern basierende System von Lepetit et al. (3.1.3).

3.1.1 Reitmayr und Drummond

Gerhard Reitmayr und Tom W. Drummond stellen in [RD06a] ein kantenbasiertes Trackingsystem für Outdoor Augmented Reality Anwendungen vor. Betrieben wird dieses System auf einem Handgerät, um Mobilität zu gewährleisten. Um mit schnellen Bewegungen zurecht zu kommen, wird der Tracker durch ein Gyroskop unterstützt. Außerdem wird die Erdanziehungskraft und das Magnetfeld gemessen um „Drift“[1] zu vermeiden. Zusätzlich existieren Referenzbilder um das System rei-

[1] Mit „Drift“ wird das Aufsummieren des Fehlers bezeichnet, der entsteht, wenn das Berechnen der neuen Pose größtenteils auf den Informationen des vorherigen Frames basiert.

Abbildung 3.1: Bilder von Liniendetektion und Matching aus [RD06a]: Die beiden Bilder auf der linken Seite stellen das gerenderte Modell dar, die drei Bilder auf der rechten Seite das mit den Modelllinien überlagerte Videobild.

nitialisieren zu können, wenn die berechnete Pose zu weit von der tatsächlichen Pose abweicht.

Das Trackingmodul detektiert zur Laufzeit Kanten aus einem texturierten 3D-Modell. Dazu wird ein grobes, texturiertes Modell in der Pose des vorherigen Frames gerendert, aus dem „Edgels“(„edge pixel“: ein Pixel, das auf einer Kante im Bild liegt) detektiert und auf das Modell rückprojiziert, um ihre 3D-Koordinaten zu erhalten. Aus diesen „Edgels“ werden im nächsten Schritt Kanten generiert, mit deren Hilfe der Fehler zur aktuellen Pose des Videobildes berechnet wird (siehe Abbildung 3.1). Dieser Fehler wird parametrisiert, um ihn mit den Daten des Sensors (Rotationsgeschwindigkeit, 3D-Beschleunigungsvektor und 3D-Magnetfeldvektor) in einem *extended Kalman Filter* (EKF, siehe [WB95b]) zu vereinigen. Der EKF benutzt ein Modell mit 12 Parametern (6 für die Pose und 6 für die Geschwindigkeit) um den Zustand des Systems zu schätzen.

Da die Sensoren des Systems nur Informationen über die Orientierung und nicht über Translationen liefern, kann es schnell vorkommen, dass der Kantenvergleich falsche Korrespondenzen liefert. Deshalb wurde ein Wiederherstellungsschritt hinzugefügt, der jedes Mal ausgeführt wird, wenn ein statistischer Test nach der Poseschätzung eine zu große Abweichung detektiert. Diese Wiederherstellung basiert auf der Korrespondenzsuche von Punktmerkmalen im aktuellen Kamerabild und einem vorher online während des Trackings aufgenommenen Referenzbild. Schlägt auch dies fehl, so werden die Beschleunigungsdaten des Sensors auf Null zurückgesetzt und der aktuelle Frame wird übersprungen.

Das System wurde von Reitmayr und Drummond anhand von Ground Truth Daten getestet. Dazu wurden zwei verschiedene 3D-Modelle benutzt, die aus wenigen, großen Flächen und einer sehr detaillierten Textur bestehen. Die in [RD06a] beschriebene Ergebnisse zeigen, dass die Pose durchschnittlich ca. eine halben Meter von den Ground Truth Daten abweicht. Reitmayr und Drummond geben jedoch

an, dass diese Abweichung im Kamerabild selber nur minimal zu sehen ist, und somit Überlagerungen einigermaßen akkurat dargestellt werden können. Auch sagen sie, dass das System sich in den Tests größtenteils recht robust verhalten hat. Aussetzer gibt es vor allem, wenn die Kamera zu nah an den Objekten ist, und somit der Kantendetektor keine guten Korrespondenzen mehr findet.

Dieser Ansatz ist speziell für Außenaufnahmen entworfen worden, ist abhängig von stark texturierten Objekten und hat den großen Vorteil, dass das Modell aus nur wenigen Polygonen besteht. Für wenig texturierte Objekte hingegen, wie es in industriellen Anwendungsfällen oft vorkommt, kann dieses Verfahren nicht gut verwendet werden.

3.1.2 Wuest und Stricker

Harald Wuest und Didier Stricker beschreiben in [WS06] ein Trackingsystem, das speziell für industrielle Anwendungen konzipiert ist und auf der Extraktion von Konturen aus einem einfachen, nicht texturierten 3D-Modell basiert.

Um die Kontur des Modells zu extrahieren, wird eine Kantenkarte (engl. „edge map“) aus dem in der vorhergesagten[2] Pose gerenderten Modell generiert. Dazu wird aus dem z-Buffer mit Hilfe eines Differentialoperators zweiter Ordnung ein Kantenbild erstellt, das anschließend mit einem Canny-Artigen Extraktionsalgorithmus abgetastet wird, um 3D-Konturen des Objekts zu erhalten. Aus diesen Konturen werden in einem weiteren Schritt Kontrollpunkte und deren Orientierungen berechnet, so dass ein parametrisiertes Linienmodell des Objekts entsteht. Im anschließenden Trackingschritt wird zu jedem Kontrollpunkt im Modell der korrespondierende Punkt im Kamerabild gesucht. Hierzu wird auf einer orthogonalen Suchlinie mit Hilfe von Gradientenmasken nach dem Gradientenmaximum im Kamerabild gesucht (siehe Abbildung 3.2). Um dem Kontrollpunkt kein falsches Maximum zuzuweisen, werden an diesem Punkt mehrere Hypothesen aufrecht erhalten, es werden also mehrere mögliche Korrespondenzen gespeichert.

Auf Basis dieser Art von „Linienkorrespondenzen“ wird eine Minimierung des Rückprojektionsfehlers mit dem „Tukey-Schätzer“ durchgeführt. Da im vorherigen Schritt mehrere mögliche Korrespondenzen zu einem Kontrollpunkt gespeichert wurden, wird genau der Punkt gewählt, der dem auf die Bildebene projizierten Kontrollpunkt am nächsten liegt.

[2] Die Vorhersage basiert hier auf den Linienmerkmalen der vorherigen Pose.

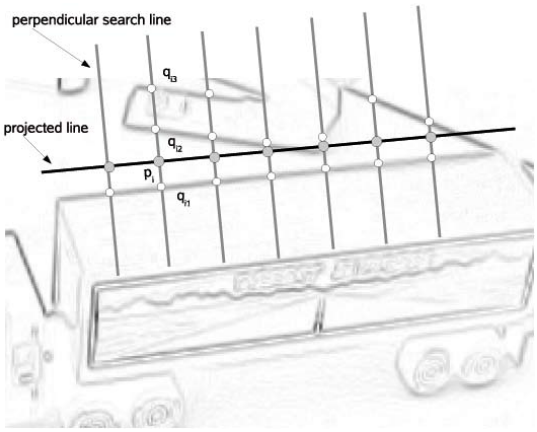


Abbildung 3.2: Korrespondenzsuche an bestimmten Kontrollpunkten einer Linie aus [WS06].

Wuest und Stricker haben Tests einerseits mit einer virtuellen Kamerafahrt und andererseits mit realen Bildserien durchgeführt. Alle Tests in [WS06] haben gezeigt, dass das Trackingsystem bei langsamen Kamerabewegungen recht robust und genau arbeitet, jedoch bei schnelleren Bewegungen keine korrekten Ergebnisse mehr erzielen kann. Die Genauigkeit und der Rechenaufwand hängen stark von der Auflösung des Videobildes ab. Eine größere Auflösung erhöht die Genauigkeit, wobei jedoch die Rechenzeit gleichzeitig ansteigt. Es muss also mit sinnvoll ausgewählten Schwellwerten (z. B. Anzahl der Kontrollpunkte) ein guter Kompromiss zwischen Genauigkeit und Rechenaufwand gefunden werden.

Dieses Verfahren ist speziell für schwach texturierte Objekte und Objekte mit harten Kanten entworfen worden, was es gut in industriellen Aufgaben einsetzbar macht. Vorteile dieses Ansatzes sind, dass durch die ausschließliche Verwendung des Modells zur Poseschätzung kein Drift entstehen kann und keine Vorverarbeitungsschritte (wie z. B. die Erstellung von Referenzbildern) von Nöten sind. Das System kann jedoch laut [WS06] bei detaillierterer Texturierung nicht mehr verwendet werden, da zu viele Kanten entstehen, so dass keine eindeutigen Korrespondenzen mehr gefunden werden können.

3.1.3 Lepetit et al.

Vincent Lepetit, Luca Vacchetti, Daniel Thalmann und Pascal Fua stellen in [LVTF03] ein Trackingsystem vor, das auf einem groben Modell der Szene, Refe-

Abbildung 3.3: Einige der zu einem bestimmten 3D-Punkt gespeicherte Ansichten aus [LVTF03].

renzbildern und den Informationen des vorherigen Frames basiert. Laut den Autoren werden somit „Drift“ und „Jitter“[3] eliminiert.

Das System beginnt mit einem offline-Schritt, in dem Referenzbilder der Szene mit Hilfe eines 3D-Modells der Szene generiert werden. Zu jedem dieser Referenzbilder werden 2D/3D-Korrespondenzen, das Bild der Nachbarschaft jeder Korrespondenz mit der entsprechenden Oberflächennormalen im Modell und die jeweilige Projektionsmatrix $P_K = A_K[R_K|t_K]$ (A_K ist hier die intrinsische Kameramatrix) gespeichert. Aus der Nachbarschaft jedes Merkmals wird ein Deskriptor berechnet, der auf Eigenbildberechnungen basiert. Dazu wird zuerst jedes Patch in verschiedenen Ansichten „gerendert“ (siehe Abbildung 3.3) (die Pixel werden mit Hilfe einer Homographie so verschoben, dass sie dem gewünschten Blickwinkel entsprechen), um dann daraus einen Eigenvektor zu jedem 3D-Merkmal zu generieren. Mit Hilfe der Hauptkomponentenanalyse wird dieser Vektor so effizient gestaltet, dass ein einfacher Vergleich zwischen zwei Deskriptoren möglich wird. In der Initialisierungsphase werden anschließend die Merkmale des Videobildes mit den gespeicherten 3D-Merkmalen gematcht und aus diesen 2D/3D-Korrespondenzen mit POSIT (siehe 2.7.1) und RANSAC (siehe 2.6.1) die Ausgangspose bestimmt.

In der online Trackingphase werden in jedem Frame Features mit dem Harris Corner Detector (siehe 2.2.2) detektiert und mit den Daten eines Keyframes gematcht. Der passende Keyframe wird mit Hilfe des aktuell gerenderten 3D-Modells und dessen Facet-Normalen ausgewählt. Die neue Pose der Kamera wird anhand der Minimierung des Rückprojektionsfehlers unter Verwendung des „Tukey-Schätzers“ (siehe 2.6.2) bestimmt. Da jedoch bei der ausschließlichen Verwendung von Keyframes Jitter auftreten kann, wird die Poseschätzung zusätzlich durch Informationen aus dem letzen Frame stabilisiert.

[3] Mit „Jitter“ wird das Springen des Bildes (bzw. der Pose) von einem Frame zum anderen bezeichnet.

Lepetit et al. geben in [LVTF03] an, dass ihr System aufgrund der Kombination von online- und offline-Informationen sehr genau, robust, jitter- und driftfrei ist sowie mit 15 Frames/Sekunde fast echtzeitfähig ist. Auch Verdeckungen und kleine Ungenauigkeiten im Modell beeinflussen das Ergebnis nicht. Von Nachteil ist jedoch, dass vor dem Trackingprozess Referenzbilder gemacht werden müssen. In den Tests wurde ein relativ kleines Objekt benutzt, das mit 10 Referenzbildern initialisiert wurde. Sollte das System jedoch für Outdoor-Anwendungen eingesetzt werden, kann die Anzahl der vorher aufzunehmenden Bilder immens steigen, was einen starken Vorlauf produziert.

3.2 Eigener Ansatz

Nachdem im vorherigen Kapitel ein Überblick über bestehende Trackingansätze gegeben wurde, folgt in diesem Kapitel eine Beschreibung des dieser Diplomarbeit zugrunde liegenden Systems.

Die Implementierung des Trackingsystems basiert auf der *Open Computer Vision Library* (OpenCV) sowie auf *OpenGL* und *GLU/GLUT* bzw. *GLUI*. OpenCV ist eine Open Source Bildverarbeitungsbibliothek von Intel [Int], die unter Anderem Funktionen zum Einlesen von Videos, Motion Tracking, zur Segmentierung und zur Struktur aus Bewegung bereitstellt.

3.2.1 Verwendete Komponenten

Die in dieser Diplomarbeit benutzen Punktdetektoren bestehen zum größten Teil aus Fremdcode. So wurden die OpenCV-Implementierungen des „Harris Corner Detectors“ und des „Kanade-Lucas-Tomasi-Detectors“ benutzt. Die Implementierung des „SIFT-Operators“ stammt von Rob Hess [Hes]. Der Code des „SUSAN“- und des „FAST“-Detektors werden auf den Homepages der jeweiligen Autoren Stephen Smith [Smi] und Edward Rosten [Ros] bereitgestellt und wurden für die vorliegende Diplomarbeit lediglich im Hinblick auf die Verwendung mit OpenCV angepasst. Der „Förstner-Operator“ wurde nach den Vorschriften in [FG87] neu implementiert.

Das Matching wird (mit Ausnahme des „SIFT-Operators“, der auf Basis des „SIFT-Deskriptors“ arbeitet) mit der „normalisierten Kreuzkorrelation“ (NCC, siehe Kapitel 2.4.2) durchgeführt, wobei sich der verwendete Deskriptor auf die mit einer Gaußmaske gewichtete Pixelnachbarschaft des zu untersuchenden Merkmals beschränkt.

Abbildung 3.4: Anhand der Matchingergebnisse des SIFT Operators ist gut zu sehen, dass die Beschränkung des Suchraums für mögliche Matches eine starke Verbesserung der Punktkorrespondenzen zur Folge hat. Im oberen Bild wurde keine Beschränkung des Suchraums vorgenommen. Im mittleren Bild wurde der Suchraum auf ein Fenster von 50×50 Pixel festgelegt und im unteren Bild wurde zusätzlich noch RANSAC verwendet, um das Ergebnis weiter zu verbessern.

Die Größe der Nachbarschaft ist hierbei frei wählbar, jedoch sollte darauf geachtet werden, dass diese nicht zu groß und nicht zu klein gewählt wird. Eine große Nachbarschaft kann zwar stabilere Ergebnisse erzielen, verlangsamt jedoch den Algorithmus zunehmend. Aus diesem Grund sollte das zu betrachtende Fenster eine Größe von 9×9 oder 11×11 besitzen.

Außerdem ist es sinnvoll, nicht alle gefundenen Merkmale in beiden Bildern miteinander zu vergleichen, da ein großer Suchraum den Rechenaufwand der NCC und auch des SIFT-Operators rasant ansteigen lässt. Im Trackingkontext kann sich die geringe Veränderung des Bildinhalts zu Nutze gemacht werden. So ist es bei einer

Breite von 720 und einer Höhe von 540 Pixeln völlig ausreichend nur in einer Umgebung von 30×30 Pixeln nach möglichen Korrespondenzen zu suchen. Ohne diese Einschränkung würde jedes Merkmal im Modellbild mit jedem Merkmal im Videobild verglichen werden. Vor allem bei der Verwendung eines Gebäudes mit vielen ähnlich aussehenden Fenstern hat sich gezeigt, dass die Einschränkung noch einen weiteren Vorteil hat, denn so kommt es seltener zu falschen Punktkorrespondenzen (siehe 3.4). Zusätzlich profitiert die Rechenzeit des RANSAC-Algorithmus enorm von einer geringeren Anzahl an schlecht gematchten Punktkorrespondenzen, denn je weniger Ausreißer vorhanden sind, desto weniger Iterationen werden gebraucht, um diese zu eliminieren.

Die Berechnung der Pose wird mit Hilfe eines robusten „M-Schätzers“ berechnet, wobei sich der „Tukey-Schätzer“ (siehe Kapitel 2.6.2) als besonders geeignet herausgestellt hat (siehe [Ble04], [Ewe06], [LVTF03], [WS06] und [VLF04]). Die Minimierung der Tukey-Funktion wurde mit Hilfe des „Downhill-Simplex“-Verfahrens durchgeführt.

Das verwendete Modell liegt im Wavefront .obj Format mitsamt Textur vor und stellt das B-Gebäude der Universität Koblenz dar. Um das Modell in OpenGL zu laden, wurde ein „Objekt-Loader“ für *.obj* Dateien implementiert, der die .obj-Textdatei parst und aus den so ausgelesenen Informationen über Vertex-, Facet- und Texturekoordinaten eine OpenGL-Displayliste erstellt. Zusätzlich steht für dieses Modell noch eine Textur zur Verfügung, die mit Hilfe von OpenGL-Funktionen auf dem Modell dargestellt wird.

3.2.2 Veränderbare Parameter

Um den Mehrwert der Computergraphik in einem Trackingsystem zu untersuchen, wurde im Trackingsystem dieser Diplomarbeit ein in OpenGL gerendertes 3D-Modell der zu trackenden Szene verwendet. Dieses Modell kann mit verschiedenen Renderingparametern aufgerufen werden, um letztendlich zu testen, welche Parameter mit welchen Featuredetektoren gute Ergebnisse erzielen. Implementiert wurden folgende Variationen:

- mit/ohne Textur
- veränderbare Position der Lichtquelle
- veränderbare ambiente und diffuse Terme der Lichtquelle

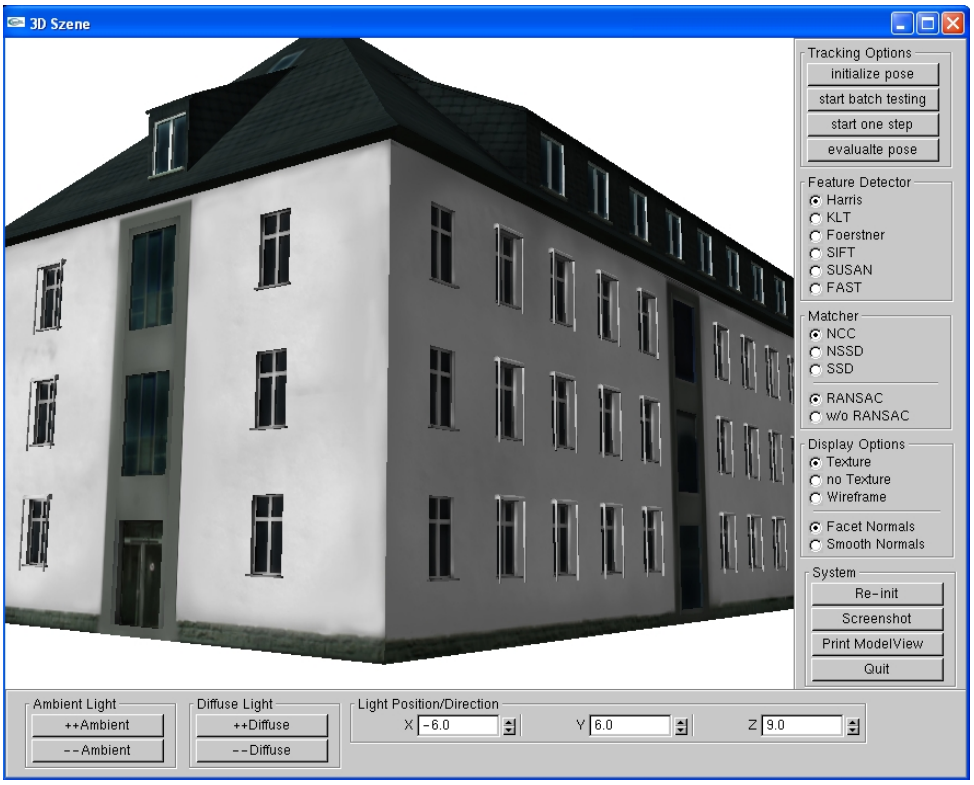


Abbildung 3.5: Screenshot der GUI.

- facet/interpolierte Vertex-Normalen[4]

Außerdem kann im System dynamisch ausgewählt werden, welcher Featuredetektor mit welchem Matcher das Tracking übernehmen soll und ob dies mit oder ohne RANSAC durchgeführt werden soll (siehe Abbildung 3.5).

3.2.3 Trackingablauf

Dieses Kapitel gibt einen Überblick über den Aufbau des dieser Diplomarbeit zugrunde liegenden Trackingsystems. Außerdem wird beschrieben, wie das Prinzip der „Analyse durch Synthese“ zur Unterstützung des Trackingablaufs umgesetzt wurde.

Abbildung 3.6 gibt den groben Ablauf des Systems wider. Die Ausgangssituation

[4]Im Gegensatz zu Facet-Normalen, die orthogonal zum Dreieck stehen, werden bei den interpolierten Normalen alle angrenzenden Facet-Normalen interpoliert, so dass weichere Übergänge an Kanten entstehen

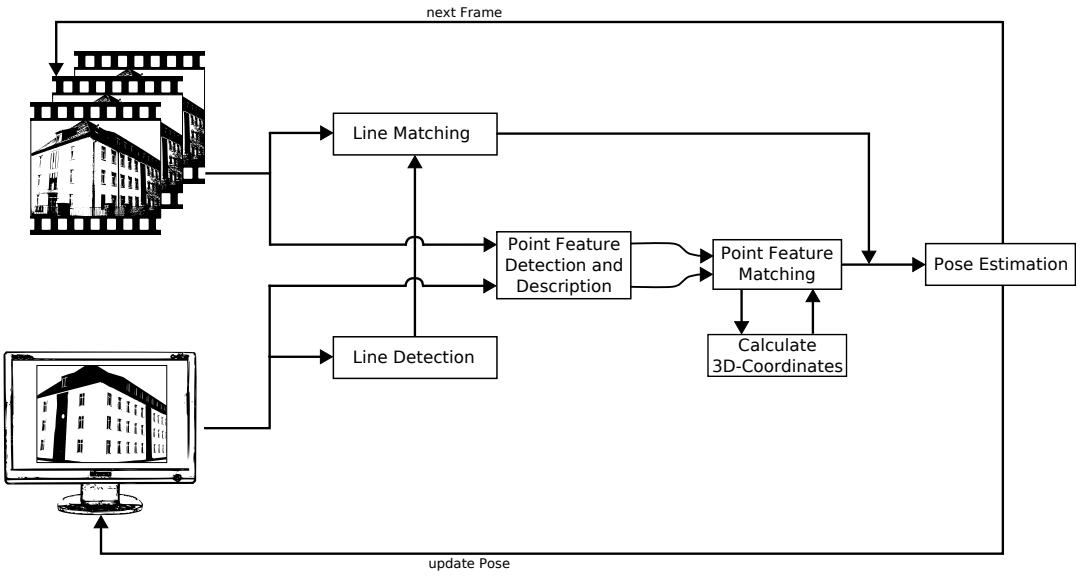


Abbildung 3.6: Übersicht des im Rahmen dieser Diplomarbeit erstellten Trackingsystems: Im Modellbild werden Linien detektiert, die mit Hilfe des in Kapitel 2.5.3 beschriebenen Verfahrens mit dem Videobild gematcht werden. Aus beiden Bildern werden in einem weiteren Schritt Punktmerkmale extrahiert und gematcht. Für die Punktmerkmale des Modellbildes werden ihre 3D-Koordinaten bestimmt. Aus den Linien- und Punktkorrespondenzen wird die neue Pose geschätzt, die schließlich im gerenderten Modell aktualisiert und der nächste Frame bearbeitet wird.

ist, dass auf der einen Seite Frames des Videobildes nacheinander eingelesen werden und auf der anderen Seite das in der aktuellen Pose gerenderte Modell vorliegt. Im ersten Verarbeitungsschritt wird das GLUT-Fenster mit Hilfe der Funktion *glReadPixels* eingelesen und als *IplImage* (eine von OpenCV bereitgestellte Datenstruktur für Bilder) gespeichert.

Soll die Poseberechnung mit der Detektion von Linien unterstützt werden, wird zuerst mit dem Canny-Operator *cvCanny* aus dem Modellbild ein Kantenbild generiert, woraufhin die Hough-Transformation (*cvHoughLines2*) markante Linien extrahiert. Diese Linien werden, mit Start- und Endpunkt definiert, in einem STL-Vector *vector<Line>* gespeichert, nacheinander mit dem in Kapitel 2.5.3 beschriebenen Verfahren mit möglichen Korrespondenzen im aktuellen Frame der Videosequenz gematcht und in einem STL-Vector *vector<LineMatch>* gespeichert.

Im nächsten Schritt werden auf dem Modellbild sowie auf dem Videoframe Merk-

male detektiert und gematcht. Zu diesem Zeitpunkt sind die Korrespondenzen jedoch noch nicht fehlerfrei genug, um direkt mit der Poseschätzung zu beginnen, weshalb zuerst mit dem RANSAC-Algorithmus (siehe Kapitel 2.6.1) Ausreißer eliminiert werden müssen. Dieser berechnet zur Validierung der Korrespondenzen Homographien mit Hilfe der „Direkten Linearen Transformation“ (DLT) und minimiert den Fehler zwischen den Merkmalspaaren, so dass nur noch möglichst gute Korrespondenzen zur anschließenden Poseschätzung übrig bleiben.

Um 2D/3D-Korrespondenzen zu erhalten, werden die 2D-Merkmale des Modellbildes mit Hilfe der OpenGL-Funktion *glReadPixels* und der GLU-Funktion *gluUnProject* in 3D-Merkmale im Weltkoordinatensystem umgewandelt. Dabei ist zu beachten, dass es manchmal vorkommen kann, dass im Modellbild detektierte Merkmale die nah am Rand des Objekts liegen von der Funktion *glReadPixels* nicht auf das Objekt, sondern auf die „far plane“ projiziert werden und somit ein z-Buffer-Wert von 1.0 erhalten. Da dies starke Ausreißer produzieren würde, werden alle Ergebnisse dieser Art aus der Menge der Merkmalspaare entfernt. Dies ist legitim, weil die „far plane“ in ausreichender Entfernung zum hinteren Teil des Objekts platziert wurde.

Die im vorherigen Schritt berechneten 2D/3D-Korrespondenzen werden nun benutzt, um die Pose der Kamera im aktuellen Videoframe mit Hilfe eines M-Schätzers zu approximieren. Alle Methoden zur Poseschätzung (außer des „Downhill-Simplex-Verfahrens“, das in einer gesonderten Klasse implementiert wurde) sind in einer Klasse *Pose.cpp* gekapselt. Als M-Schätzer wird der „Tukey-Schätzer“ verwendet (siehe Kapitel 2.6.2), da dieser sich auch in anderen Trackingsystemen als robust herausgestellt hat (siehe [Ble04], [Ewe06], [LVTF03], [WS06] und [VLF04]).

Zur Minimierung der Tukey-Funktion wird in diesem System das „Downhill-Simplex-Verfahren“ eingesetzt. Es kann jedoch theoretisch auch jedes andere nichtlineare Optimierungsverfahren (siehe Kapitel 2.7.2) verwendet werden. Für die Schätzung der Kamerapose muss die extrinsische Kameramatrix $E = [R|t]$ in eine möglichst kleine Anzahl von Parametern zerlegt werden. Aufgrund der geringeren Anzahl an Parametern wurden zur Parametrisierung der Rotationsmatrix die Eulerwinkel den Quaternionen vorgezogen[5]. Dieser sechsdimensionale Parametervektor (dre Parameter für die Translation und drei für die Rotation) dient dem „Downhill-Simplex-Verfahren“ als Eingabe. Außerdem benötigt das „Downhill-Simplex-Verfahren“ eine Start-Simplex, deren Definition kritisch für die Suche nach dem globalen Minimum ist. So bleibt der Algorithmus z. B. bei einer zu kleinen

[5]Implementiert wurde zu Testzwecken auch eine Poseschätzung mit Quaternionen, die jedoch keine sichtbaren Vorteile erzielen konnte.

Start-Simplex möglicherweise in einem lokalen Minimum hängen, weshalb in diesem Trackingsystem mit mehreren Start-Simpizes gearbeitet wird.

In jeder Iteration der Poseschätzung wird mehrmals der Rückprojektionsfehler zwischen den auf die Bildebene projizierten 3D-Merkmalen und den 2D-Merkmalen des Videobildes bestimmt. Um den Rückprojektionsfehler zu berechnen wird zuerst aus dem vom „Downhill-Simplex-Verfahren“ neu bestimmten Parametervektor eine extrinsische Kameramatrix E (mit den Eulerwinkeln r_x, r_y, r_z und dem Translationsvektor $(t_x, t_y, t_z)^T$) und anschließend eine Modelview-Matrix erstellt:

$$E = \begin{pmatrix} \cos r_y \cos r_z & \cos r_x \sin r_z & -\sin r_y & t_x \\ \sin r_x \sin r_y \cos r_z - \cos r_y \sin r_z & \sin r_x \sin r_y \sin r_z + \cos r_x \cos r_z & -\sin r_x \cos r_y & t_y \\ \cos r_x \sin r_y \cos r_z + \sin r_x \sin r_z & \cos r_x \sin r_y \sin r_z - \sin r_x \cos r_z & -\cos r_x \cos r_y & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Aus dieser Matrix E wird, wie in Gleichung 3.2 beschrieben, eine OpenGL-Modelview-Matrix berechnet. Anschließend wird die GLU-Funktion *gluProject* verwendet, um die 3D-Merkmale des Modells auf die Bildebene zu projizieren. Aus der Differenz der projizierten 3D-Merkmale und ihrer Korrespondenzen im Videoframe wird der Rückprojektionsfehler aufsummiert und letztendlich minimiert.

Mit dieser approximierten Pose wird in einem letzten Schritt die Kamera (siehe Abschnitt „Die Kamera in OpenGL“ 3.2.3) aktualisiert und das Modell neu gerendert, so dass der nächste Videoframe bearbeitet werden kann.

Die Kamera in OpenGL

Bevor mit dem Tracking begonnen wird, ist es wichtig, dass das Modell möglichst realistisch in OpenGL gerendert werden kann. Deshalb sollte die Projection-Matrix den tatsächlichen intrinsischen Kameraparametern der Videokamera entsprechen. Um diese zu ermitteln, wurde in einem Vorverarbeitungsschritt mit der von OpenCV bereitgestellten Methode *cvCalibrateCamera* die Brennweite in x- und y-Richtung (f_x und f_y) und der Hauptpunkt (p_x, p_y) der Kamera bestimmt. Aus diesen Parametern kann eine OpenGL Projection-Matrix wie folgt erstellt werden:

$$GL\_PROJECTION = \begin{pmatrix} 2\frac{f_x}{w} & 0 & -2\frac{p_x}{w}+1 & 0 \\ 0 & 2\frac{f_y}{h} & 2\frac{p_y}{h}-1 & 0 \\ 0 & 0 & \frac{far+near}{near-far} & 2\frac{far \cdot near}{near-far} \\ 0 & 0 & -1 & 0 \end{pmatrix} \tag{3.1}$$

wobei *far* und *near* die vordere bzw. die hintere Clipping Plane in OpenGL darstellen. Diese Festsetzung der intrinsischen Kameraparameter hat zur Folge, dass die Aufnahmen der Videokamera ohne den Einsatz eines Zooms gemacht werden sollten, da sich durch die Veränderung des Zoomfaktors die Brennweite verändern würde.

Die Modelview-Matrix ist eine 4×4 Matrix, die aus der Rotation R und Translation t der Szene besteht:

$$GL\_MODELVIEW = \begin{pmatrix} R & t \\ 0^T & 1 \end{pmatrix} \tag{3.2}$$

Um die Modelview-Matrix in die Pose der Kamera $P_K = [R_K|t_K]$ zu transformieren, muss die Rotationsmatrix R transliert sowie der Translationsvektor t mit der translierten, negierten Rotationsmatrix multipliziert werden:

$$R_K = R^t \tag{3.3}$$

$$t_K = -R^T t \tag{3.4}$$

Da in OpenGL keine tatsächliche Kamera existiert[6], wurde eine Kameraklasse *Camera.cpp* implementiert, die den Status der virtuellen Kamera aufrechterhalten kann. Die Kameraklasse speichert die aktuellen Projection- und Modelview-Matrizen. Außerdem stellt sie Methoden zur Rotation und Translation der Kamera sowie Methoden zur Berechnung der aktuellen Kamerapose aus der Modelview-Matrix zur Verfügung.

[6]Die „Kamera“ in OpenGL liegt immer im Ursprung des Weltkoordinatensystems und die Szene wird mit der Modelview-Matrix rotiert und transliert.

Initialisierung

Das zu dieser Diplomarbeit implementierte Trackingsystem arbeitet allein auf Basis eines 3D-Modells der Szene, weshalb außer der Ermittlung der intrinsischen Kameraparameter keine spezielle Vorbereitung (wie z. B. die Erstellung von Referenzbildern) nötig ist.

Der erste Schritt der Initialisierung des Systems besteht darin, das Modell der Szene wie in Kapitel 3.2.1 beschrieben in OpenGL zu laden. Der zweite Schritt umfasst das Einlesen des Videos mit Hilfe der OpenCV Methode *cvCaptureFromAVI* und dessen Speicherung als eine Liste von einzelnen Frames. Alternativ könnte das System auch mit einer angeschlossenen Kamera arbeiten, was jedoch aufgrund der fehlenden Echtzeitfähigkeit in dieser Diplomarbeit nicht weiter verfolgt wurde.

Als Initialpose der Kamera wird eine per Augenmaß abgeschätzte, ungefähre Position des Modells benutzt. Möglich ist auch die initiale Berechnung der Pose mit dem in OpenCV implementierten „POSIT"-Algorithmus *cvPosit* (siehe Kapitel 2.7.1), der aber sehr stark von der Güte und Verteilung der Punktkorrespondenzen im Bild abhängig ist und nicht immer gute Ergebnisse erzielt. Da diese Diplomarbeit jedoch ihren Schwerpunkt auf der Untersuchung der verschiedenen Punktdetektoren und der veränderbaren Renderingparameter legt, ist eine ungefähre Abschätzung für diese Versuchsbedingungen ausreichend und vor allem auch besser für Vergleiche geeignet.

Umsetzung des Prinzips der „Analyse durch Synthese"

Wie bereits in Kapitel 2.1 beschrieben, bedeutet „Analyse durch Synthese", dass ein synthetisches Signal an ein reales Signal angenähert wird, um eine approximierte Lösung für das reale Signal zu finden. In der Implementierung eines Trackingsystems bedeutet dies, dass ein synthetisches Modell der zu trackenden Szene so gerendert wird, dass die Pose der virtuellen Kamera der Pose der realen Kamera möglichst genau entspricht.

In dem dieser Diplomarbeit zugrunde liegenden Trackingsystem wurde ein 3D-Modell des B-Gebäudes der Universität Koblenz und dessen Kamerapose in OpenGL als synthetisches Signal benutzt. Die reale Szene wird also synthetisch simuliert, so dass ein Vergleich zwischen beiden Darstellungen gemacht werden kann, damit die virtuelle Kamera der realen Kamera unter Verwendung von Schätzmethoden angenähert werden kann. Um einen Vergleich zu ermöglichen, muss jedoch beachtet

werden, dass beide Kameras im selben Weltkoordinatensystem definiert sind und die Achsen des Kamerakoordinatensystems identisch ausgerichtet sind.

Der Mehrwert eines realistischen 3D-Modells ist, dass das Videosignal direkt mit diesem verglichen werden kann und somit keine Referenzbilder oder Informationen aus vorherigen Frames nötig sind. Die Vorverarbeitungsschritte beschränken sich lediglich auf das einmalige Erstellen des 3D-Modells und die Bestimmung der intrinsischen Kameraparameter. Außerdem kann mit Hilfe von OpenGL und GLU auf die 3D-Koordinaten eines 2D-Punktes zugegriffen werden, ohne z. B. die Schnittpunkte von Geraden und Dreiecken berechnen zu müssen.

Kapitel 4

Experimente und Ergebnisse

In diesem Kapitel wird das in Abschnitt 3.2 beschriebene System anhand von manuell ausgewählten Referenzpunkten getestet, da keine Ground Truth Daten der Szene vorlagen. In anderen Arbeiten zu Trackingsystemen, wie in [Ble04], wurden virtuelle Kamerafahrten als „Ground-Truth"-Daten benutzt, von denen jedoch in dieser Diplomarbeit abgesehen wurde, da der Unterschied zwischen realem Kamerabild und synthetischer Szene zu groß ist, um aussagekräftige Vergleichsergebnisse zu erhalten. Wichtig beim Vergleich von synthetischen mit realen Bildern ist nämlich vor allem, dass die Featuredetektoren in beiden Bildern möglichst an derselben Stelle Features finden, so dass viele gute Matches gefunden werden können. Welche Detektoren dies unter den verschiedenen Bedingungen am besten realisieren, soll in diesem Kapitel vorgestellt werden.

Um die Güte der Pose vergleichbar zu machen, wurden per Hand auf dem Screenshot der in OpenGL gerenderten Initialpose markante Punkte auf dem Objekt ausgewählt und mit Hilfe der GLU-Funktion *gluUnProject* in 3D-Koordinaten transformiert. In jedem Frame der insgesamt 50 Bilder umfassenden Videosequenz wurden die entsprechenden Objektpunkte ausgewählt und in eine von OpenCV lesbare *YAML*-Datei geschrieben. Anhand dieser manuell erstellten Punktkorrespondenzen werden in den folgenden Tests die errechneten Posen der verschiedenen Featuredetektoren mit unterschiedlichen Matchingverfahren und Renderingparametern miteinander verglichen. Als Fehlermaß dient der RMSE[1] des Rückprojektionsfehlers. Diese pro Frame berechneten Werte werden anschließend für einen gesamten Wert pro Videosequenz aufsummiert und der Mittelwert berechnet. Jede Testreihe wurde

[1] *Root Mean Square Error:* $\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}$

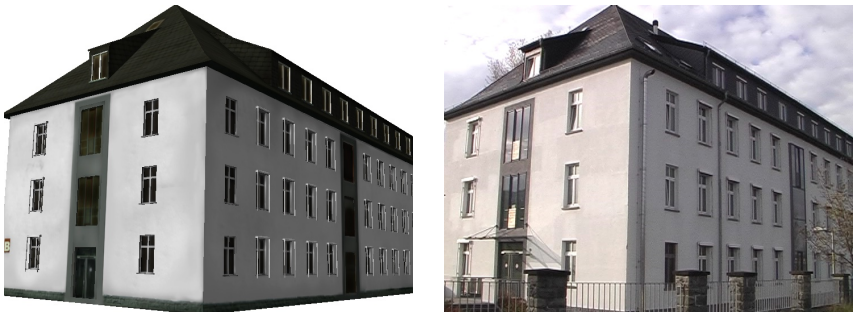


Abbildung 4.1: Vergleich des Modellbildes mit dem realen Videobild: die Geometrie ist nicht exakt nachgebildet worden.

mehrmals wiederholt, da vor allem durch den Einsatz von RANSAC ein Zufallsfaktor in die Berechnungen einfließt, der zu variierenden Ergebnissen führt. Zu beachten ist, dass der berechnete Fehler zwar tatsächliche Pixelfehler widerspiegelt, aber durch das nicht ganz geometrisch korrekte Modell sehr hoch ausfällt. Wie in Abbildung 4.1 zu erkennen ist, sind die Fenster an der rechten Wand des Gebäudes zu weit vorne, und auch an der Front stimmen die Abstände nicht ganz überein.

Die implementierte Liniendetektion wird hier nicht getestet, da sie zu diesem Zeitpunkt die Poseschätzung eher verschlechtert als verbessert. Das Problem hierbei war, dass aufgrund der teilweise falschen Geometrie des Modells und der zusätzlich starken Ähnlichkeit der Fenster des Gebäudes, keine durchweg guten Linienkorrespondenzen erzielt werden konnten.

Da die Echtzeitfähigkeit kein angestrebtes Ziel dieser Diplomarbeit war, wird die Rechenzeit des Trackers hier nicht getestet. Der Fokus lag auf dem Vergleich verschiedener Featuredetektoren und Matchingverfahren im Hinblick auf die Güte der Pose und vor allem darauf, wie die unterschiedlichen Renderingparameter die Poseschätzung positiv bzw. negativ beeinflussen.

Zuerst wird in Abschnitt 4.2 ein Vergleich der implementierten Featuredetektoren und Matcher unter gleich bleibenden Renderingbedingungen durchgeführt. In Abschnitt 4.3 werden die Renderingparameter variiert. Abschließend folgt in Abschnitt 4.4 eine Zusammenfassung der Ergebnisse.

4.1 Vorbereitung

Vor Beginn der Tests, wurden die Featuredetektoren so eingerichtet, dass sie die jeweils besten Ergebnisse erzielen können. Fast jeder Detektor beruht auf Konstanten bzw. Schwellwerten, die abhängig vom Bildinhalt einen großen Einfluss auf die Ergebnisse haben. Wichtig hierbei ist, dass die Detektoren genug Merkmale aus beiden Bildern liefern, so dass genügend Matches gebildet werden können. Es hat sich einerseits gezeigt, dass lediglich 10-20% der Merkmale durch die Matcher als Korrespondenzen ausgewählt werden und andererseits, dass die geschätzte Pose mit steigender Anzahl an Korrespondenzen besser wird.

Der FAST- und der SUSAN-Detektor waren besonders schwer einzustellen. Während z. B. der Harris Corner Detector bei nur jeweils 800 gefundenen Features in den Bildern durchschnittlich 125 Korrespondenzen liefert, können mit FAST bei 830 Merkmalen im Modellbild und 1000 im Videobild lediglich ca. 50 Korrespondenzen erstellt werden. Um mit FAST auf 100 Korrespondenzen zu kommen, müssen mindestens jeweils 1200 und 1400 Merkmale detektiert werden, was wiederum mehr Rechenzeit durch die NCC bzw. NSSD in Anspruch nimmt. Dies legt die Vermutung nahe, dass FAST keine gute Modellbild-Videobild-Wiederholbarkeitsrate besitzt, Features also nicht an denselben Stellen in beiden Bildern gefunden werden. Bei dem SUSAN-Detektor stellt sich das Problem der zu geringen Korrespondenzen ein wenig anders dar, denn es kann nicht wirklich darauf Einfluss genommen werden, wie viele Matches in den einzelnen Bildern gefunden werden. SUSAN findet im Modellbild durchschnittlich nicht einmal die Hälfte der Features, die im Videobild gefunden werden. So resultieren aus 530 und 1200 gefundenen Features lediglich ca. 60 Korrespondenzen. Mit KLT werden zwar ähnlich viele Korrespondenzen gefunden wie mit dem Harris Corner Detektor, jedoch scheinen die Korrespondenzen weniger gut verteilt im Bild zu liegen und an bestimmten Stellen ein wenig gebündelt zu sein.

Wichtig zur Berechnung einer Pose ist vor allem die gute Verteilung der Punktkorrespondenzen im Bild. Werden in einem Teil des Objekts keine oder wenige Korrespondenzen detektiert, reicht die Tiefeninformation nicht aus, um verlässliche Rotationen und Translationen zu berechnen. Während SIFT eine recht gute Verteilung aufweist, ist die der KLT- und SUSAN-Detektoren an manchen Stellen etwas zu sehr gebündelt (siehe Abbildungen 4.3 und 4.4).

Auch die gute Einstellung des RANSAC-Algorithmus hat sich als wichtig herausgestellt. Die Größe des Schwellwerts t^2 (siehe Kapitel 2.6.1) hat einen großen Einfluss

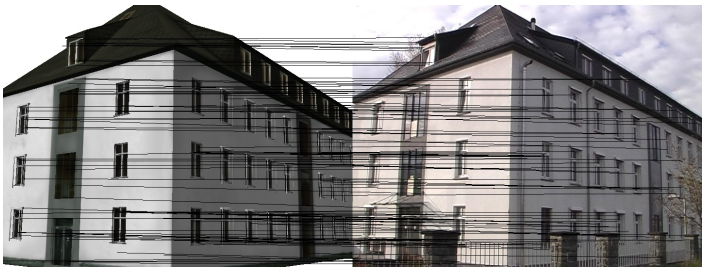


Abbildung 4.2: SIFT-Punktkorrespondenzen

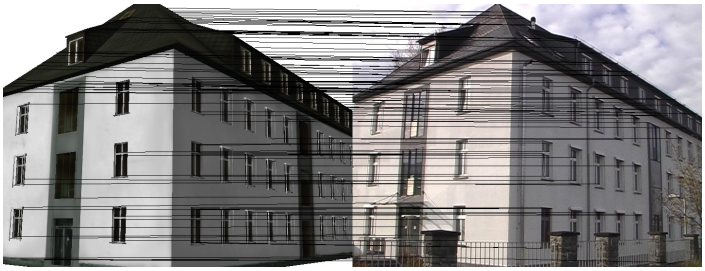


Abbildung 4.3: Harris-Punktkorrespondenzen

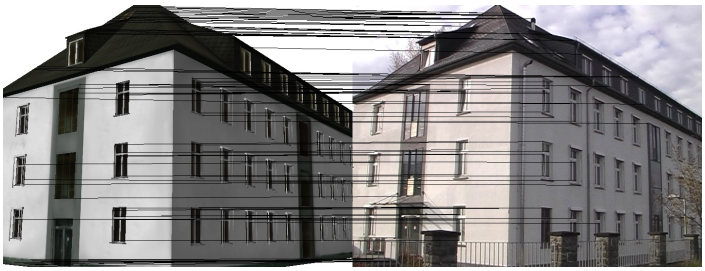


Abbildung 4.4: KLT-Punktkorrespondenzen

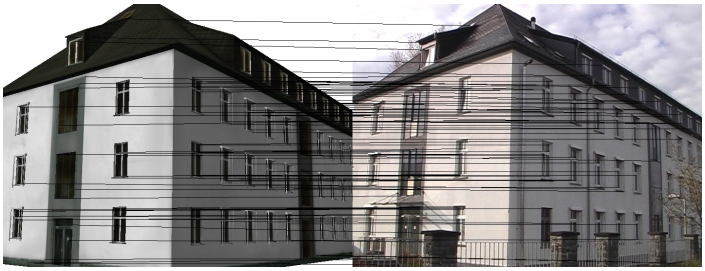


Abbildung 4.5: Förstner-Punktkorrespondenzen

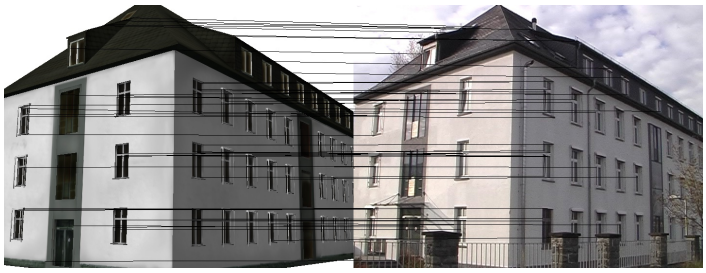


Abbildung 4.6: SUSAN-Punktkorrespondenzen

Abbildung 4.7: FAST-Punktkorrespondenzen

auf die Güte der Pose. Ist dieser zu klein gewählt, so ist die Menge der Inlier zu gering für eine verlässliche Poseschätzung. Bei einem zu großen Wert jedoch sind zu viele schlechte Korrespondenzen in der Menge der Inlier. In [HZ04] wird für Homographien der Schwellwert $t^2 = 5,99\sigma^2$ vorgeschlagen. In den Tests hat sich eine Standardabweichung von $\sigma = 10$ bewährt.

Ein letzter wichtiger Schwellwert ist das Entscheidungskriterium der Matcher, wann ein Merkmalspaar eine gute Korrespondenz darstellt. Die NCC entscheidet dies über den zwischen -1 und 1 liegenden Korrelationswert. Während diese Schwelle beim Vergleich von zwei realen Bildern bei 0.9 liegen kann, muss er im Kontext mit texturierten Modellbildern auf 0.7 und untexturierten Modellbildern sogar auf 0.6 heruntergesetzt werden, damit genügend Korrespondenzen gefunden werden können. Der Schwellwert für die NSSD spiegelt den Fehler zwischen den jeweiligen Pixelnachbarschaften wider und konvergiert gegen 0. In dieser Implementierung wurde dieser Wert für ein gutes Ergebnis auf 10 festgelegt.

4.2 Vergleich der Featuredetektoren

In diesem Abschnitt werden die implementierten Featuredetektoren anhand der resultierenden Pose verglichen. Dazu wurden auch die Matchingverfahren (zum einen die normalisierte Kreuzkorrelation und zum anderen die NSSD) getestet. Als Renderingparameter wurde ein texturiertes, möglichst realistisch beleuchtetes Modell mit interpolierten Vertex-Normalen benutzt.

Außerdem wurden die Tests auch noch ausschließlich auf den ersten 20 Frames der Videosequenz durchgeführt, da bei der Aufnahme in den letzten 25 Frames schnellere Kamerabewegungen gemacht wurden. Schnellere Bewegungen haben nicht nur das Resultat, dass sich die Pose von Frame zu Frame stärker verändert, sondern auch, dass die einzelnen Frames verwaschener sein können.

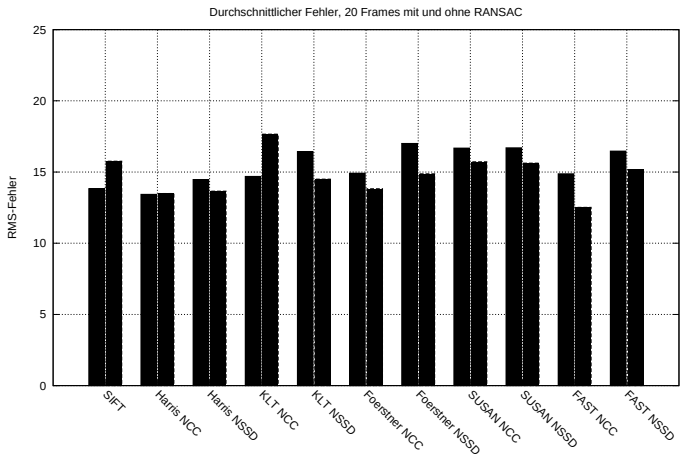


Abbildung 4.8: Vergleich der Ergebnisse mit RANSAC (jeweils linke Balken) und ohne RANSAC (jeweils rechte Balken) für 20 Frames

Abbildungen 4.8 und 4.9 zeigen den Vergleich zwischen Poseschätzungen mit und ohne RANSAC (jeweils für 20 bzw. 50 Frames). Interessant zu sehen ist hierbei, dass bei weniger Frames bzw. langsameren Kamerabewegungen durch den Einsatz von RANSAC keine Vorteile erzielt werden konnten. Bei der längeren Videosequenz jedoch profitieren fast alle Verfahren von der Verwendung von RANSAC. Da die Korrespondenzen durch die Einschränkung des Suchraums schon sehr akkurat sind, kann es sein, dass die geringere Anzahl an Korrespondenzen ausschlaggebend für die etwas schlechteren Ergebnisse bei der kürzeren Sequenz ist. Werden

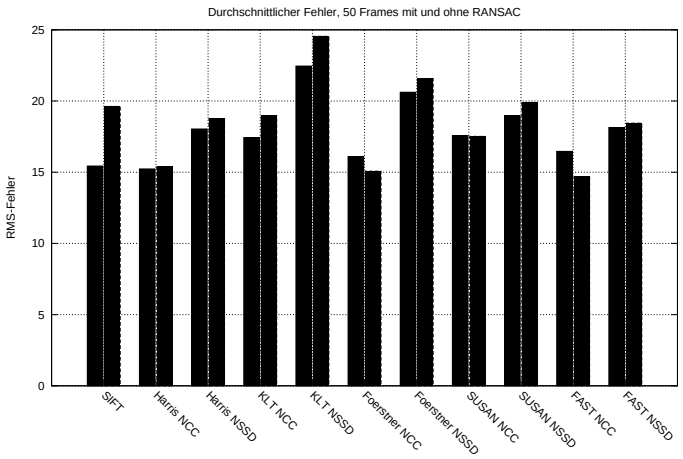


Abbildung 4.9: Vergleich der Ergebnisse mit RANSAC (jeweils linke Balken) und ohne RANSAC (jeweils rechte Balken) für 50 Frames

die Sprünge zwischen den Frames durch eine schnellere Bewegung der Kamera jedoch größer, so kann es dazu kommen, dass schlechtere Korrespondenzen gefunden werden, weil die eigentlich bessere Korrespondenz außerhalb des Suchraums liegt oder sich eine falsche Korrespondenz mit höherer Korrelation im Suchraum befindet. Eine Vergrößerung des Suchraums würde die erste Fehlerquelle zwar ausschließen, aber die zweite noch verstärken, da ein größerer Suchraum immer auch eine größere Gefahr für das Matching darstellt. Je größer der Suchraum gewählt wird, desto wichtiger wird jedoch wieder die Eliminierung der Ausreißer durch den RANSAC-Algorithmus.

In Abbildungen 4.10 und 4.11 sind exemplarisch die Fehler in den einzigen Frames mit RANSAC und NCC bzw. NSSD dargestellt. Während der Fehler beim Matchen mit der NCC nur gering steigt, kommt die NSSD ungefähr ab dem 37. Frame an ihre Grenzen. Auch mit SIFT ist am Ende der Sequenz keine gute Pose mehr zu berechnen. Diese Ergebnisse zeigen, dass das Trackingsystem noch Defizite bei schnellen Kamerabewegungen hat. Dies könnte mit Hilfe einer Reinitialisierung des Systems behoben werden, bei der der Suchraum vergrößert und der Schwellwert t^2 für RANSAC etwas verringert wird.

In Anhang A.1 können detailliertere Vergleiche zwischen den beiden getesteten Matchingverfahren gefunden werden. Diese zeigen, wie ungenau die mit NSSD berechneten Posen im Gegensatz zu den mit NCC berechneten vor allem auf dem letzten

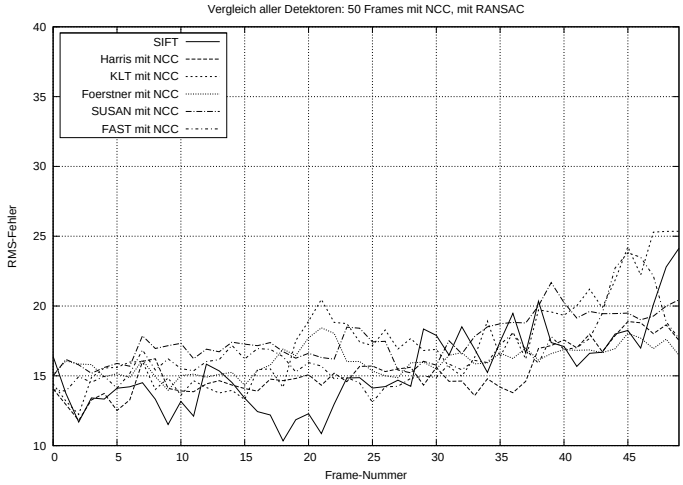


Abbildung 4.10: Fehler pro Frame aller Detektoren mit NCC.

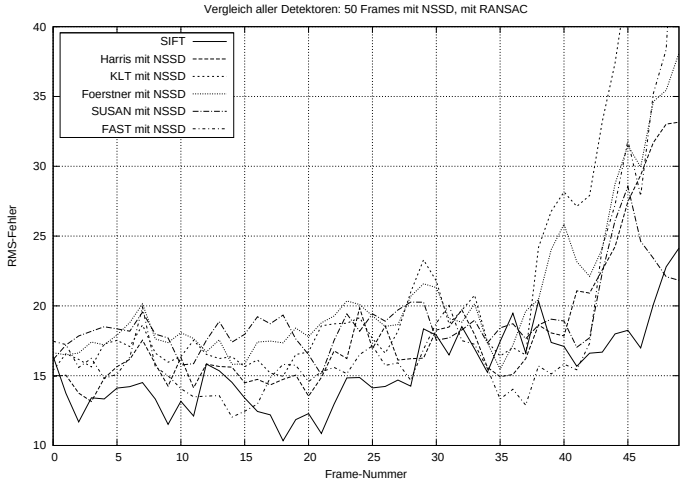


Abbildung 4.11: Fehler pro Frame aller Detektoren mit NSSD.

Drittel der Bildsequenz sind. Jedoch ist die Rechenzeit für die NSSD etwas geringer, so dass es möglicherweise ausreicht, die NCC nur bei schnellen Bewegungen der Kamera zur Reinitialisierung zu verwenden.

Abbildung 4.12: Darstellung aller verwendeter Renderingparameter: Mit Beleuchtung, interpolierten Normalen und Textur (oben links), mit Beleuchtung, Facet-Normalen und Textur (oben Mitte), ohne Beleuchtung, mit interpolierten Normalen und Textur (oben links), mit falscher Beleuchtung, interpolierten Normalen und Textur (unten links) sowie mit Beleuchtung, interpolierten Normalen und ohne Textur (unten rechts)

4.3 Vergleich verschiedener Renderingparameter

In diesem Abschnitt werden die Auswirkungen der verschiedenen in Kapitel 3.2.2 beschriebenen Renderingparameter getestet. Dazu wurden alle Featuredetektoren und als Matchingverfahren die normalisierte Kreuzkorrelation verwendet. Von der Beschränkung dieser Tests auf nur einen Featuredetektor wurde abgesehen, da die Vermutung nahe lag, dass die verschiedenen Detektoren unterschiedlich auf die variierenden Bedingungen reagieren. Aufgrund der Tests aus Kapitel 4.2 wurde jedoch nur noch die normalisierte Kreuzkorrelation als Matchingmetrik verwendet, um die Ergebnisse übersichtlicher zu gestalten.

Normalen

Abbildung 4.12 zeigt die verschiedenen Screenshots der einzelnen Renderingparameter. Es wurden Versuche mit unterschiedlicher Normalenberechnung, Beleuchtung und ohne Textur durchgeführt. Die Unterschiede in der Berechnung der Normalen sind, wie in Abbildung 4.13 zu sehen, eher marginal. Lediglich der KLT-Detektor

scheint ein wenig von weicheren Kanten zu profitieren. Diese Ergebnisse sind kaum überraschend, da das Modell zum größten Teil aus rechtwinkligen Ecken besteht, so dass die interpolierten Normalen nur am Dach und ein wenig an den Fenstern Auswirkungen haben.

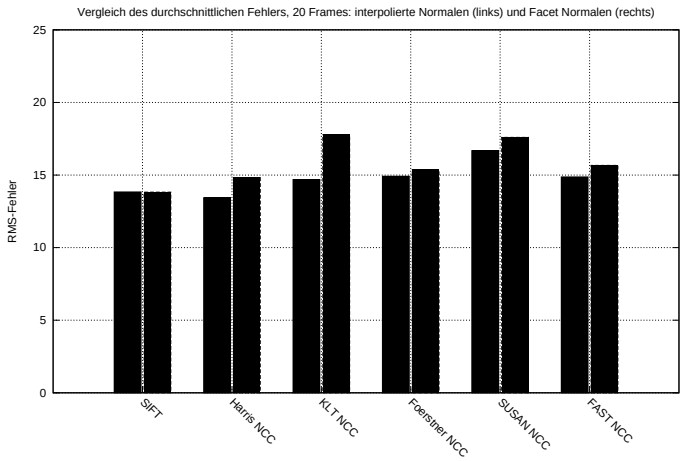


Abbildung 4.13: Vergleich zwischen interpolierten (jeweils linke Balken) und Facet-Normalen (jeweils rechte Balken)

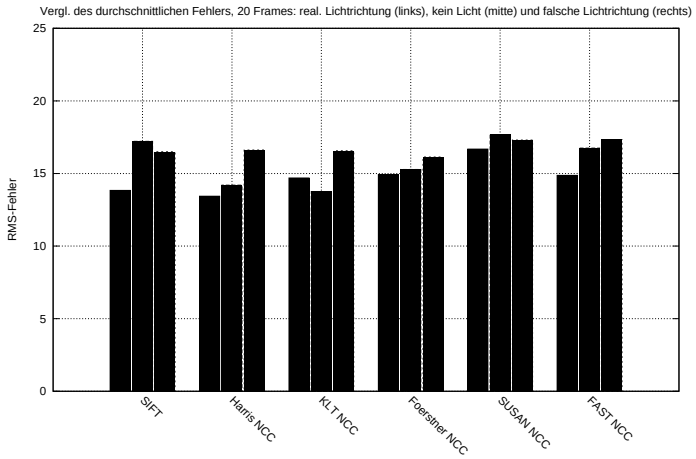


Abbildung 4.14: Vergleich zwischen der Darstellung mit möglchst realen Lichtverhältnissen (jeweils linke Balken) und keinem Licht (jeweils rechte Balken)

Licht

Für den zweiten Test wurde der Vektor der Lichtrichtung auf 0 gesetzt und das ambiente Licht so weit erhöht, dass die Szene nicht insgesamt zu dunkel ist. Die Auswirkung von einem möglichst realistischen Lichteinfall gegenüber keiner Beleuchtung sind jedoch eher gering (siehe Abbildung 4.14). Da die normalisierte Kreuzkorrelation intensitätsinvariant ist, haben unterschiedliche Beleuchtungen kaum einen Einfluss auf das Matchingergebnis. Einzig bei SIFT und FAST ist ein größerer Fehler zu erkennen. Interessant ist auch das Ergebnis für den KLT, der ohne Beleuchtung bessere Ergebnisse zu erzielen scheint.

Es ist weiterhin interessant zu untersuchen, wie sich die Detektoren verhalten, wenn die Lichtrichtung falsch ist. Sollte in einem Trackingsystem die Beleuchtung gerendert werden, so ist es wichtig, auch eine relativ genaue Angabe über Position und Richtung des Lichts zu erhalten. Vor allem bei Außenaufnahmen ist dies kritisch, da sich die Position und Intensität der Sonne während des Trackingablaufs verändern kann. Abbildung 4.14 zeigt, dass eine falsche Lichtposition einen großen Einfluss auf das Trackingergebnis haben kann. Leider wurden an dieser Stelle keine Schattenwürfe untersucht, die sicherlich weitere Bildinformationen für interessante Merkmale liefern könnten. Das Dach des Gebäudes z. B. wirft im Videobild einen Schatten auf die Hauswand (siehe Abbildung 4.15), der im Modell nicht dargestellt werden konnte.

Vor allem aber beim Matching von Linien haben sich die Schattenkanten als ein Problem herausgestellt. Dadurch, dass das Objekt nicht exakt genug modelliert ist, kommt es zu oft vor, dass einige Edgels auf der Schattenkante des Daches anstatt der tatsächlichen Dachkante oder der Regenrinne anstatt der vorderen rechten Ecke des Gebäudes in die Berechnung mit einfließen. Wenn zu viele dieser falschen Edgels vorkommen, dann ist die berechnete korrespondierende Linie so schief, dass die gesamte Poseschätzung falsch wird.

Textur

Der letzte Test beschäftigt sich mit der Frage, ob die Darstellung einer Textur bei der Poseschätzung von Vorteil ist. Abbildung 4.16 zeigt den Vergleich zwischen einem texturierten und einem untexturierten Modell. Bei dem untexturierten Modell mussten die ambienten und diffusen Terme ein wenig niedriger eingestellt werden, da das Modell ansonsten zu hell gewesen wären. Vor allem SIFT, SUSAN und FAST

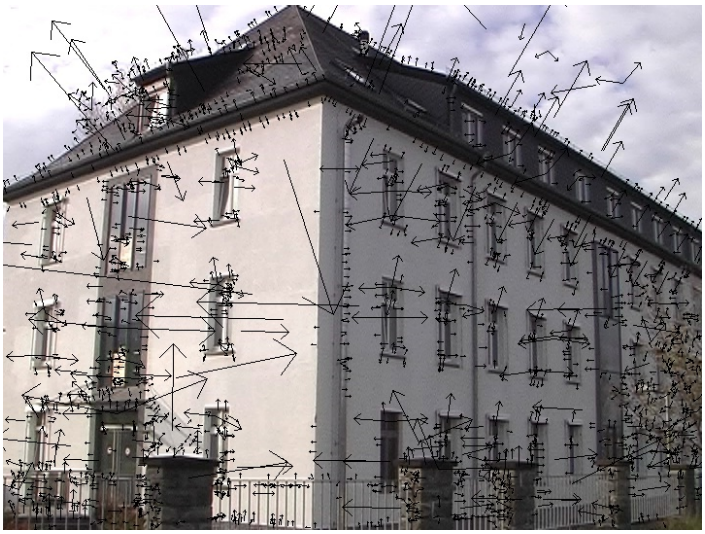


Abbildung 4.15: Am Beispiel des SIFT-Detektors ist zu sehen, dass Schattenkanten wichtige Bildinformationen beinhalten können.

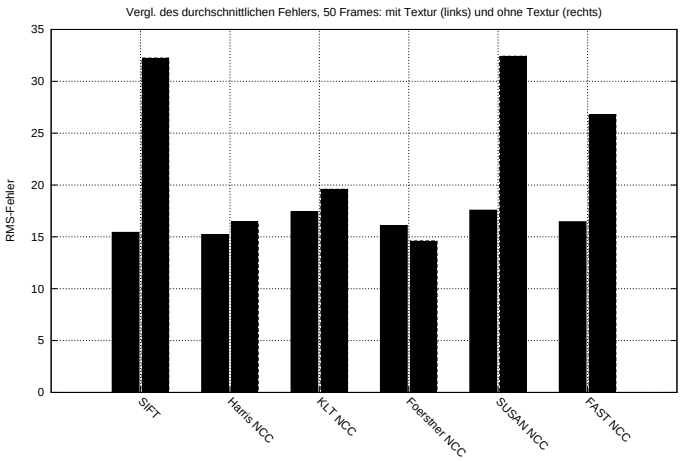


Abbildung 4.16: Vergleich der Detektoren mit/ohne Textur.

liefern ohne Textur viel schlechtere Ergebnisse, wohingegen der Förstner-Operator eine etwas bessere Poseschätzung möglich macht. In Abbildung 4.17 ist noch einmal detaillierter zu sehen, wie der Fehlerverlauf über die gesamten 50 Frames ist. Schon sehr früh werden die Ergebnisse von SIFT und FAST ungenauer und steigen bis zum Schluss in Regionen, von denen sie sich teilweise nicht mehr erholen können. Vor allem bei der Poseschätzung mit Hilfe des FAST-Detektors kann es vorkommen,

dass die im vorherigen Frame berechnete Pose so stark von der realen abweicht, dass keine Korrespondenzen mehr gefunden werden können. Auch hier wäre eine Reinitialisierung von Vorteil.

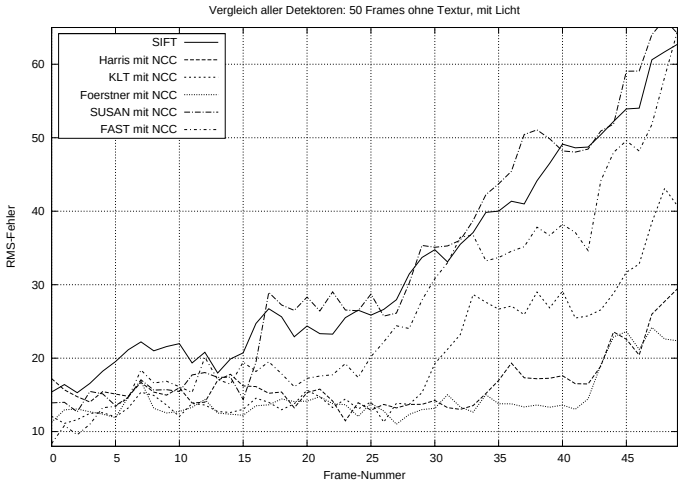


Abbildung 4.17: Der Vergleich der Detektoren auf einem untexturierten Modell zeigt, dass alle außer dem Förstner-Operator und dem Harris Corner Detector zum Ende der Videosequenz keine brauchbaren Ergebnisse mehr liefern.

4.4 Fazit

Das zu der vorliegenden Diplomarbeit implementierte Trackingsystem wurde in diesem Kapitel im Hinblick auf variierende Featuredetektoren, Matchingverfahren sowie Renderingparameter getestet. Dabei wurde der Schwerpunkt, aufgrund der am Anfang dieses Kapitels beschriebenen Probleme mt dem Modell, auf den Vergleich aller veränderbarer Parameter und weniger auf die tatsächliche Güte gelegt.

Im Vergleich der Featuredetektoren untereinander erzielten SIFT mit RANSAC, der Harris Corner Detector (NCC) mit und ohne RANSAC, der Förstner-Operator (NCC) mit und ohne RANSAC sowie der FAST-Detektor (NCC) mit und ohne RANSAC die besten Ergebnisse. Außerdem hat der Test der Matchingmetriken gezeigt, dass die NSSD vor allem bei schnelleren Bewegungen keine verlässlichen Ergebnisse mehr liefert, so dass in solchen Situationen die NCC verwendet werden sollte.

Die Untersuchungen der verschiedenen Renderingparameter haben ergeben, dass die Verwendung einer Textur bei fast allen Detektoren zu einer Verbesserung der Ergeb-

nisse geführt hat. Lediglich der Förstner-Operator kann nicht von einer Textur profitieren. Bei SIFT, FAST und SUSAN hat das Weglassen der Textur eine besonders große Verschlechterung der geschätzten Pose bewirkt.

Die Art der Normalenberechnung hat bei dieser Szene keine nennenswerten Auswirkungen auf die Güte der Pose gehabt. Jedoch ist es vorstellbar, dass durch die Verwendung von interpolierten Normalen bei Objekten mit stumpfwinkligeren Ecken eine Verbesserung zu bemerken ist. Bei solchen Objekten könnten durch Facet-Normalen ungewollte Kanten im Objekt entstehen, die das Matchingergebnis stören.

Die Beobachtungen zur Beleuchtung haben ergeben, dass durch die Verwendung einer Lichtquelle bei den meisten Detektoren eine kleine Verbesserung zu bemerken ist. Wenn jedoch eine Lichtquelle benutzt wird, ist es wichtig, dass diese relativ realistisch gerendert wird. Ein falsch positioniertes Licht hat in den meisten Fällen eine schlechtere Poseschätzung zur Folge.

Abbildung 4.18: Beispielbilder, in denen das Modell mit der jeweils berechneten Pose über das Videobild gelegt wurde. Die obere Reihe ist mit dem Förstner-Operator (NCC mit RANSAC, mit Textur, Beleuchtung und interpolierten Normalen) und die untere Reihe mit dem SUSAN-Detector (NCC mit RANSAC, ohne Textur, mit Beleuchtung und interpolierten Normalen) berechnet worden. Die jeweils linken Bilder stammen vom Beginn der Sequenz, die mittleren von der Mitte und die rechten vom Ende.

Abbildung 4.18 zeigt exemplarisch einige Bilder, in denen das Modell mit der berechneten Pose gerendert wurde und über den Videoframe gelegt wurde. Die oberen, mit dem Förstner-Operator berechneten Posen gehören zu den besten Resultaten

dieses Trackingsystems. Hier ist gut zu sehen, dass das Modell immer genau dort „einschnappt“, wo die meisten Korrespondenzen gefunden wurden. Dieses Verhalten ist zum größten Teil auf die Qualität des Modells zurückzuführen und hat ein einigermaßen großes Jittern zur Folge. Die unteren Bilder stellen den schlechtesten Fall dar: eine Poseschätzung mit SUSAN auf einem untexturierten Modell.

Kapitel 5

Zusammenfassung

In der vorliegenden Diplomarbeit wurde ein auf „Analyse durch Synthese“ sowie Featuredetektoren basierendes Trackingsystem implementiert, beschrieben und getestet. Das Ziel war die Untersuchung im Hinblick auf den Mehrwert der Computergraphik in einem markerlosen Trackingablauf, indem der Ansatz der „Analyse durch Synthese“ zur Poseschätzung eingesetzt wird.

Dieses System benutzt demnach ein synthetisch erzeugtes Bild der Szene, um die Position und Orientierung der realen Kamera zu schätzen. Dazu wurde ein von der Universität Koblenz-Landau bereitgestelltes 3D-Modell in OpenGL gerendert und mit GLU dargestellt. Anhand von detektierten Merkmalen im Modell- und Kamerabild wurden Punktkorrespondenzen erstellt, die auf den Zusammenhang der Positionen und Orientierungen der in beiden Bildern dargestellten Objekte schließen ließen. Mit Hilfe der 3D-Informationen des Modells, konnten Tiefeninformationen zu den 2D-Merkmalen im Modellbild berechnet werden, die es ermöglichten durch eine wiederholte Berechnung und Minimierung des Rückprojektionsfehlers die Pose der realen Kamera zu schätzen.

Zur Merkmalsdetektion wurden der SIFT-Operator, der Harris Corner Detector, der KLT-Detektor, der Förstner-Operator, SUSAN sowie FAST verwendet. Für das Matching der Merkmale wurde die NCC, die NSSD und RANSAC implementiert. Der Tukey-Schätzer übernahm zusammen mit dem Downhill-Simplex-Verfahren die Minimierung des Rückprojektionsfehlers zur Bestimmung der Pose.

Die durchgeführten Tests haben gezeigt, dass die Verwendung eines texturierten 3D-Modells gute Trackingergebnisse erzielen kann, ein untexturiertes jedoch weniger gute Ergebnisse erzielt. Die Beleuchtung spielt eine eher untergeordnete Rolle, ledig-

lich SIFT und FAST profitieren von einer Lichtquelle. Wenn jedoch eine Beleuchtung benutzt wird, hat sich die korrekte Positionierung als immens wichtig herausgestellt. Alle Detektoren haben bei einer falschen Positionierung der Lichtquelle ein schlechteres Verhalten gezeigt.

Das System arbeitet aufgrund der ausreichend vorliegenden Informationen durch das Modell gänzlich ohne Referenzbilder, was den Vorteil hat, dass die Vorlaufzeit für das Tracking auf die Bestimmung der intrinsischen Kameraparameter minimiert werden kann. Des Weiteren haben Verdeckungen und andere Objekte im Videobild keinen großen Einfluss auf das Trackingergebnis, da die an diesen Stellen gefundene Merkmale nicht mit Merkmalen im Modellbild gematcht werden. Der einzige Nachteil der entsteht ist, dass im Videobild mehr Merkmale detektiert werden müssen, um genügend wichtige Features zu erhalten.

5.1 Ausblick

Dieser Abschnitt befasst sich mit möglichen Verbesserungen und Erweiterungen des implementierten Trackingsystems für nachfolgende Forschungsarbeiten.

Es hat sich gezeigt, dass die korrekte Modellierung des verwendeten 3D-Modells eine wichtige Rolle spielt. Zum einen kann das Matchingergebnis hierdurch noch verbessert werden, da die in beiden Bildern gefundenen Merkmale so auch tatsächlich immer nahe beieinander liegen. Zum anderen stellt sich ein ungenaues Modell während der Poseschätzung als ein Problem dar, was als Resultat ein größeres Jittern zur Folge hat. Je nach Verteilung der gefundenen Merkmale bringt die berechnete Pose das Modell entweder an der Front oder an der Seite zur Deckung. Wie in Abbildungen 4.1 und 4.18 zu sehen ist, sind vor allem die der Fenster nicht korrekt, so dass die Poseschätzung immer fehlerhaft sein wird. Die Liniendetektion ist ein weiterer Punkt, der bei korrektem Modell sicherlich zur Stabiltät der Trackingergebnisse beitragen könnte.

Zu diesem Zeitpunkt ist das Initialisierungsproblem noch nicht gelöst. Die Berechnung durch POSIT ist nicht verlässlich genug für eine gute Ausgangspose, da der Algorithmus sehr stark auf korrekte Punktkorrespondenzen angewiesen ist. Diese können jedoch durch das System aufgrund des fehlerhaften 3D-Modells nicht geliefert werden. Das erste Bild des Videoframes muss also momentan noch möglichst genau mit dem gerenderten Modell übereinstimmen, um direkt mit dem Tukey-Schätzer arbeiten zu können.

Des Weiteren sind die verwendeten Deskriptoren nicht perfekt für modellbasiertes Tracking, denn die Unterschiede zwischen gerendertem Bild und realem Bild sind auch trotz Textur nicht immer gering. Bei einem eher leicht texturierten Modell, wie das in dieser Diplomarbeit verwendete, sind starke Unterschiede zu bemerken, die sich vor allem beim Matching von Linien gezeigt haben. Ein speziell für diese Art von Matching entworfener Deskriptor könnte das Problem lösen.

Außerdem ist die Berechnung der Features in beiden Bildern rechenintensiver, was die Frage aufwirft, ob es möglich ist vorberechnete 3D-Deskriptoren für das Modell zu verwenden. Diese würden dann auf Basis der vorherigen Pose aus einem Datensatz ausgelesen und mit den online erstellten Deskriptoren des Videobildes gematcht werden. Hierbei ist noch zu klären, wie viele solcher Deskriptoren nötig wären, um ausreichend viele Punktkorrespondenzen zu erhalten.

Da eine etwas schlechter berechnete Pose durch die Beschränkung des Suchraums im folgenden Frame zu weniger oder schlechteren Korrespondenzen führen kann, sollte ein Reinitialsierungsalgorithmus implementiert werden, der die Kamerapose wieder in die richtige Position und Orientierung bringt. Dazu ist es wichtig, dass das System automatisch erkennen kann, wann die Berechnung der Pose zu schlecht für weiteres erfolgreiches Tracking ist. Den Fehler der nichtlinearen Optimierung zu benutzen, wäre ein erster Anhaltspunkt. Dies ist aber nur relativ verlässlich, solange die der Poseschätzung zugrunde liegenden Punktkorrespondenzen sehr gut im Bild verteilt sind, denn genau dieser Rückprojektionsfehler wird ja minimiert. Da eine schlechte Verteilung jedoch einen wichtigen Grund für eine ungenaue Schätzung darstellt, kann dieses Fehlermaß nicht alleine verwendet werden. Es muss also eine weitere Referenz gefunden werden, die die globale Position des Objekts überprüft und deren Güte bewertet. Möglich wäre z. B. die Vertrauenswürdigkeit der Korrespondenzen anhand des Korrelationsmaßes mit einfließen zu lassen, oder sogar - wie vor allem beim Förstner-Operator möglich - die Vertrauenswürdigkeit eines Merkmals.

Die Detailliertheit des Modells könnte noch weiter getestet werden. Zu untersuchen hierbei ist, ob es nicht ausreicht, wenn Modelldetails, wie z. B. die planar auf der Hauswand liegenden Fenster, in der Textur integriert werden. Das B-Gebäude müsste nur von allen vier Seiten fotografiert und die so entstandenen Bilder als Textur über das Modell gelegt werden. Dadurch würde das Modell vereinfacht und gleichzeitig die Genauigkeit ohne großen Aufwand erhöht werden.

Dieses System wurde im Hinblick auf Outdoor-Trackinganwendungen entworfen, so dass lediglich ein 3D-Modell der Szene existieren muss, um das Tracking zu starten. Heutzutage sind Projekte im Aufbau, die sich genau mit solchen 3D-Nachbildungen

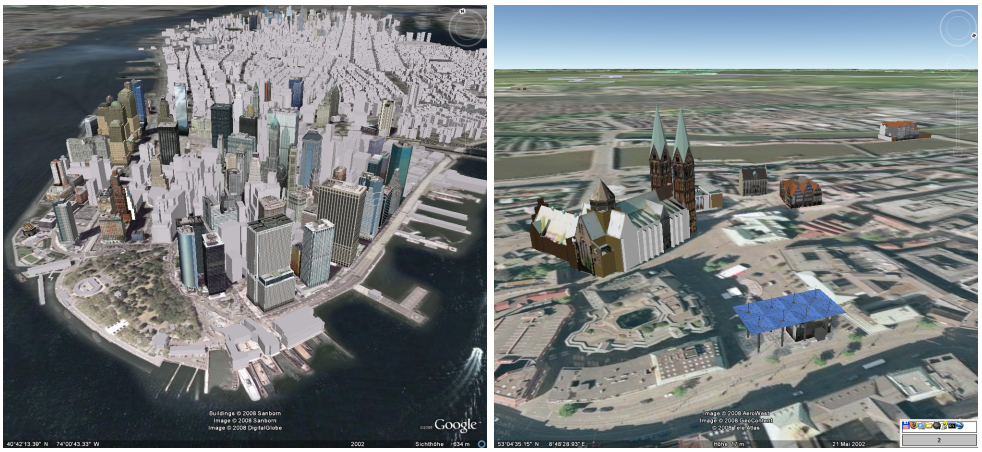


Abbildung 5.1: 3D-Modelle in Google Earth: links Manhattan und rechts Bremen

der realen Welt befassen. Google z. B. hat begonnen in ihrem Google Earth Projekt ganze Straßenzüge dreidimensional darzustellen (siehe Abbildung 5.1). Es ist anzunehmen, dass in den nächsten Jahren mehr und mehr Modelle auch aus anderen Projekten zur Verfügung stehen werden, um ein auf „Analyse durch Synthese“ basierendes Trackingsystem für weit reichende Anwendungsbereiche einzusetzen.

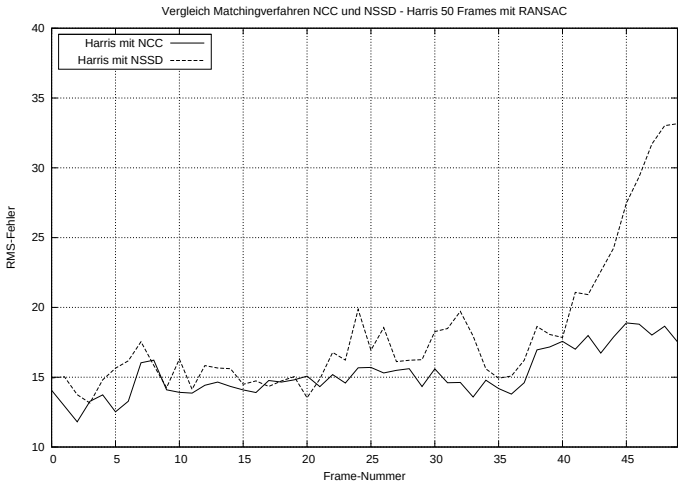
Anhang A

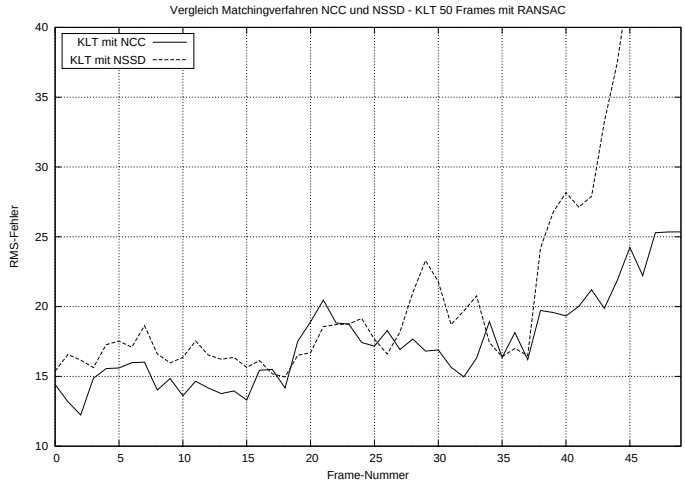
Detaillierte Ergebnisse

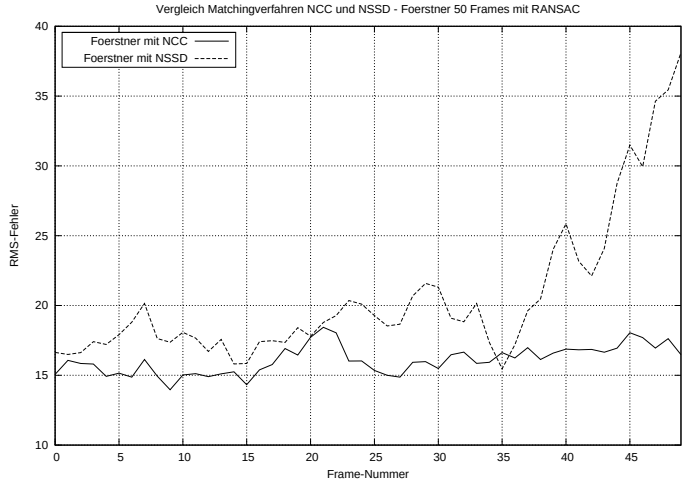
In diesem Anhang werden einige Abbildungen von detaillierten Ergebnissen der Tests dargestellt.

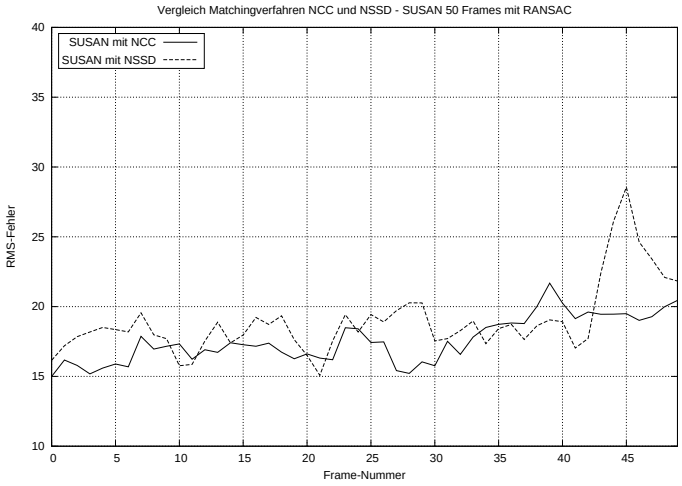
A.1 Vergleiche Matchingverfahren

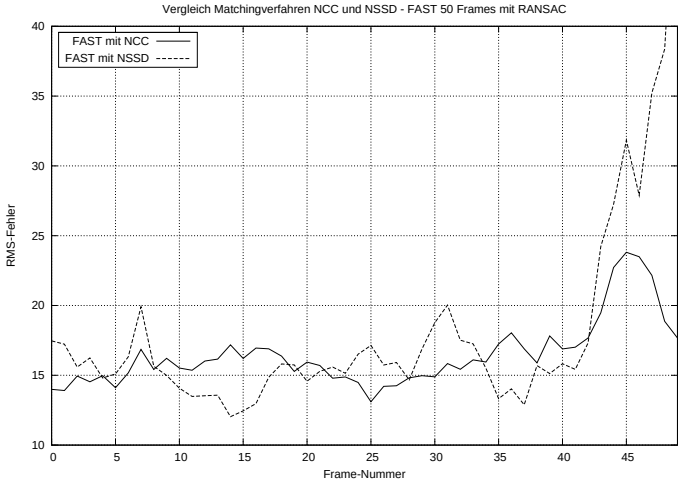
Mit RANSAC



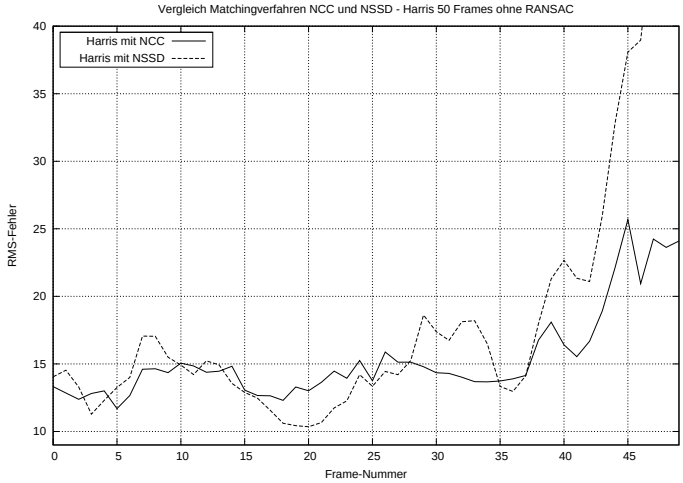

Vergleich Matchingverfahren NCC und NSSD - KLT 50 Frames mit RANSAC
KLT mit NCC
KLT mit NSSD
RMS-Fehler
Frame-Nummer
10
15
20
25
30
35
40
0
5
10
15
20
25
30
35
40
45

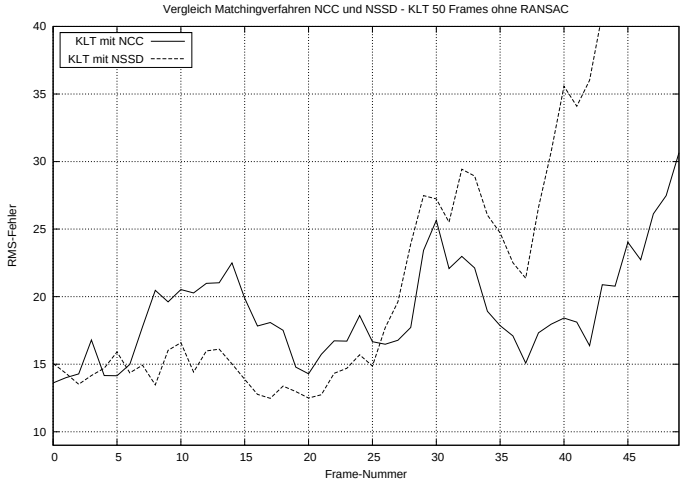

Vergleich Matchingverfahren NCC und NSSD - Foerstner 50 Frames mit RANSAC
Foerstner mit NCC
Foerstner mit NSSD
RMS-Fehler
Frame-Nummer
10
15
20
25
30
35
40
0
5
10
15
20
25
30
35
40
45

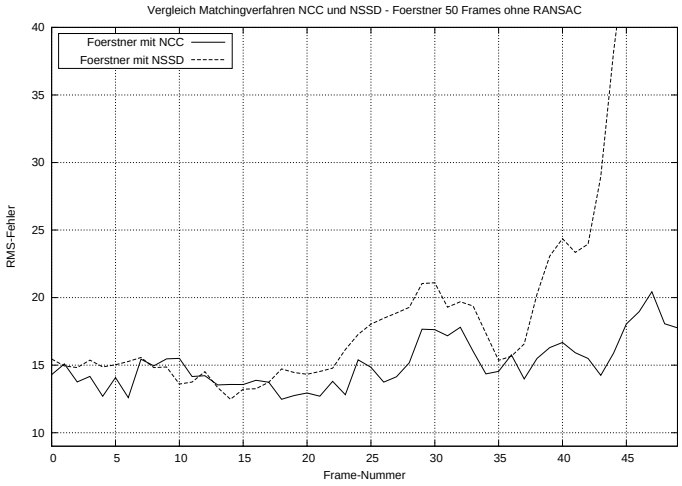

Vergleich Matchingverfahren NCC und NSSD - SUSAN 50 Frames mit RANSAC
SUSAN mit NCC
SUSAN mit NSSD
RMS-Fehler
Frame-Nummer
40
35
30
25
20
15
10
0
5
10
15
20
25
30
35
40
45

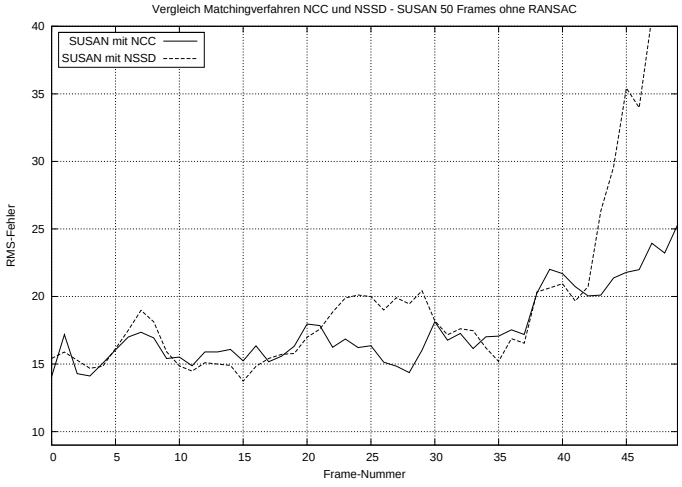

Vergleich Matchingverfahren NCC und NSSD - FAST 50 Frames mit RANSAC
FAST mit NCC
FAST mit NSSD
RMS-Fehler
Frame-Nummer
40
35
30
25
20
15
10
0
5
10
15
20
25
30
35
40
45

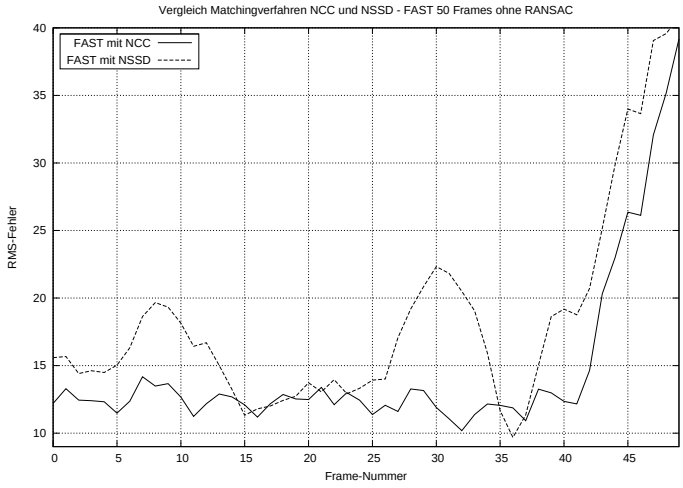
Ohne RANSAC


Vergleich Matchingverfahren NCC und NSSD - Harris 50 Frames ohne RANSAC
Harris mit NCC
Harris mit NSSD
RMS-Fehler
Frame-Nummer
40
35
30
25
20
15
10
0
5
10
15
20
25
30
35
40
45


Vergleich Matchingverfahren NCC und NSSD - KLT 50 Frames ohne RANSAC
KLT mit NCC
KLT mit NSSD
RMS-Fehler
Frame-Nummer
40
35
30
25
20
15
10
0
5
10
15
20
25
30
35
40
45


Vergleich Matchingverfahren NCC und NSSD - Foerstner 50 Frames ohne RANSAC
Foerstner mit NCC
Foerstner mit NSSD
RMS-Fehler
Frame-Nummer
40
35
30
25
20
15
10
0
5
10
15
20
25
30
35
40
45


Vergleich Matchingverfahren NCC und NSSD - SUSAN 50 Frames ohne RANSAC
SUSAN mit NCC
SUSAN mit NSSD
RMS-Fehler
Frame-Nummer
40
35
30
25
20
15
10
0
5
10
15
20
25
30
35
40
45


Vergleich Matchingverfahren NCC und NSSD - FAST 50 Frames ohne RANSAC
FAST mit NCC
FAST mit NSSD
RMS-Fehler
Frame-Nummer
40
35
30
25
20
15
10
0
5
10
15
20
25
30
35
40
45

Literaturverzeichnis

[Alt02] W. ALT. *Nichtlineare Optimierung* (Vieweg Verlag, 2002)

[ART] ARTOOLKIT. *http://www.hitl.washington.edu/artoolkit*

[BFH+61] C. G. BELL, H. FUJISAKI, J. M. HEINZ, K. N. STEVENS & A. S. HOUSE. *Reduction of Speech Spectra by Analysis-by-Synthesis Techniques*. In: *Journal of the Acoustical Society of America*, *Bd. 33* (12): (1961) S. 1725–1736

[Ble04] G. BLESER. *Entwicklung eines 3D-markerlosen Kamera-Trackingverfahrens zur Echtzeit-Augmented-Reality-Bildsynthese*. Diplomarbeit, Universität Koblenz-Landau, 2004

[BS79] N. I. BADLER & S. W. SMOLIAR. *Digital Representations of Human Movement*. In: *ACM Comput. Surv.*, *Bd. 11* (1): (1979) S. 19–38

[BTvG06] H. BAY, T. TUYTELAARS & L. VAN GOOL. *Surf: Speeded Up Robust Features*. In: *9th European Conference On Computer Vision* (Graz Austria, 2006)

[DD92] D. DEMENTHON & L. S. DAVIS. *Model-Based Object Pose in 25 Lines of Code*. In: *ECCV '92: Proceedings of the Second European Conference on Computer Vision* (Springer-Verlag, London, UK, 1992), S. 335–343

[DH72] R. O. DUDA & P. E. HART. *Use of the Hough Transformation to Detect Lines and Curves in Pictures*. In: *Commun. ACM*, *Bd. 15* (1): (1972) S. 11–15

[Dic05] T. DICKSCHEID. *Automatische Referenzpunktverfeinerung in Panoramabildern mittels SIFT-Operator*. Studenarbeit, Universität Koblenz-Landau, 2005

[Ewe06] D. EWERING. *Modellbasiertes Tracking mittels Linien- und Punktkorrelationen*. Diplomarbeit, Universität Koblenz-Landau, 2006

[FA91] W. T. FREEMAN & E. H. ADELSON. *The Design and Use of Steerable Filters*. In: *IEEE Trans. Pattern Analysis and Machine Intelligence*, *Bd. 13* (9): (1991) S. 891–906

[Fau93] O. FAUGERAS. *Three-dimensional computer vision: a geometric viewpoint* (MIT Press, Cambridge, MA, USA, 1993)

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

[FG87] W. FÖRSTNER & E. GÜLCH. *A Fast Operator for Detection and Precise Location of Distinct Points, Corners and Centers of Circular Features*. In: *Proceedings of ISPRS Intercommission Conference on Fast Processing of Photogrammetric Data* (Interlaken, Schweiz, 1987), S. 281–305

[Gru41] J. A. GRUNERT. *Das photenotische Problem in erweiterter Gestalt nebst Bilder über seine Anwendung in der Geodäsie*. In: *Grunerts Archiv für Mathematik und Physik*, *Bd. 1*: (1841) S. 238–248

[Hes] R. HESS. *http://web.engr.oregonstate.edu/ hess/index.html*

[HLON94] R. M. HARALICK, C.-N. LEE, K. OTTENBERG & M. NÖLLE. *Review and analysis of solutions of the three point perspective pose estimation problem*. In: *Int. J. Comput. Vision*, *Bd. 13* (3): (1994) S. 331–356

[HMT83] D. C. HOAGLIN, F. MOSTELLER & J. W. TUKEY. *Understanding Robust and Exploratory Data Analysis* (John Wiley and Sons Ltd, 1983)

[Hou62] P. HOUGH. *Methods and Means for Recognizing Complex Patterns*. U.S. Patent 3 069 654, 1962

[HRRA86] F. R. HAMPEL, E. M. RONCHETTI, P. J. ROUSSEEUW & S. W. A. *Robust Statistics: The Approach Based on Influence Functions* (John Wiley and Sons Ltd, 1986)

[HS88] C. HARRIS & M. STEPHENS. *A Combined Corner and Edge Detection*. In: *Proceedings of the Fourth Alvey Vision Conference* (1988), S. 147–151

[Hub81] P. HUBER. *Robust Statistics* (Wiley, New York, 1981)

[HZ04] R. I. HARTLEY & A. ZISSERMAN. *Multiple View Geometry in Computer Vision* (Cambridge University Press, ISBN: 0521540518, 2004), second Aufl.

[Int] INTEL. *http://www.intel.com/technology/computing/opencv/index.htm*

[Lin93] T. LINDEBERG. *Scale-Space Theory in Computer Vision* (Kluwer Academic Publishers, Norwell, MA, USA, 1993)

[LLF05] V. LEPETIT, P. LAGGER & P. FUA. *Randomized Trees for Real-Time Keypoint Recognition*. In: *Cvpr '05: Proceedings of The 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2* (IEEE Computer Society, Washington, DC, USA, 2005), S. 775–781

[Low99] D. G. LOWE. *Object Recognition from Local Scale-Invariant Features*. In: *ICCV '99: Proceedings of the International Conference on Computer Vision-Volume 2* (IEEE Computer Society, Washington, DC, USA, 1999), S. 1150

[Low04] D. G. LOWE. *Distinctive Image Features from Scale-Invariant Keypoints*. In: *Int. J. Comput. Vision*, *Bd. 60* (2): (2004) S. 91–110

[LVTF03] V. LEPETIT, L. VACCHETTI, D. THALMANN & P. FUA. *Fully automated and stable registration for augmented reality applications*. In: *Second IEEE and ACM International Symposium on Mixed and Augmented Reality* (IEEE Comput. Soc. Press, 2003). CVlab, Swiss Fed. Inst. of Technol., Lausanne, Switzerland

[MC97] B. MARCEL & M. CATTOEN. *Edge and Line Detection in Low Level Analysis*. In: *3rd Workshop on Electronic Control and Measuring System* (1997), S. 89–97

[Mit97] T. M. MITCHELL. *Machine Learning* (McGraw-Hill Higher Education, 1997)

[MN78] D. MARR & H. K. NISHIHARA. *Representation and recognition of the spatial organization of three dimensional shapes*. In: *Proceedings of the Royal Society of London B*, *Bd. 200*: (1978) S. 269–294

[Moe99] T. B. MOESLUND. *The Analysis-by-Synthesis Approach in Human Motion Capture: A Review*. In: *The 8th Danish conference on pattern recognition and image analysis* (1999)

[Mor77] H. MORAVEC. *Towards Automatic Visual Obstacle Avoidance*. In: *Proceedings of the 5th International Joint Conference on Artificial Intelligence* (1977), S. 584

[MS05] K. MIKOLAJCZYK & C. SCHMID. *A performance evaluation of local descriptors*. In: *IEEE Transactions on Pattern Analysis & Machine Intelligence*, *Bd. 27* (10): (2005) S. 1615–1630
URL `http://lear.inrialpes.fr/pubs/2005/MS05`

[NA02] M. S. NIXON & A. S. AGUADO. *Feature Extraction & Image Processing* (Butterworth Heinmann/ Newnes, 2002)
URL `http://eprints.ecs.soton.ac.uk/6806/`

[ODD96] D. OBERKAMPF, D. F. DEMENTHON & L. S. DAVIS. *Iterative pose estimation using coplanar feature points*. In: *Comput. Vis. Image Underst.*, *Bd. 63* (3): (1996) S. 495–511

[RD05] E. ROSTEN & T. DRUMMOND. *Fusing points and lines for high performance tracking.* In: *IEEE International Conference on Computer Vision*, Bd. 2 (2005), S. 1508–1511
URL `http://mi.eng.cam.ac.uk/~er258/work/rosten_2005_tracking.pdf`

[RD06a] G. REITMAYR & T. DRUMMOND. *Going Out: Robust Model-Based Tracking for Outdoor Augmented Reality*. In: *ISMAR '06: Proceedings of the 5th IEEE and ACM International Symposium on Mixed and Augmented Reality* (IEEE Computer Society, Los Alamitos, CA, USA, 2006), S. 109–118

[RD06b] E. ROSTEN & T. DRUMMOND. *Machine learning for high-speed corner detection*. In: *European Conference on Computer Vision (to appear)* (2006)

URL `http://mi.eng.cam.ac.uk/~er258/work/rosten_2006_machine.pdf`

[Ros] E. ROSTEN. *http://svr-www.eng.cam.ac.uk/ er258/work/fast.html*

[SB97] S. M. SMITH & J. M. BRADY. *SUSAN - A New Approach to Low Level Image Processing*. In: *Int. J. Comput. Vision*, *Bd. 23* (1): (1997) S. 45–78

[Smi] S. M. SMITH. *http://users.fmrib.ox.ac.uk/ steve/susan/*

[ST94] J. SHI & C. TOMASI. *Good Features to Track*. In: *1994 IEEE Conference on Computer Vision and Pattern Recognition (cvpr'94)* (1994), S. 593 – 600

[Sut64] I. E. SUTHERLAND. *Sketchpad: a Man-Machine Graphical Communication System*. In: *DAC '64: Proceedings of the SHARE design automation workshop* (ACM, New York, NY, USA, 1964), S. 6.329–6.346

[TM08] T. TUYTELAARS & K. MIKOLAJCZYK. *A survey on local invariant features*. In: *Foundations and Trends in Computer Graphics and Vision* URL `http://homes.esat.kuleuven.be/~tuytelaa/FT_survey_inv_features_submitted.pdf`

[TV98] E. TRUCCO & A. VERRI. *Introductory Techniques for 3-D Computer Vision* (Prentice Hall PTR, Upper Saddle River, NJ, USA, 1998)

[VLF04] L. VACCHETTI, V. LEPETIT & P. FUA. *Combining Edge and Texture Information for Real-Time Accurate 3D Camera Tracking*. In: *ISMAR '04: Proceedings of the 3rd IEEE and ACM International Symposium on Mixed and Augmented Reality* (IEEE Computer Society, Los Alamitos, CA, USA, 2004), S. 48–57

[WB95a] H. WANG & M. BRADY. *Real-time corner detection algorithm for motion estimation*. In: *Image and Vision Computing*, *Bd. 13* (9): (1995) S. 695–703

[WB95b] G. WELCH & G. BISHOP. *An Introduction to the Kalman Filter. Techn. Ber.*, Chapel Hill, NC, USA, 1995

[WS06] H. WUEST & D. STRICKER. *Tracking of industrial objects by using CAD models*. In: *3rd Workshop Virtuelle und Erweiterte Realität der GI-Fachgruppe VR/AR* (Koblenz, 2006)

[ZW94] R. ZABIH & J. WOODFILL. *Non-Parametric Local Transforms for Computing Visual Correspondence*. In: *Eccv '94: Proceedings of the Third European Conference - Volume ii on Computer Vision* (Springer-Verlag, London, UK, 1994), S. 151–158