Untersuchung des Einflusses von Pixelunsicherheiten auf Bundle-Feature-Parametrisierungen in der visuellen simultanen Lokalisierung und Kartenerstellung

Christoph Spallek

Mai 2009

Inhaltsverzeichnis

Kapitel 1

Einleitung und Motivation

Der Bereich der visuellen kamerabasierten Lokalisierung und Kartenerstellung (VSLAM) benötigt selbst auf aktuellen Desktop-Rechnern hohe Rechenleistungen, damit die entsprechenden Programme in Echtzeit ablaufen können. Ein Problem ist dabei der Wunsch eine möglichst große Karte der Welt zu erstellen. Jede Kartenerweiterung führt aber zu quadratisch steigenden Kosten bei der Berechnung. Es wird daher nach Modellen geforscht, welche die Informationen der Karte komprimieren können. Um damit letztendlich die Laufzeit der Algorithmen zu senken. In der Arbeit von Pietzsch [Pie08] wurde ein solches Modell vorgestellt, welches die Größe des Zustandsraumes reduzieren kann und damit zu Geschwindigkeitsverbesserungen führte. Es lehnt sich an die Arbeit von Azarbayejani und Pentland [AP94] an. Allerdings werden in diesem Modell Vereinfachungen vorgenommen. In dem Paper von Chiuso et al. [CFM+02] werden diese Vereinfachungen aber kritisiert und als zu einschränkend beschrieben. Chiuso et al. haben dies mit einer theoretischen Ausarbeitung und einfachen Experimenten belegt.

Diese Arbeit wird den kritisierten Sachverhalt untersuchen. Dazu werden die Beschränkungen aus dem Modell von Pietzsch beseitigt. Und dann die beiden Modelle auf Basis ihrer Schätzgenauigkeit verglichen. Hierfür wird das Test-Framework aus Pietzsch und Funke [PF09] genutzt. Es ermöglicht die Untersuchung anhand praxisnaher kontrollierbarer Experimente. Denn die Testumgebung ist eine synthetisch generierte realitätsnahe Simulation der Welt. Damit können die praktisch relevanten Einschränkungen genauer untersucht werden.

Wir werden am Anfang eine Einführung in die Grundlagen des VSLAM geben und dabei auf die Besonderheiten des von uns verwendeten Systemes eingehen. Danach stellen wir das Modell von Pietzsch vor. Und darauf aufbauend die Veränderungen, die wir vorgenommen haben, um dessen Einschränkungen zu beseitigen. Nach dem theoretischen Teil der Arbeit gehen wir dann zum experimentellen Teil über. In diesem Kapitel stellen wir die Testumgebung vor und präsentieren die Testresultate der zwei unterschiedlichen Modelle. Abschließend erfolgt die Interpretation und Auswertung der gewonnenen Ergebnisse.

Kapitel 2

Grundlagen

2.1 SLAM

SLAM bezeichnet die simultane Lokalisierung und Kartenerstellung (Simultaneous Localization And Mapping). Es ist allgemein das Problem, von einer unbekannten Welt eine Karte aufzubauen und gleichzeitig von einem Vehikel in dieser Welt die Position zu verfolgen. Der Begriff SLAM stellt dabei einen Oberbegriff dar, da es eine Vielzahl von Möglichkeiten gibt an dieses Problem heranzugehen.

So kann die Karte auf verschiedene Arten repräsentiert werden. In einigen Systemen ist die Karte dabei als Umriss der realen Welt dargestellt, in anderen wiederum besteht die Karte aus kleinen Bildern, die von einer Kamera geschossen wurden. Das Vehikel ist ein Objekt, welches sich in der Welt bewegen kann. Dabei kann es sich um einen radgetriebenen Roboter handeln, der sich in einem Höhlensystem bewegt oder einen Tauchroboter in einer Unterwasserwelt. Aber auch um eine Kamera die am Kopf eines Menschen befestigt ist. Je nachdem um was für ein Vehikel es sich handelt besitzt es unterschiedlichste Sensorik. Mit dieser Sensorik kann es seine Umgebung erkunden und die eigene Fortbewegung messen (Odometriesensoren). Typische Sensoren zur Umgebungserkundung sind unter anderem Laser-, Radar-, Sonar- und Bildkamerasysteme. Jeder dieser Sensoren bietet bestimmte Vor- und Nachteile. Ein Laserscanner hat beispielsweise eine höhere Genauigkeit als eine Radargerät, ist aber anfälliger für Reflektionen und ist abhängig von guten Sichtbedingungen. Auch weitere Faktoren wie Kosten und Gewicht spielen ein Rolle bei der Bewertung. Weiterhin kann das Vehikel Odometriesensoren besitzen. Diese Messen die Fortbewegung des Vehikels. Ein Beispiel ist ein Umdrehungszahlmesser an einem Roboterrad. Kombiniert mit der Kenntnis über den Umfang des Rades liefert dieser die zurückgelegte Entfernung des Roboters. Am Anfang befindet sich das Vehikel an einer unbekannten Position und besitzt unvollständige Informationen über die Welt. Durch die sensorische Erkundung baut es Schritt für Schritt die Karte auf. Je länger sich das Vehikel in der Welt bewegt umso mehr Informationen sammelt es über die Welt. Gleichzeitig versucht es seine Position in der Welt, also auf der Karte, zu bestimmen. In Abbildung 2.1 ist dieses Vorgehen illustriert.

Für das SLAM-Problem gibt es verschiedene Techniken. Das Grundprinzip ist aber bei allen ähnlich. Das Vehikel besitzt zu jedem Zeitpunkt eine Schätzung

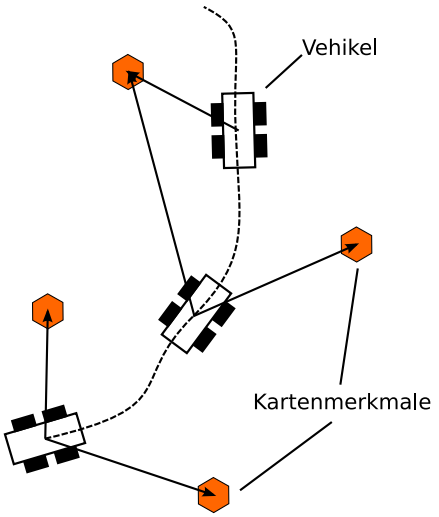


Abbildung 2.1: Das Vehikel bewegt sich auf der gestrichelten Linie. Die Kartenmerkmale sind charakteristische Punkte in der Welt, die durch die Sensoren erkannt werden (dargestellt durch die Pfeile). Bei fortschreitender Bewegung werden weitere Kartenmerkmale beobachtet und alte wiedererkannt.

seiner aktuellen Position. Dazu können auch Geschwindigkeit und Beschleunigung gehören. Zu Beginn werden diese Werte vorgegeben. Die Position wird im allgemeinen auf die Ursprungskoordinaten der Weltkarte initialisiert. Das Vehikel schätzt mithilfe seiner eventuell vorhandenen Odometriesensoren und der Kenntnis über seine aktuelle Geschwindigkeit und Beschleunigung eine neue Position ausgehend von der vorherigen geschätzten Position. Mit den Umgebungssensoren wird die Welt vermessen und mittels dieser Messungen eine Karte aufgebaut.

Alle Informationen, also die Positionsschätzungen und die Sensormessungen sind allerdings mit einer Unsicherheit behaftet. Ein Umdrehungszahlmesser an einem Roboterrad misst zuviele Umdrehungen, wenn das Rad Schlupf hat. Demzufolge ist der ausgegebene Wert für die zurückgelegte Entfernung zu groß. Ein Laserscanner hat nur eine eingeschränkte Genauigkeit, die Entfernungsmessung erfolgt beispielsweise nur auf 1 cm genau. Zusätzlich kann der Strahl reflektiert werden und damit zu große Entfernungen anzeigen oder ein bewegliches Objekt gerät in den Strahl und verfälscht so die Messung. Um diese Ungenauigkeiten zu korrigieren betrachtet man Messungen nur als Schätzung und belegt sie mit einer Unsicherheit. Man hat gewisse Modellannahmen zu treffen, wie diese Unsicherheiten aussehen. Ein Möglichkeit ist die Annahme, dass die exakte Messung durch ein normalverteiltes Rauschen überlagert wird. Diese Annahme ermöglicht die Angabe eines Konfidenzintervalles um die Messung herum. Für den Roboter mit Radschlupf, der sich in einer 2D-Welt bewegt, ist dann zum Beispiel folgende Aussage möglich. Mit einer Wahrscheinlichkeit von 95% befindet sich der Roboter in einem geschätztem ellipsenförmigen Gebiet um die ebenfalls zu schätzende Position herum. Oder anders formuliert, in 95% der Messungen befindet sich der Roboter in dem geschätzten Gebiet. Zu Beachten ist dabei, dass wenn die Modellannahmen nicht zutreffen die daraus abgeleiteten Aussagen keine Gültigkeit

besitzen müssen.

Werden fortwährend mit Unsicherheit behaftete Messungen durchgeführt, so wird man sich über die geschätzte Vehikelposition und damit auch die Karte immer unsicherer. Um die Unsicherheit zu reduzieren werden die bisher ermittelten Karteninformationen mit einbezogen. Aus der bisher gelernten Karte kann man Erwartungen extrahieren, wo sich bestimmte Kartenelemente befinden. Hat man eine Wand in der Karte erkannt und schätzt die Roboterposition 2 m vor dieser Wand, dann erwartet man, dass die Umgebungssensorik diese Wand in 2 Meter Entfernung misst. Misst man die Wand aber nun in 3 Meter in Entfernung, dann kann man die Position des Roboter korrigieren und die Karte anpassen. Dabei kombiniert man die Positionschätzung mit der Sensormessung. Dabei wird die Unsicherheit der Informationen mit einbezogen. Sichere Informationen werden dabei mit einer größeren Gewichtung versehen. So könnte man, bei größerem Vertrauen in die Umgebungssensoren als in die Odometriesensoren, die Wand 20 cm weiter weg vom Roboter auf der Karte platzieren und die Positionsschätzung des Roboter 50 cm auf der Karte zurückschieben. Die Unsicherheiten werden dabei auch angepasst. Bei großen Abweichungen von Schätzung und Messung, vergrößert sich die Unsicherheit, bei kleinen Abweichungen verringert sie sich. Die genaue Beurteilung was eine große beziehungsweise kleine Abweichung ist, hängt von der eingesetzten Sensorik, den jeweils angenommenen Schätz- und Messmodellen sowie der gewünschten Genauigkeit ab.

Ein SLAM-Algorithmus muss also folgende Prozeduren durchführen können:

- Prädiktion der Vehikeldaten (Position, Geschwindigkeit, ...)
- Kartenerstellung
- Verknüpfung der erstellten Karte mit den Messungen
- Korrektur von Vehikelposition und der Karte

Dabei bilden die Prädiktion und die Korrektur eine Einheit. Sie werden nacheinander durch ein Verfahren abgearbeitet. Es gibt wiederum verschiedene Verfahren, die die Prädiktion und Korrektur ermöglichen. Verbreitete Verfahren nutzen das Erweiterte Kalman-Filter (EKF). Dies ist ein rekursives Filter, das 1960 von Rudolf Kálmán entwicklet wurde. Das EKF nutzt dazu die Schätzung und die Unsicherheit. Dabei geht das Filter von einem normalverteilten Rauschen aus. Weitere Verfahren sind Partikel-Filter, diese stellen die Wahrscheinlichkeitsverteilung direkt als Wolke von Partikeln dar und ermöglicht so die Handhabung von nicht normalverteilen Rauschen.

2.2 VSLAM

Im Folgendem wird das konkrete System und SLAM-Verfahren beschrieben, dass wir verwendet haben.

Eine Instanz des SLAM-Problems ist die visuelle simultane Lokaliserung und Kartenerstellung (VSLAM). Dabei wird ein Kamerasystem als Umgebungssensor genutzt. Es stellt bei uns auch gleichzeitig das Vehikel dar. Kamerabasierte Sensoren bieten mehrere Vorteile. Sie sind im Allgemeinen einfach und billig zu beschaffen und haben ein leichtes Gewicht. Die erhaltenen Bilder sind für den Menschen auch gut intuitiv erfassbar. Nachteilig ist, dass auf den Bildern teils

komplexe und rechenintensive bildverarbeitende Verfahren auszuführen sind, um Karteninformationen zu extrahieren. Die Karte wird repräsentiert durch kleine Bilder (Patches), die mit einer 3D-Koordinate und einer Ausrichtung versehen sind. Dies ist im Abschnitt 2.2.1 näher erläutert.

Wir haben als Kamerasystem eine Stereokamera verwendet. Da das System nur aus der Stereokamera besteht, ist es nicht an weitere Geräte gebunden, also frei im Raum bewegbar. Aber dadurch sind auch keine Odometrieinformationen verfügbar. Man kann die Vorhersage der Kameraposition also nur aufgrund der aktuellen Schätzung über die Position und Geschwindigkeit durchführen. Der Vorteil eines Stereokamerasystems, also zwei gekoppelten Kameras, ist die einfache Extraktion der Tiefe aus dem Bild.

Als Prädiktions- und Korrekturverfahren kam das Erweiterte Kalman-Filter (EKF) zum Einsatz.

2.2.1 Feature

Die Karte besteht in unserem Programm aus mehreren Features. Ein Feature setzt sich aus folgenden Komponenten zusammen:

- Patch
- 3D-Koordinaten
- Ausrichtung

Patch Zur Erstellung der Karte wird das Bild auf lokale Merkmale untersucht. Dazu wird der Eckendetektor FAST (Features from Accelerated Segment Test) genutzt. Dieser extrahiert markante Pixel mit ihrer zugehörigen Umgebung aus einem Bild. Das sind die Patches, also zweidimensionale Texturen, die für die Kartenerstellung verwendet werden. Sie sind von entscheidender Bedeutung. Denn es wird versucht die Patches in weiteren Durchläufen erneut zu detektieren. Dies ist die Verknüpfung der erstellten karte mit den Messungen. Weitere Informationen zu dem verwendeten Eckendetektor FAST finden sich unter [RD05] und [RD06]. Alternativ können auch andere Eckendetektionsmechanismen zum Einsatz kommen, wie zum Beispiel SIFT (Scale-Invariant Feature Transform - Skaleninvariante Merkmalstransformation [Low99]). Dabei ist aber die Rechenkomplexität zu berücksichtigen. FAST ist ein sehr schneller Algorithmus und kann daher sehr gut in einer Echtzeitumgebung eingesetzt werden. Auch wenn andere Methoden eine bessere Qualität in Hinsicht auf die detektierten Ecken aufweisen, so ist doch immer auf die zeitliche Dauer der Detektion zu achten.

Für die Extraktion der Features sind drei Eigenschaften wichtig:

- Einzigartigkeit der Feature-Patches, so dass sie gut wiedererkannt werden können
- Mindestanzahl von Features in einem Bild
- Die Features sind möglichst gleichmäßig über das ganze Bild verteilt

Bei einer Wiedererkenung der Patches wird das Bild nach einem Auschnitt mit minimaler Abweichung vom Originalpatch durchsucht. Die Abweichung für einen Bildausschnitt berechnet sich über alle einzelne Abweichungen der Pixel

vom Originalpatch. Darüber wird dann die Quadratsumme $\sum_{i,j=1}^{k}(g_{i,j} - h_{i,j})$ berechnet, wobei g für das Orinalpatch steht und h für einen Bildauschnitt. Es wird das Patch mit der kleinsten Quadratsumme ausgewählt und damit ist dessen Mittelpunkt die beobachtete Feature-Position bzw. 3D-Koordinate. Der Eckendetektor ist dafür verantwortlich, dass nur Patches mit einer hohen Wiedererkennrate ausgewählt werden.

Die Mindestanzahl der Features in einem Bild und ihre Gleichverteilung darin sind Heuristiken die sich bewährt haben. Die Idee dahinter ist möglichst viele verschiedene Weltinformationen zu nutzen, damit die Schätzungen genauer werden.

3D-Koordinaten Die 3D-Koordinaten an der sich das Patch befindet können aus den Kameradaten gewonnen werden. Dazu wird das Lochkameramodell genutzt um die Pixel u, v auf der Bildebene mit den weltkoordinaten x, y, z zu verknüpfen. Zusätzlich ist die Kenntnis der Disparität d erforderlich. Diese gibt den Abstand an, den ein und derselbe Patch in den beiden Bilder der Stereokamera voneinander hat. Dazu ist nachdem ein Patch detektiert wurde im anderen Bild auf einer horizontalen Linie das Patch zu suchen, welches dem detektierten Patch am ehesten entspricht. Also wie oben beschrieben die geringste Abweichung der Quadratsumme aufweist. Der gemessene Abstand in Pixeln gibt die Disparität an.

Die Transformation zwischen den Weltkoordinaten x, y, z der Features und ihren gemessenen Bildkoordinaten u, v, d im Kamerakoordinatensystem kann man durch die folgenden Projektionen berechnen.

Dabei werden folgende Notationen verwendet:

- x, y, z - Die 3D-Koordinaten des Patches gemessen in m
- u, v - Die 2D-Koordinaten auf der Bildebene, gemessen in Pixeln
- d - Die Disparität, der Abstand von einem Patch zu seinem korrespondierenden Patch im linken Bild, gemessen in Pixeln
- b - Die BasisLinie, der Abstand zwischen den beiden Kameras, gemessen von Mittelpunkt zu Mittelpunkt in m
- f_x, f_y - Die Brennweite in x- bzw. y-Richtung, gemessen in Pixeln
- x_0, y_0 - Die Koordinaten des Bildmittelpunktes gemessen in Pixeln

Die Projektion von den 3D-Koordinaten in die Bildebene erfolgt mit:

$$\begin{pmatrix} u \\ v \\ d \end{pmatrix} = \begin{pmatrix} f_x \frac{x}{z} + x_0 \\ f_y \frac{y}{z} + y_0 \\ f_x \frac{b}{z} \end{pmatrix} \tag{2.1}$$

Analog ergeben sich aus den Bildkoordinaten die 3D-Koordinaten durch die inverse Projektion:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} (u - x_0)\frac{b}{d} \\ (v - y_0)\frac{f_x}{f_y}\frac{b}{d} \\ f_x \frac{b}{d} \end{pmatrix} \tag{2.2}$$

Die inverse Projektion wendet man bei der Initialisierung der Features an. Man hat die hat die Werte u, v, d und benötigt die 3D-Koordinaten x, y, z. Bei

der erneuten Beobachtung von Features benutzt man die Projektion. Man kennt die geschätzte 3D-Position x, y, z, aber braucht die Bildkoordinaten u, v, d an der man das Feature im aktuellen Kamerabild suchen kann.

Ausrichtung Es wird angenommen, dass im Moment der Bildaufnahme die Patches dieselbe Ausrichtung wie die Kamera besitzen. Ihre Ausrichtung, repräsentiert durch eine Normale, wird dementsprechend mit einer Senkrechten zur Bildebene initialisiert. Dies stellt eine Vereinfachung des Sachverhaltes dar, denn die Patches könne sich auch auf schrägen und gekrümmten Flächen befinden. Die Ausrichtung wird bei der Wiedererkenung genutzt. Bei einem erneuten Beobachtungsversuch der Features sind die Patches durch einen Standortwechsel des Vehikels gegebenenfalls verzerrt. Um sie trotzdem wiederzuerkennen, wird versucht die Verzerrung vorherzuberechnen. Dazu benötigt man die Ausrichtung der Patches und den aktuellen Blickwinkel auf das Patch. Durch eine kollinearen Abbildung wird das Patch in das zu erwartende Aussehen transformiert.

Mit den 3D-Koordinaten, dem Patch und seiner Ausrichtung ist das Feature vollständig beschrieben. Im folgenden verwenden wir dir Bezeichnung Feature-Vektor synonym für die 3D-Koordinaten.

2.2.2 Erweiterter Kalman-Filter

Die Aufgabe des VSLAM-Systems ist neben dem Aufbau der Karte die Verfolgung des Vehikels, also der Kamera. Wie oben beschrieben gehört dazu die Prädiktion und Korrektur der Vehikel- und Feature-Daten. Dazu arbeitet das dafür eingesetzte Erweiterte Kalman-Filter (EKF) mit einem Zustandsraum, der im folgendem beschrieben wird.

Zustandsraum

Der Zustand $\mathbf{x}_k$ des EKF zum Zeitpunkt k besteht aus den Vehikeldaten und den Feature-Vektoren. Weiterhin gehören die geschätzten Unsicherheiten in Form von Kovarianzmatrizen ebenfalls zum Zustand.

Die Vehikeldaten setzen sich zusammen aus der Postion $\mathbf{r}$ und Rotation $\mathbf{q}$ der Kamera, sowie der Translations- $\mathbf{v}$ und Rotationsgeschwindigkeit $\boldsymbol{\omega}$.

$$\mathbf{x}_c = \begin{pmatrix} \mathbf{r} \\ \mathbf{q} \\ \mathbf{v} \\ \boldsymbol{\omega} \end{pmatrix} \tag{2.3}$$

Für die Beschreibung der Features reichen die Feature-Vektoren $\mathbf{y}_i$, da der Patch und die Ausrichtung nur einmal initialisiert werden und danach unveränderlich sind. Im einfachen Fall der Kartesischen Parametrisierung (Abschnitt 3.1) besteht der Feature-Vektor daher nur aus den 3D-Koordinaten des Features. Patch und Ausrichtung sind nicht Bestandteil des Zustandes des EKF.

$$\mathbf{y}_i = \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} \tag{2.4}$$

Im Kapitel 3 werden weitere Möglichkeiten der Feature-Parametrisierung beschrieben.

Zu jedem Zustand $\mathbf{x}_k$, bestehend aus Kameradaten $\mathbf{x}_c$ und den Features $\mathbf{y}_i$, gibt es eine Kovarianz P_k. Sie beschreibt den statistischen Zusammenhang der Merkmale des Zustandes untereinander.

Modellannahmen

Zusätzlich wird die Beschleunigung der Kamera $\mathbf{w}$ erfasst. Dabei ist $\mathbf{a}$ die Translationsbeschleunigung und $\boldsymbol{\alpha}$ die Winkelbeschleunigung.

$$\mathbf{w}_k = \begin{pmatrix} \mathbf{a} \\ \boldsymbol{\alpha} \end{pmatrix} \tag{2.5}$$

Die Beschleunigung ist eine zufällige Störung, die von außen auf die Kamera einwirkt. Man nimmt an, dass die Störung $\mathbf{w}_k$ einem normalverteiltem Rauschen mit dem Erwartungswert 0 und der Kovarianz Q_k folgt.

$$\mathbf{w}_k \sim N(0, Q_k) \tag{2.6}$$

Bei der Beobachtung tritt ebenfalls eine Rauschen $\mathbf{v}_k$ auf. Auch hier nimmt man an, dass es normalverteilt mit dem Erwartungswert 0 und der Kovarianz R_k ist.

$$\mathbf{v}_k \sim N(0, R_k) \tag{2.7}$$

Für das EF-Filter nimmt man an, dass folgende Modelle für den Zustandsübergang (Kamerabewegung) beziehungsweise die Beobachtung gelten.

$$\mathbf{x}_k = f(\mathbf{x}_{k-1}, \mathbf{u}_k) + \mathbf{w}_k \tag{2.8}$$

$$\mathbf{z}_k = g(\mathbf{x}_k) + \mathbf{v}_k \tag{2.9}$$

$$\tag{2.10}$$

Gleichung 2.8 beschreibt die angenommene Bewegung der Kamera. Der neue Zustand des Systems $\mathbf{x}_k$ wird geschätzt, indem auf den alten Zustand $\mathbf{x}_{k-1}$ die deterministische Störung $\mathbf{u}_k$ (hier geschätzte Kamerabewegung) einwirkt. Die genaue Form wird durch die Funktion f beschrieben. Die Funktion kann nichtlinear sein, muss aber differenzierbar sein. Darauf wird ein zufälliges Rauschen addiert, die unbekannte Beschleunigung $\mathbf{w}_k$. Die Beobachtung $\mathbf{z}_k$ in Gleichung 2.9 hängt nichtlinear von dem neuen berechneten Zustand $\mathbf{x}_k$ und dem Beobachtungsrauschen $\mathbf{v}_k$ ab. Der Zusammenhang wird durch die Funktion g modelliert. Diese wird im Kapitel 3 näher erklärt.

Funktionsweise

Das EKF versucht den Zustand des Systems möglichst genau zu schätzen und geht dazu in zwei Schritten vor. Der erste Schritt ist die Prädiktion der Werte, beschrieben durch folgende Gleichungen.

$$\hat{\mathbf{x}}_{k|k-1} = f(\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{u}_k) \tag{2.11}$$

$$P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q_k \tag{2.12}$$

Der neue Zustand $\hat{\mathbf{x}}_{k|k-1}$ wird ähnlich zur Modellannahme 2.8 geschätzt. Die zugehörige Kovarianz $P_{k-1|k-1}$ wird dementsprechend angepasst. Nur das Rauschen kann nicht mit einbezogen werden, da es unbekannt ist. Dafür wird die Unsicherheit erhöht, indem der Kovarianzmatrix P die Rauschmatrix Q_k dazu addiert wird. Die Matrix F_k beschreibt dabei den linearisierten Übergang zwischen den Zuständen. Dazu ist die Ableitung von der Funktion f zu bilden, die Jacobi-Matrix $\frac{\partial f}{\partial \mathbf{x}}$.

$$\mathbf{F}_k = \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{u}_k} \tag{2.13}$$

Im darauffolgendem Korrekturschritt werden die Messungen der Kamera mit einbezogen.

$$\tilde{\mathbf{y}}_k = \mathbf{z}_k - g(\hat{\mathbf{x}}_{k|k-1}) \tag{2.14}$$
$$S_k = G_k P_{k|k-1} G_k^T + R_k \tag{2.15}$$
$$K_k = P_{k|k-1} G_k^T S_k^{-1} \tag{2.16}$$
$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + K_k \tilde{\mathbf{y}}_k \tag{2.17}$$
$$P_{k|k} = (I - K_k G_k) P_{k|k-1} \tag{2.18}$$

$$\mathbf{G}_k = \left.\frac{\partial g}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_{k|k-1}} \tag{2.19}$$

Dabei ist $\tilde{\mathbf{y}}_k$ der Unterschied zwischen der realen Messung $\mathbf{z}_k$ und der erwarteten Messung, auch Residualkovarianz genannt. Sie ist eine Maß für die fehlerhafte Schätzung im Prädiktionsschritt. Die Innovation S_k gibt an, wie groß die Unsicherheit für einen Wert des Zustandes ist. Eine hohe Unsicherheit schlägt sich in hohen Werten in der Innovationsmatrix nieder. Eine Berechnung der Innovation S_k beinhaltet die Anpassung der Kovarianz mittels des linearisierten Beobachtungsmodells H_k (Gleichung 2.19) und der Addition der Kovarianz des Beobachtungsrauschens R_k. Der Term K_k beschreibt die Kalman-Matrix, diese bezieht zusätzlich noch die Güte der Umgebungssensoren mit ein. Wenn man diesen mehr vertraut, dann ist der entsprechende Wert in der Kalman-Matrix auch größer. Aus der Kalman-Matrix K_k und der Residualkovarianz $\tilde{\mathbf{y}}_k$ kann nun der neue korrigierte Zustand berechnet werden. Eine Korrektur ist umso größer, je größer die Residualkovarianz ist und der entsprechende Eintrag durch die Kalman-Matrix stärker gewichtet wird. Die Kovarianz wird ebenfalls in Abhängigkeit von der Kalman-Matrix und dem Beobachtungsmodell aktualisiert.

Das EKF hat eine quadratische Komplexität sowohl für die Zeit als auch den Speicher. Dies ist eine Folge des Aufbaus der Kovarianzmatrizen. Denn in ihnen werden alle möglichen Zustandspaare korreliert. Das heißt eine Verdoppelung der Features führt fast zu einer Vervierfachung des Speicherplatzbedarfes (Die Vehikeldaten bleiben dabei konstant). Da die Kovarianzmatrizen auch im Prädiktions- und Korrekturschritt angewendet werden, erhöht sich auch der Rechenaufwand quadratisch. Da die Features die Karte bilden, ist ihre Anzahl ausschlaggebend um sich in komplizierteren Welten zurechtzufinden. Eine Reduktion der Parameter pro Feature ist daher erstrebenswert, denn dadurch ist eine höhere Feature-Anzahl möglich ohne mehr Speicher- und Rechenkapazität bereitzustellen. Demzufolge wurden verschiedene Modelle zur Parametrisierung der Features entwickelt, z.B. in Azarbayejani und Pentland [AP94] und in Montiel

et al. [MCD06]. Im nächsten Kapitel wird dabei besonders auf die Inverse-Depth-Feature-Parametrisierung und ihre Erweiterungen eingegangen.

Kapitel 3

Feature-Parametrisierungen

Für die Darstellung in der Karte muss die Position eines Features immer als 3D-Koordinate vorliegen. Die interne Rechnung im EKF kann allerdings mit einer anderen Parametrisierung erfolgen. Es ist dann jeweils eine Umrechnung erforderlich, um die Features auf der Karte darzustellen.

3.1 Kartesische Parametrisierung

Die kartesische Parametrisierung ist eine einfache Parametrisierung. Die Position des Features wird hierbei direkt durch seine Raumkoordinaten angegeben.

$$\mathbf{y}_i = \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} \tag{3.1}$$

Es ist somit keine Umrechnung zwischen den 3D-Koordianten und der Parameterdarstellung notwendig, da diese identisch sind.

3.2 Inverse-Depth-Feature-Bundle-Modell

Das Inverse-Depth-Feature-Bundle-Modell (kurz IDFB-Modell) baut auf der Inverse-Depth-Feature-Parametrisierung auf. Wir geben deswegen zuerst eine Einführung in die Inverse-Depth-Feature-Parametrisierung.

3.2.1 Inverse-Depth-Feature-Parametrisierung

Bei dieser Parametrisierung werden die Feature-Positionen in der Welt nicht direkt angegeben sondern indirekt beschrieben. Zum Zeitpunkt der ersten Beobachtung wird die aktuelle Kameraposition $\mathbf{c}$ und -rotation $\mathbf{R}$ gespeichert sowie die Entfernung der Kamera zum Feature. Aus technischen Gründen wird dabei die inkrementelle Rotation und die inverse Tiefe verwendet, dies wird in den folgenden Abschnitten erläutert. Mittels dieser Werte können dann die Raumkoordinaten berechnet werden. Abbildung 3.1 illustriert das Prinzip allgemein. Ein Feature $\mathbf{y}$ wird also durch 7 Parameter beschrieben. Diese sind die Kameraposition $\mathbf{c}$, die

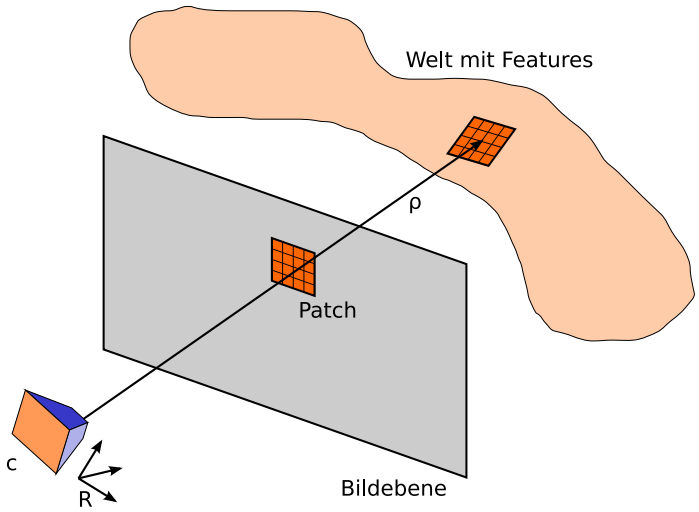


Abbildung 3.1: Wenn ein Feature beobachtet wird, dann kann die Position im Raum eindeutig durch die Kameraposition **c** und -rotation **R**, sowie der inversen Tiefe ρ bestimmt werden.

inkrementelle Kamerarotation $\boldsymbol{\phi}$ und die inverse Tiefe ρ.

$$\mathbf{y}_i = \begin{pmatrix} \mathbf{c}_i \\ \boldsymbol{\phi}_i \\ \rho_i \end{pmatrix} \tag{3.2}$$

Initialisierung

In der Initialierung werden die Werte des Feature-Vektors aus Gleichung 3.2 festgelegt, sowie weitere unveränderliche Hilfsgrößen. Die Initialsierung findet zum Zeitpunkt der ersten Beobachtung $\mathbf{h}_{new} = (u_i, v_i, d_i)^T$ des Features statt.

Die Hilfsgrößen sind fest und demzufolge nicht im Feature-Vektor enthalten. Dies ist einmal die Kamerarotation $\mathbf{q}_i^0$ angegeben als Quaternion.

$$\mathbf{q}_i^0 = \mathbf{q} \tag{3.3}$$

Weiterhin wird noch ein Einheitsvektor $\mathbf{m}_i$ berechnet, dieser beschreibt einen normierten Einheitsstrahl von der Kamera zu dem Feature. Dabei werden ebenfalls die Werte der ersten Beobachtung genutzt.

$$\mathbf{m}_i = \frac{1}{\sqrt{1 + \left(\frac{u-x_0}{f_x}\right)^2 + \left(\frac{v-y_0}{f_y}\right)^2}} \begin{pmatrix} \frac{u-x_0}{f_x} \\ \frac{v-y_0}{f_y} \\ 1 \end{pmatrix} \tag{3.4}$$

Der veränderliche Feature-Vektor kann nun wie folgt bestimmt werden. Es wird die Kameraposition r des aktuellen Zustandes im Feature-Vektor gespeichert. Die Kameraposition ist dabei das optische Zentrum der Kamera.

$$\mathbf{c}_i = \mathbf{r} \tag{3.5}$$

Die inkrementelle Kamerarotation $\boldsymbol{\phi}_i$ gibt die Differenz zur initialen Kamerarotation q_i^0 an. Sie ist am Anfang 0 und kann durch Schätzungen in folgenden Iterationen des EK-Filters verändert werden. Wir geben die Kamerarotation nicht direkt an, da die Ableitung der Einheitsrotation $\mathbf{q} = (1, 0, 0, 0)^T$ singulär ist. Durch die Aufteilung in einen festen Teil $\mathbf{q}_i^0$ und einen veränderlichen Anteil $\boldsymbol{\phi}$ und der Anwendung der Umkehrregel kann man dieses Problem umgehen.

$$\boldsymbol{\phi}_i = \log(\mathbf{q} \circ \mathbf{q}_i^{0^{-1}}) \tag{3.6}$$

Über die Parallaxe (2.2.1) wird der Abstand des Features zur Kamera bestimmt, dies ist die Tiefe. Im Feature-Vektor wird aber die inverse Tiefe $\rho_i = 1/d$ abgespeichert. Dies erleichtert die Behandlung von Features, da laut Montiel et al. [MCD06] dadurch deren Unsicherheit besser durch eine Normalverteilung, die vom EKF angenommen wird, approximiert werden kann. Nach Gleichung 2.1 ergibt sich

$$\rho_i = d\frac{\mathbf{m}_i^z}{f_x b}. \tag{3.7}$$

$\mathbf{m}_i^z$ gibt dabei die z-Komponente von $\mathbf{m}_i$ an.

Zusammengefasst ist die Initialisierung eines Features $\mathbf{y}_i$

$$\mathbf{y}_i = g(\mathbf{x}_c, \mathbf{h}_{new}) = \begin{pmatrix} \mathbf{c}_i \\ \boldsymbol{\phi}_i \\ \rho_i \end{pmatrix} = \begin{pmatrix} \mathbf{r} \\ \log(\mathbf{q} \circ \mathbf{q}_i^{0^{-1}}) \\ d\frac{\mathbf{m}_i^z}{f_x b} \end{pmatrix} \tag{3.8}$$

Berechnung der Messposition

Für die weiteren Iterationen des EFK sind aus den Features $\mathbf{y}_i$ die zu erwartenden Messpositionen $\mathbf{h}_i = (u_i, v_i, d_i)^T$ zu bestimmen.

Die 3D-Koordinaten des Features können wie folgt berechnet werden. Dabei gibt $R(\cdot)$ die Umwandlung eines Quaternions zu einer Rotationsmatrix an und $Q(\cdot)$ die Umwandlung von einem Eulerschen Winkeln in ein Quaternion.

$$\mathbf{y}_i = \mathbf{c}_i + \frac{1}{\rho_i} R(Q(\boldsymbol{\phi}_i))R(\mathbf{q}_i^0)m \tag{3.9}$$

Die zu messende Bildpostion $h_i = (u_i, v_i, d_i)^T$ ergibt sich dann mittels der Projektionsgleichung 2.1 zu

$$\mathbf{h}_i = \begin{pmatrix} u_i \\ v_i \\ d_i \end{pmatrix} = proj(p^{c,\omega}(\mathbf{y}_i^\omega)). \tag{3.10}$$

Dabei gibt $p^{c,\omega}$ die Transformation der Koordinaten vom Weltkoordinatensystem (ω) in das Koordinatensystem der Kamera (c) an. Dies ist erforderlich, weil das EKF mit den Weltkoordinaten arbeitet, während hingegen die Messung im Koordinatensystem der Kamera erfolgt.

Jacobi-Matrizen

Für das EKF sind die linearisierten Ableitungen $F_k = \frac{\partial f}{\partial x}$ und $G_k = \frac{\partial g}{\partial x}$ zu bestimmen (siehe Abschnitt 2.2.2). Dabei beschreibt x den ganzen Zustandsraum, also den Kamerazustand $\mathbf{x}_c$ und die Features y_i.

Die Bestimmung von F_k ist in der Arbeit von Pietzsch [Pie09] nachzulesen.

Für die Jacobi-Matrix G_k muss die Funktion $g(\mathbf{x}_c, \mathbf{h}_{new})$ nach dem Kamerazustand $\mathbf{x}_c$ und der neuen Messung $\mathbf{h}_{new}$ abgeleitet werden. Alle anderen Ableitungen nach den Features y_i sind 0, da sie nicht in der Funktion $g(\mathbf{x}_c, \mathbf{h}_{new})$ auftreten.

Die Ableitung $\frac{\partial g}{\partial \mathbf{x}_c}$ ist

$$\frac{\partial g}{\partial \mathbf{x}_c} = \begin{bmatrix} \frac{\partial c_i}{\partial \mathbf{r}} & 0 & 0 & 0 \\ 0 & \frac{\partial \boldsymbol{\phi}_i}{\partial \mathbf{q}} & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.11}$$

mit den partiellen Ableitungen

$$\frac{\partial c_i}{\partial \mathbf{r}} = \mathrm{I}_{3\mathrm{x}3} \tag{3.12}$$

und

$$\frac{\partial \boldsymbol{\phi}_i}{\partial \mathbf{q}} = \frac{\partial \log(\mathbf{q} \circ \mathbf{q}_i^{0^{-1}})}{\partial \mathbf{q} \circ \mathbf{q}_i^{0^{-1}}} \frac{\partial \mathbf{q} \circ \mathbf{q}_i^{0^{-1}}}{\partial \mathbf{q}} \tag{3.13}$$

Dabei ist mit $\mathbf{q} \circ \mathbf{q}_i^{0^{-1}} = (1, 0, 0, 0)^T$

$$\frac{\partial \log(\mathbf{q} \circ \mathbf{q}_i^{0^{-1}})}{\partial \mathbf{q} \circ \mathbf{q}_i^{0^{-1}}} = \begin{pmatrix} 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{pmatrix} \tag{3.14}$$

Der Term

$$\frac{\partial \mathbf{q} \circ \mathbf{q}_i^{0^{-1}}}{\partial \mathbf{q}} \tag{3.15}$$

bestimmt sich nach A.3.

Die Funktion g nach $\mathbf{h}_{new}$ abgeleitet ergibt

$$\frac{\partial g}{\partial \mathbf{h}_{new}} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \frac{\partial \rho}{\partial d} \end{bmatrix} \tag{3.16}$$

mit

$$\frac{\partial \rho}{\partial d} = \frac{\mathbf{m}_i^z}{f_x b}. \tag{3.17}$$

Damit sind alle nötigen Ableitungen für die Jacobi-Matrizen der IDF-Parametrisierung beschrieben.

3.2.2 Inverse-Depth-Feature-Bundle-Parametrisierung

Die Inverse-Depth-Feature-Parametrisierung kodiert ein Feature durch 7 Parameter, die kartesische Parametrisierung durch 3 Parameter. Man hat mit der IDF-Parametrisierung also sogar den Aufwand erhöht. Aber diese Parametrisierung erlaubt eine signifikante Einsparung, wenn die Features in Bundles (Bündeln) initialsiert werden. Das heißt, zu dem Zeitpunkt der ersten Beobachtung eines Features wird nicht nur ein Feature initialsiert, sondern ein Bündel von mehreren Features. Alle diese Features teilen sich die Kameraposition und -rotation von der ersten Beobachtung gemeinsam. Dieser gemeinsame Anteil,

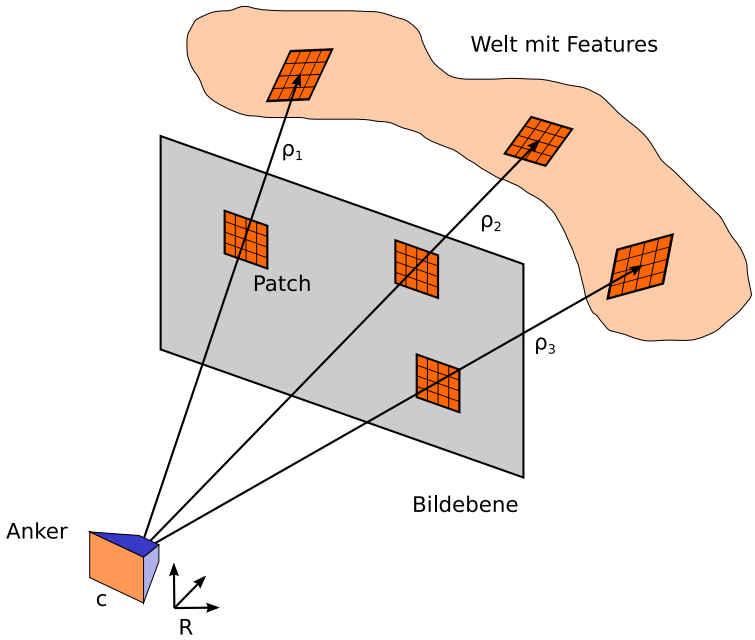


Abbildung 3.2: Ein Anker speichert Kameratranslation **c** und -rotation **R** für mehrere Features. Die Features haben nur einen individuellen Parameter, die inverse Tiefe ρ

also Kameraposition **c** und Kamerarotation **R**, repäsentiert durch $\boldsymbol{\phi}$ und $\mathbf{q}^0$, ist der Anker. Der Anker existiert nur einmal für ein Bündel von Features und eine Anpassung des Ankers durch das EKF beeinflusst alle Features in diesem Bündel. Dadurch, dass ein Anker von mehreren Features genutzt wird, gibt es aber auch eine Einschränkung. Diese Technik ist nur unter der Voraussetzung korrekt, dass keine Unsicherheit in den initialen Pixelpositionen besteht. Werden also von exakt derselben Szene mehrere Bilder geschossen, dann müssen in jedem Bild die Features an genau den gleichen Pixelpositionen beobachtet werden. Wenn dies nicht der Fall ist, dann ist die Parametrisierung nur eine Approximation der Realität und es könnte dadurch zu Fehlschätzungen bei der Kameraverfolgung und Kartenerstellung kommen. Modelle dieser Art wurden von Chiuso et. al [CFM+02] als subminimal bezeichnet.

Ein Bündel von n Features kann durch dies Inverse-Depth-Bundle-Feature-Parametrisierung durch $6 + n$ Parameter beschrieben werden. Für jedes der n Features den Parameter inverse Tiefe ρ plus den gemeinsamen Anker, der 6 Parameter (Kameraposition und -rotation) einnimmt. Die Umrechnungen in die 3D-Koordinaten ist analog zu der normalen Inverse-Depth-Feature-Parametrisierung.

Im Vergleich zur Kartesischen Parametrisierung, welche $3n$ Parameter benötigt, bedeutet dies im Idealfall fast eine Einsparung von $2n$ Parametern. Die Parameterreduktion ist aber nur erfolgreich, wenn das Verhältnis Anker zu Features möglichst klein ist. Das wird durch folgende Punkte erreicht:

a) neue Anker selten initialisieren

b) mehrere Features pro Anker initialisieren

Der erste Punkt bedeutet, dass neue Features und damit der zugehörige Anker nur initialsiert werden sollten, wenn die bereits vorhandenen Features nicht mehr ausreichen um ein Verfolgen der Kamera zu gewährleisten. Wenn die Lageveränderung der Kamera doch eine erneute Initialsierung notwendig macht, dann sollten möglichst mehrere Features initialisert werden. Die optimale Anzahl hängt dabei vom jeweiligen Anwendungsfall und den vorhandenen Rechenkapazitäten ab, aktuell werden pro Anker maximal 40 Features verteilt über das Bild initialisiert.

3.3 Entkoppeltes Inverse-Depth-Feature-Bundle-Modell

Im vorherigen Abschnitt wurde die Inverse-Depth-Bundle-Feature-Parametrisierung beschrieben. Durch die Reduktion des Feature-Vektors ermöglicht das IDFB-Modell eine sehr kompakte Angabe eines Features. Allerdings ist wie bereits erwähnt dieses Modelle unter Umständen nur subminimal. Da es nur angewendet werden kann unter der Annahme der Sicherheit der initialen Pixelpositionen. Durch Faktoren wie Kamerarauschen oder Änderung der Linsengeometrie werden die Bilder, die mit einer realen Kamera aufgenommen aber verändert. Sind diese Änderungen zu stark, kann es passieren, dass die initialen Pixelpositionen nicht eindeutig sind. Es ist also eine erhöhte Unsicherheit vorhanden.

Um zu kontrollieren ob die Inverse-Depth-Feature-Parametrisierung in diesem Fall noch auf das SLAM-Problem anwendbar ist, haben wir eine Vergleichsmodell entwickelt, das entkoppelte Inverse-Depth-Feature-Bundle-Modell. Wie am Namen erkennbar sind hierbei die Features voneinander entkoppelt. Dazu haben wir die Einschränkung, dass alle Features eines Bündels dieselbe Rotation aufweisen müssen, aufgehoben. Hierfür haben wir die Rotation aus dem Anker in die Features verlagert. Dies ermöglicht den Umgang mit der erhöhten Unsicherheit, da das EKF mehr Möglichkeiten hat die Feature-Vektoren anzupassen. Denn durch die individuelle Rotation können die oben genannten individuellen Störungen ausgeglichen werden. Der Anker besteht jetzt nur noch aus 3 Parametern nämlich der Kameraposition $\mathbf{c}$. Der individuelle Anteil dagegen ist auf 4 Parameter angewachsen, der Kamerarotation $\boldsymbol{\phi}$und der inversen Tiefe ρ.

Alternativ hätte die kartesische Parametrisierung auch eine freiere Rotation erlaubt. Der Vergleich fand aber nicht mit der kartesischen Parametrisierung statt, da sich die Herangehensweisen dabei gravierend unterschieden hätten. So nutzt die kartesische Parametrisierung beispielsweise keine inverse Tiefe. Veränderte Ergebnisse wäre damit nicht eindeutig auf die Entkopplung zurückzuführen.

Initialisierung

Dadurch, dass die Rotation nicht mehr unveränderlich und von u und v abhängig ist, verändert sich die Berechnung der Initialiserung.

Um die individuelle Rotation für jedes Feature zuzulassen haben wir den zusätzlichen Term $\mathbf{q_s}$ in die Rotation $\mathbf{q}$ eingeführt. Die Rotation $\boldsymbol{\phi}$ ergibt sich damit zu:

$$\boldsymbol{\phi}_i = \log(\mathbf{q}) = \log(\mathbf{q_s} \circ \mathbf{q} \circ \mathbf{q}_i^{0^{-1}}) \tag{3.18}$$

Da sich im Wesentlichen die Features nur auf der Kameraebene verschieben, kann man die Rotation $\mathbf{q_s}$ durch zwei einzelne Drehungen approximieren. Einmal die

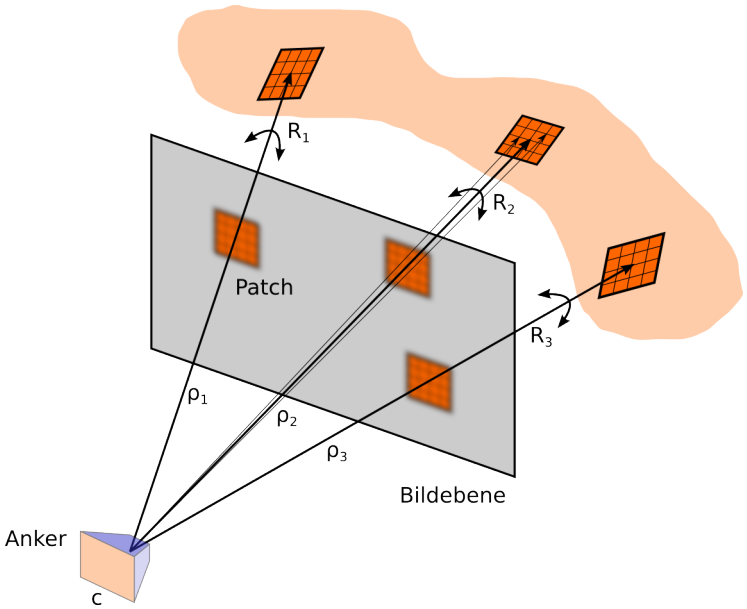


Abbildung 3.3: Durch Störungen könnten die Patches nicht mehr die korrekten Positionen widerspiegeln. Dies kann durch eine individuelle Rotation ausgeglichen werden. Der Anker speichert demzufolge nur noch die Kameratranslation $\mathbf{c}$ für mehrere Features. Die Features haben nun die individuellen Parameter Kamerarotation $\mathbf{R}$ und inverse Tiefe ρ.

Rotation $\mathbf{q_{s_x}}$ um die x-Achse und einmal um die y-Achse $\mathbf{q_{s_y}}$. Beide Drehungen werden hintereinander ausgeführt.

$$\mathbf{q_s} = \mathbf{q_{s_x}} \circ \mathbf{q_{s_y}} \tag{3.19}$$

Die Drehung $\mathbf{q_{s_x}}$ um die x-Achse kann beschrieben werden durch

$$\mathbf{q_{s_x}} = \begin{pmatrix} w_x \\ x_x \\ y_x \\ z_x \end{pmatrix} = \begin{pmatrix} \cos(\frac{\theta_v}{2}) \\ \sin(\frac{\theta_v}{2}) \\ 0 \\ 0 \end{pmatrix}. \tag{3.20}$$

Der Winkel θ_v berechnet sich dabei aus der initialen vertikalen Pixelkoordinate v_0 und der angenommen verschobenen vertikalen Pixelkoordinate v. Die Werte y_0 und f_y sind in Abschnitt 2.2.1 erklärt.

$$\theta_v = \arctan(\frac{v - y_0}{f_y}) - \arctan(\frac{v_0 - y_0}{f_y}) \tag{3.21}$$

Die Drehung q_{s_y} wird analog zu q_{s_x} beschrieben. Veranschaulicht ist diese in Abbildung 3.4.

$$\mathbf{q_{s_y}} = \begin{pmatrix} w_y \\ x_y \\ y_y \\ z_y \end{pmatrix} = \begin{pmatrix} \cos(\frac{\theta_u}{2}) \\ 0 \\ \sin(\frac{\theta_u}{2}) \\ 0 \end{pmatrix} \tag{3.22}$$

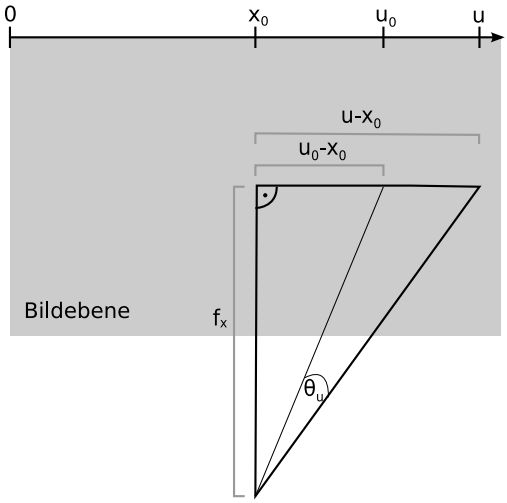


Abbildung 3.4: Im Bild ist die Drehung q_{s_y} dargestellt. Die Verschiebung der erkannten Feature-Position von u_0 auf u wird durch die zusätzliche Drehung um den Winkel θ_u ermöglicht. Die Berechnung des Winkels erfolgt durch die Anwendung der trigonometrischen Funktionen.

$$\theta_u = \arctan(\frac{u - x_0}{f_x}) - \arctan(\frac{u_0 - x_0}{f_x}) \tag{3.23}$$

Die Rotation $\mathbf{q_s}$ kann dann mit 3.20 und 3.22 nach A.1 wie folgt berechnet werden:

$$\mathbf{q_s} = \begin{pmatrix} w_x w_y \\ w_y x_x \\ w_x y_y \\ x_x y_y \end{pmatrix} \tag{3.24}$$

Berechnung der Messposition

Die Berechnung der zu erwartenden Messpositionen $\mathbf{h}_i = (u_i, v_i, d_i)^T$ verändert sich nicht und ist wie in Abschnitt 3.2.1 ist

$$\mathbf{y}_i = \mathbf{c}_i + \frac{1}{\rho_i} R(Q(\boldsymbol{\phi}_i)) R(\mathbf{q}_i^0) m \tag{3.25}$$

$$\mathbf{h}_i = \begin{pmatrix} u_i \\ v_i \\ d_i \end{pmatrix} = proj(p^{c,\omega}(\mathbf{y}_i^\omega)). \tag{3.26}$$

Jacobi-Matrizen

Da sich die Zusammensetzung der Kamerarotation $\boldsymbol{\phi}$ verändert hat und die Kamerarotation sowie die inversen Tiefe nun von veränderlichen Pixelkoordinaten u und v abhängen, ist auch die linearisierten Ableitung G_k von g neu zu bestimmen. Die Jacobi-Matrix F_k bleibt gleich, weil sich nur die Initialiserung geändert hat.

Die Ableitung $\frac{\partial g}{\partial \mathbf{x}_c}$ ist

$$\frac{\partial g}{\partial \mathbf{x}_c} = \begin{bmatrix} \frac{\partial \mathbf{c}_i}{\partial \mathbf{r}} & 0 & 0 & 0 \\ 0 & \frac{\partial \boldsymbol{\phi}_i}{\partial \mathbf{q}} & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.27}$$

Für die Berechnung und Ableitungen der Kameraposition ist keine Modellanpassung notwendig, da dieser Teil identisch zur IDFB-Parametrisierung ist. Somit gilt wieder

$$\frac{\partial \mathbf{c}_i}{\partial \mathbf{r}} = \mathrm{I}_{3\mathrm{x}3}. \tag{3.28}$$

Die Ableitung von $\boldsymbol{\phi}$ nach $\mathbf{q}$ muss dagegen angepasst werden.

$$\frac{\partial \boldsymbol{\phi}_i}{\partial \mathbf{q}} = \frac{\partial \log(\mathbf{q_s} \circ \mathbf{q} \circ \mathbf{q}_i^{0^{-1}})}{\partial \mathbf{q_s} \circ \mathbf{q} \circ \mathbf{q}_i^{0^{-1}}} \frac{\partial \mathbf{q_s} \circ \mathbf{q} \circ \mathbf{q}_i^{0^{-1}}}{\partial \mathbf{q} \circ \mathbf{q}_i^{0^{-1}}} \frac{\partial \mathbf{q} \circ \mathbf{q}_i^{0^{-1}}}{\partial \mathbf{q}} \tag{3.29}$$

Analog zu A.11 und mit $\mathbf{q_s} \circ \mathbf{q} \circ \mathbf{q}_i^{0^{-1}} = (1,0,0,0)^T$ist:

$$\frac{\partial \log(\mathbf{q_s} \circ \mathbf{q} \circ \mathbf{q}_i^{0^{-1}})}{\partial \mathbf{q_s} \circ \mathbf{q} \circ \mathbf{q}_i^{0^{-1}}} = \begin{pmatrix} 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{pmatrix} \tag{3.30}$$

Da bei der Initialsierung keine weiteren Informationen verfügbar sind, nimmt man wieder an, dass alle Features dieselbe Rotation aufweisen. Damit ist jedes $\mathbf{q_s}$ anfangs eine identische Abbildung. $\mathbf{q_s}$ beschreibt also eine Drehung um 0°, dargestellt durch das Einheitsquaternion $(1,0,0,0)^T$. Mit $\mathbf{q_s} = (1,0,0,0)^T$ und A.2 folgt also:

$$\frac{\partial \mathbf{q_s} \circ \mathbf{q} \circ \mathbf{q}_i^{0^{-1}}}{\partial \mathbf{q} \circ \mathbf{q}_i^{0^{-1}}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.31}$$

Der Term

$$\frac{\partial \mathbf{q} \circ \mathbf{q}_i^{0^{-1}}}{\partial \mathbf{q}} \tag{3.32}$$

bestimmt sich erneut nach A.3.

Durch die veränderlichen Pixelkoordinaten u und v im entkoppleten IDFB-Modell sind bei der Ableitung von g nach $\mathbf{h}_{new}$ die Veränderungen umfangreicher.

$$\frac{\partial g}{\partial \mathbf{h}_{new}} = \begin{bmatrix} 0 & 0 & 0 \\ \frac{\partial \boldsymbol{\phi}}{\partial u} & \frac{\partial \boldsymbol{\phi}}{\partial v} & 0 \\ \frac{\partial \rho}{\partial u} & \frac{\partial \rho}{\partial v} & \frac{\partial \rho}{\partial d} \end{bmatrix} \tag{3.33}$$

Für $\frac{\partial \boldsymbol{\phi}_i}{\partial u}$ ergibt sich

$$\frac{\partial \boldsymbol{\phi}_i}{\partial u} = \frac{\partial \log(\mathbf{q_s} \circ \mathbf{q} \circ \mathbf{q}_i^{0^{-1}})}{\partial \mathbf{q_s} \circ \mathbf{q} \circ \mathbf{q_i^{0^{-1}}}} \frac{\partial \mathbf{q_s} \circ \mathbf{q} \circ \mathbf{q}_i^{0^{-1}}}{\partial \mathbf{q_s}} \frac{\partial \mathbf{q_s}}{\partial u} \tag{3.34}$$

Erneut ist nach A.11:

$$\frac{\partial \log(\mathbf{q})}{\partial \mathbf{q}} = \begin{pmatrix} 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{pmatrix} \tag{3.35}$$

Mit $\mathbf{q} \circ \mathbf{q_i^{0^{-1}}} = (1,0,0,0)^T$ und A.3 folgt:

$$\frac{\partial \mathbf{q_s} \circ \mathbf{q} \circ \mathbf{q}_i^{0^{-1}}}{\partial \mathbf{q_s}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.36}$$

q_{s_x} ist nicht abhängig von u, damit sind die partiellen Ableitungen $\frac{\partial x_x}{\partial u} = \frac{\partial w_x}{\partial u} = 0$. Die Ableitung $\frac{\partial \mathbf{q_s}}{\partial u}$ vereinfacht sich dann zu

$$\frac{\partial \mathbf{q_s}}{\partial u} = \begin{pmatrix} \frac{\partial w_y}{\partial u} w_x \\ \frac{\partial w_y}{\partial u} x_x \\ \frac{\partial y_y}{\partial u} w_x \\ \frac{\partial y_y}{\partial u} x_x \end{pmatrix} \tag{3.37}$$

Die einzelnen partiellen Ableitungen sind mit $\theta'_u = \frac{f_x}{f_x^2 + (u - x_0)^2}$

$$\frac{\partial w_y}{\partial u} = \cos'(\frac{\theta_u}{2}) = -\sin(\frac{\theta_u}{2})\frac{\theta'_u}{2} = -\frac{1}{2}\sin(\frac{\theta_u}{2})\frac{f_x}{f_x^2 + (u - x_0)^2} \tag{3.38}$$

$$\frac{\partial y_y}{\partial u} = \sin'(\frac{\theta_u}{2}) = \cos(\frac{\theta_u}{2})\frac{\theta'_u}{2} = \frac{1}{2}\cos(\frac{\theta_u}{2})\frac{f_x}{f_x^2 + (u - x_0)^2} \tag{3.39}$$

$\frac{\partial \mathbf{q_s}}{\partial v}$ wird dementsprechend berechnet.

$$\frac{\partial \mathbf{q_s}}{\partial v} = \begin{pmatrix} \frac{\partial w_x}{\partial v} w_y \\ \frac{\partial x_x}{\partial v} w_y \\ \frac{\partial w_x}{\partial v} y_y \\ \frac{\partial x_x}{\partial v} y_y \end{pmatrix} \tag{3.40}$$

$$\frac{\partial w_x}{\partial v} = \cos'(\frac{\theta_v}{2}) = -\sin(\frac{\theta_v}{2})\frac{\theta'_v}{2} = -\frac{1}{2}\sin(\frac{\theta_v}{2})\frac{f_y}{f_y^2 + (v - y_0)^2} \tag{3.41}$$

$$\frac{\partial x_x}{\partial v} = \sin'(\frac{\theta_v}{2}) = \cos(\frac{\theta_v}{2})\frac{\theta'_v}{2} = \frac{1}{2}\cos(\frac{\theta_v}{2})\frac{f_y}{f_y^2 + (v - y_0)^2} \tag{3.42}$$

Da die Inverse Tiefe ρ nun ebenfalls von u und v abhängt sind die dortigen Ableitungen ebenfalls zu bestimmen.

$$\rho = d\frac{\mathbf{m}_i^z}{f_x b} \tag{3.43}$$

mit

$$\mathbf{m}_i^z = \sqrt{1 + (\frac{u - x_0}{f_x})^2 + (\frac{v - y_0}{f_y})^2} \tag{3.44}$$

$$\frac{\partial \rho}{\partial u} = -\frac{d}{f_x b}\frac{1}{\sqrt{1 + (\frac{u-x_0}{f_x})^2 + (\frac{v-y_0}{f_y})^2}^3}\frac{u - x_0}{f_x^2} \tag{3.45}$$

$$\frac{\partial \rho}{\partial v} = -\frac{d}{f_x b}\frac{1}{\sqrt{1 + (\frac{u-x_0}{f_x})^2 + (\frac{v-y_0}{f_y})^2}^3}\frac{v - y_0}{f_y^2} \tag{3.46}$$

$$\frac{\partial \rho}{\partial d} = \frac{\mathbf{m}_i^z}{f_x b} \tag{3.47}$$

Kapitel 4

Testumgebung

4.1 Erzeugung der Testsequenzen

Für das Testen des Programmes ist eine Folge von Bildern (Sequenz) erforderlich. Die Bildersequenz kann aus einer echten Kamera stammen, aber auch künstlich erzeugt sein. Auf dieser Sequenz lässt man dann das Programm ablaufen. Dabei werden alle relevanten Daten mitgespeichtert. Für die Auswertung haben wir folgende Messgrößen verwendet:

- Position der Kamera
- Rotation der Kamera
- Kovarianz der Position der Kamera
- Positionen der Features

Dieser Vorgang wird jeweils für das IDFB-Modell und das entkoppelte IDFB-Modell ausgeführt. Für einen Vergleich zwischen diesen beiden Modellen ist ein einfacher Abgleich der gewonnenen Daten unzureichend. Denn ein Unterschied, in zum Beispiel der Kamerapostion, zeigt nicht an in welchem Modell die Kameraposition besser geschätzt wurde. Dazu ist es notwendig die wahre Kameraposition zu kennen.

Bei einer Aufzeichnung der Sequenz mit einer echten Kamera sind die wahren Positionen der Kamera und der Features nicht bekannt. Sie müssten durch direkte Messungen festgestellt werden. Aber die dabei auftretenden Messungenauigkeiten und der Aufwand (mehrere hundert Features) machen dies unmöglich.

Deswegen haben wir die Sequenzen künstlich erzeugt. Dazu haben wir Pov-Ray, ein Computergraphikprogramm zum Raytracing, benutzt. Um eine Sequenz zu erzeugen ist es nötig die Szene und eine Trajektorie anzugeben. Die Szene beschreibt den Aufbau und die Anordnung der Objekte im Raum. Die Trajektorie gibt an auf welcher Bahn sich die Kamera bewegen soll und in welche Richtung sie dabei blickt. Sie wird dann diskretisiert, sodass sich die Bahnkurve aus einzelnen Punkten zusammensetzt. Damit man daraus eine Sequenz erhält, wird nun für jeden Punkt der Trajektorie die Szene aus einem anderen Blickwinkel gerendert. Die Folge von Bildern ergibt hintereinander abgespielt bei 30 fps eine Kamerafahrt durch die Szene. Das ist die Sequenz. Abbildung 4.1 illustriert dies.

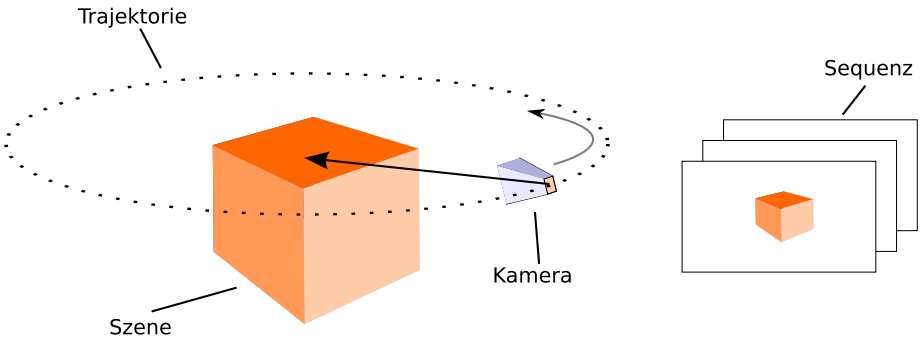


Abbildung 4.1: Beispiel der Würfelsequenz. Die Kamera bewegt sich auf der Trajektorie um den Würfel herum. Von den einzelnen Punkten der Trajektorie wird die Szene gerendert. Abgespielt mit 30 fps ergibt dies die Sequenz.

Auf der Sequenz werden, wie oben erwähnt, die beiden Programme mit den unterschiedlichen Modellen laufen gelassen. Die geschätzten Kamerapositionen und -rotationen ergeben die geschätzten Trajektorie. Man hat also nun eine wahre Trajektorie und die zwei geschätzten Trajektorien. Der Vergleich der Position und Rotation der Kamera ist damit möglich. Allerdings sind die wahren Positionen der Features noch unbekannt. Denn die Features werden erst bei dem Programmdurchlauf detektiert. Um die wahren Feature-Positionen zu bestimmen, werden von jedem detektierten Feature die Koordinaten in der Bildebene aufgezeichnet. Mittels dieser Koordinaten wird jeweils ein Strahl von der Kameraposition zur den Bildkoordinaten berechnet. Ausgehend von der wahren Trajektorie wird durch das Ray-Tracing der Schnittpunkt des Strahls mit dem erste Objekt auf seiner Bahn berechnet. Dieser Schnittpunkt ist die wahre Feature-Position. Für die Auswertung hat man also einmal die wahren Daten und jeweils die geschätzten Daten von den zwei verschiedenen Modellen.

4.2 Anpassung an reale Verhältnisse

Bisher ist die Sequenz ideal, dass heißt jedes Bild ist scharf und rauschfrei. Für eine realitätsnähere Simulation wird deswegen Bewegungsunschärfe (Motion-Blur) und Kamerarauschen eingeführt.

Für Motion-Blur wird ein Bild der Sequenz mit den nachfolgenden bzw. vorhergehenden Bildern kombiniert um eine künstliche Bewegungsunschärfe zu erzeugen. Dazu nutzen wir die Animationsoptionen von POV-Ray, um die beiden umliegenden Bilder der Sequenz auf das Bild in der Mitte zu addieren.

Zusätzlich wird noch ein künstliches Rauschen über die Bilder gelegt. Reales Kamerarauschen setzt sich aus einer Vielzahl von Rauscharten zusammen. Unter anderem gehören dazu:

- Photonenrauschen
- Dunkelstromrauschen
- Thermisches Rauschen
- Quantisierungsrauschen

- Ausleserauschen
- defekte Pixel (Hot Pixel)

Alle diese unterschiedlichen Rauscharten kann man im Wesentlichen durch folgende 3 Modelle simulieren.

- gaußverteitles Rauschen
- Impulsrauschen
- gleichverteiltes Rauschen (Uniform Noise)

Das Impulsrauschen und das gleichverteilte Rauschen sind meist geräteabhängig und reproduzierbar. Es ist also eine Rauschreduzierung möglich, indem man die Bilder duch eine entsprechende Maske korrigiert. Weiterhin ist das Impulsrauschen rechenaufwändig und könnte zu Nebeneffekten führen. Beispielsweise könnten einzelne starke Pixelausreißer das Ergebnis signifikant verfälschen, wenn ein Feature dadurch an einer anderen Position beobachtet wird. Das VSLAM-Programm reagiert empfindlich auf veränderte Feature-Positionen (siehe Abschnitt 5.4) und dadurch ist eine Ursachenerkennung schwierig und aufwändig. Eine Unterscheidung zwischen zufälligen Fehlern oder Modellfehlern in den Ergebnissen wäre kompliziert. Wir haben uns daher darauf beschränkt, zu jedem Bild gaußverteiltes Rauschen mit dem Erwartungswert $\mu = 0$ und der Varianz $\sigma = 2$ hinzuzufügen. Dies erlaubt unserer Ansicht nach eine ausreichende Überprüfung der These.

4.3 Testsequenzen

Insgesamt haben wir vier verschiedene Testsequenzen ablaufen lassen. Dabei wurden drei Szenen verwendet und diese mit vier verschiedenen Trajektorien kombiniert.

Die Sequenzen sind so aufgebaut, dass sie problemlos hintereinandern wiederholt werden können. Ein Schleife dauert jeweils 10 Sekunden und wiederholt sich 60-mal. So dass die Gesamtlänge einer Sequenz 600 Sekunden beträgt.

Zur Texturierung der Objekte wurde immer dieselbe Textur verwendet. Dabei ist zu beachten, dass die Textur genügend stark strukturiert ist, damit der Eckendetektor markante Punkte finden kann. Auch sollte die Textur nicht zu gleichmäßig sein, weil sonst möglicherweise markante Punkte an einer falschen verschobenen Stelle wiedererkannt werden. Abbildung 4.2 zeigt die von uns verwendete Textur.

Die folgenden Angaben zu den Sequenzenbeschreibungen gelten für ein rechtshändiges Koordinatensystem.

4.3.1 Würfelsequenz

Bei der Würfelsequenz besteht die Szene aus einem Würfel mit einer Kantenlänge von 2,5 m. Dessen geometrischer Schwerpunkt befindet sich genau im Koordinatenursprung des Gesamtsystems. Die Kameratrajektorie ist eine Kreisbahn mit dem Radius 4 m über dem Würfel. Das Zentrum der Kreisbahn ist 4 m über dem Koordinatenursprung, also an dem Punkt [0 m,0 m,4 m]. Die Blickrichtung

Abbildung 4.2: Die von uns verwendete Textur. Die hellen Steinflächen bieten kaum Struktur, so dass dort das VSLAM-Programm fast keine Features initialisiert.

der Kamera ist dabei immer um 45° nach unten zum Würfel geneigt und so gedreht, dass die Kamera immer zur z-Achse ausgerichtet ist. Der Startpunkt der Bewegung sind die Koordinaten [0 m,-4 m,4 m]. Die Kamera bewegt sich auf also kreisförmig um den Würfel herum mit Blick auf ihn. Das bedeudet, dass die Features nur auf einem kurzen Kreisabschnitt beobachtet werden können. Ansonsten sind sie bei dem restlichen Umlauf nicht sichtbar, da sie sich verdeckt auf einer abgewandten Würfelseite befinden.

4.3.2 Einfache Würfelsequenz

Es wird bei dieser Sequenz wieder der Würfel mit 2,5 m Kantenlänge verwendet. Zusätzlich ist dieser auf der x- und z-Achse noch um 45° gedreht. Die Kamera

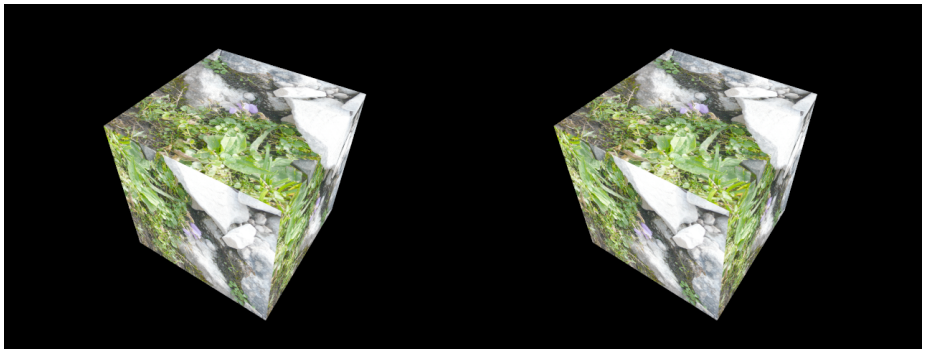


Abbildung 4.3: Eine Bild aus der Würfelsequenz.

bewegt sich wieder auf einer Kreisbahn 4 m über dem Koordinatenursprung. Dabei ist diesmal der Radius nur 0,3 m und die Kameradrehung um die x-Achse beträgt 81°. Wie in der normalen Würfelsequenz ist die Kamera zur z-Achse ausgerichtet. Startpunkt ist [0 m,-0,3 m,4 m]. Resultierend dreht sich die Kamera die ganze Sequenz über einer Würfelspitze. Dadurch können alle Features immer beobachtet werden und die Feature-Anzahl bleibt demzufolge konstant.

4.3.3 Wandssequenz

Diese Szene besteht aus 2 Ebenen, die mit einem Winkel von 90° aufeinander stehen. Also einem Boden und einer Wand dahinter. Vor der Wand befinden sich auf einer Länge von 2 m fünf verschiedene Gegenstände, die die Wand teilweise verdecken. Dazu gehören eine Kugel, ein Zylinder, ein Quader, ein Kegelstumpf und ein Prisma. Die Gegenstände sind zwischen 10 und 50 cm groß und befinden sich ungefähr 4 m vor der Wand. Die Trajektorie beschreibt eine Links/Rechtsbewegung 1 m über dem Boden und über eine Gesamtlänge von 2 m. Begonnen wird am rechten Punkt, dann bewegt sich die Kamera 2 m nach links und wieder 2 m nach rechts. Damit an den Umkehrpunkten keine zu hohen Beschleunigungen auftreten, wird die Geschwindigkeit dort jeweils auf Null herabgebremst. Die Entfernung der Kamera von der Wand beträgt ungefähr 5 m. Die Features am Rand der Bewegung werden nur einmal pro Durchlauf beobachtet. Features in der Mitte dagegen zweimal, beim Hin- und Zurückbewegen.

4.3.4 Tunnelsequenz

In der letzten Sequenz haben wir eine Bewegung durch einen Tunnel simuliert. Der Tunnel besteht aus einer Röhre (Zylinder) von 10 m Länge und 2 m Durchmesser. Die Röhre wird längs durch eine Ebene geteilt, so dass sich eine “Fahrbahn” ergibt, die von einer halbkreisförmigen Röhre umgeben ist. Auf der Fahrbahn befinden sich noch 5 Würfel mit der Kantenlänge 20 cm, die über die ganze Länge verteilt sind. Die Kamera bewegt sich in 20 cm Höhe über der Fahrbahn von dem Anfang der Röhre bis 2 m vor dem Ende der Röhre und wieder zurück zum Startpunkt. Die Bewegungen am Anfang und Ende sind dabei wieder abgebremst, um große Beschleunigungsänderungen zu vermeiden. Die Kamera blickt geradeaus nach vorne mit einer leichten Neigung von 11,25° nach unten. Features am Ende des Tunnels sind die ganze Zeit sichtbar. Sie können aber bei großer Entfernung nur schwer beobachtet werden. Umso weiter ein Feature am Anfang des Tunnels liegt, umso seltener wird es beobachtet, weil es sich die meiste Zeit hinter der Kamera befindet.

4.4 Parametereinstellungen

Die vier verschiedenen Testsequenzen wurden mit verschiedenen Parametereinstellungen des Programmes kombiniert. Es wurde für die Testläufe an drei verschiedenen Parametern Änderungen vorgenommen:

- Subpixelgenauigkeit
- volle Suche

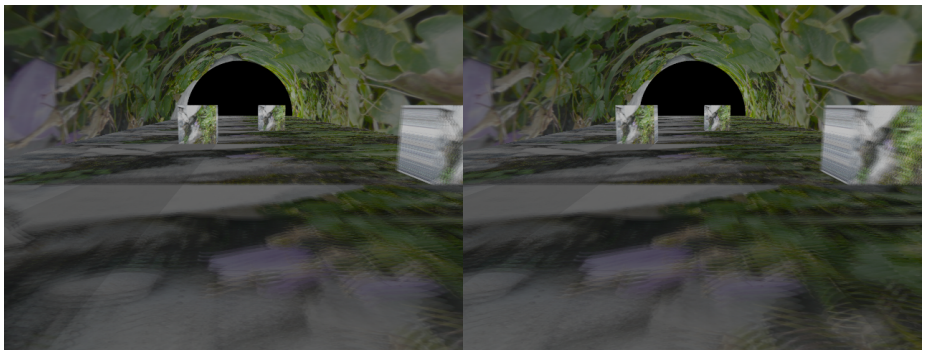


Abbildung 4.4: Eine Bild aus der Tunnelsequenz. Gut erkennbar ist die Bewegungsunschärfe am Boden, sowie die Disparität. Im linken Bild ist der Würfel rechts schon fast verschwunden, während er im rechten Bild noch gut sichtbar ist.

- minimale Fläche eines Features

Die Parametereinstellungen betreffen alle die Suche nach bekannten Feature-Positionen bei der erneuten Beobachtung.

Der erste veränderbare Parameter ist die Subpixelgenauigkeit. Bei aktivierter Subpixelgenauigkeit wird die Position eines Features, also der Mittelpunkt des Patches, nicht nur mit Pixelgenauigkeit genau bestimmt, sondern mit Subpixelgenauigkeit. Die Pixelgenauigkeit funktioniert wie in Abschnitt 2.2.1 beschrieben. Es wird das Patch mit der geringesten Abweichung vom Originalpatch ausgewählt und dessen Mittelpunkt als aktuelle Feature-Position bestimmt. Die Subpixelgenauigkeit behält diese Schritte bei. Es wird danach aber noch ein Paraboloid durch die Nachbarschaft des vorläufig ausgewählten Mittelpunktpixels gelegt. Dieser Paraboloid wird so angepasst, dass die Gesamtabweichung der Quadratsummen der einzelnen Nachbarschaftspatches möglichst gering ist. Das Minimum des Paraboloides ist dann die neue Feature-Position. Diese muss nicht mehr zwangsläufig auf exakt einem Pixel liegen, sondern kann auch einen Wert dazwischen annehmen.

Bei dem Parameter volle Suche wird das Patch in der vollen Unsicherheitsregion gesucht, die durch die Kovarianzmatrix angegeben wird. Bei ausgeschalteter vollen Suche, wird das Patch nur mit markanten Punkten innerhalb dieser Region verglichen. Die markanten Punkte werden wieder durch den Eckendetektor FAST ermittelt.

Der letzte Parameter ist die minimale Fläche. Jedes Feature hat durch das Patch eine Fläche. Die 21×21 Pixel messende Fläche des Patches wird in Abhängigkeit von der Entfernung und dem Blickwinkel der Kamera verändert (siehe Abschnitt 2.2.1). Durch diese kollineare Abbildung verändert sich auch der Flächeninhalt in Pixeln. Wenn die resultierende Pixelfäche dadurch zu groß oder zu klein ist, dann sind starke Verzerrungen zu erwarten. In diesem Fall versucht das VSLAM-Programm nicht das Feature zu finden, auch wenn es im Blickfeld liegt. Die minimalen Fläche eines Features gibt die untere Grenze an, darunter erfolgt kein Beobachtungsversuch. Dieses Verhalten ist zum Beispiel nützlich bei der Würfeldrehung. Ein Feature wird auf einer Würfelseite erkannt, durch das Wegdrehen wird die Fläche immer kleiner und verzerrter. Eine zuverlässige

Beobachtung wird immer schwieriger, deswegen ist es sinnvoll die Beobachtung ab einer bestimmten Fläche nicht mehr durchzuführen.

4.5 Weitere Einstellungen

Zusätzlich haben wir für ausgewählte Durchläufe die beobachteten Feature-Positionen der beiden Modelle gespeichert. Die Positionen wurden dann jeweils dem Programm mit dem anderen Modell als Eingabe geliefert. So dass die Programme keinen Einfluss mehr auf die Position der initialen Features hatten. Die Feature-Positionen waren also im gesamten Durchlauf fest vorgegeben.

Die Testläufe wurden abgebrochen, wenn das Programm mehr als 450 Features aktiv im EKF verwaltete. Denn dies deutet daraufhin, dass das Programm der Bahnkurve nicht mehr stabil gefolgt war und deswegen mehr Features als notwendig initialisiert wurden.

4.6 Testvergleichsmethoden

Die Daten wurden verglichen indem die Abweichungen der geschätzten Daten mit den wahren Daten verglichen wurden. Dies wird für jeden Zeitschritt ausgeführt. Bei einer komplett durchgelaufenen Sequenz gibt es also 18000 Messwerte für jede der folgenden Messgrößen.

Für die Abweichung der geschätzten Kamperaposition $\hat{c}$ von der wahren Kameraposition c haben wir die Euklidische Distanz berechnet.

$$\sqrt{\sum_{j=1}^{3}(\hat{c}-c)_j^2} \tag{4.1}$$

Der j-te Index beschreibt die x-,y- und z-Koordinaten.

Um die Unsicherheit in Form der Positionskovarianz $\sum$ einzubeziehen, haben wir die Mahalanobis-Distanz ausgewertet. Diese ist eine Erweiterung der Euklidischen Distanz. Eine geschätzte Position hat zusätzlich nicht nur dann eine größere Distanz, wenn sie sich weiter weg von der wahren Position befindet, sondern es werden auch Positionen die als zu sicher bewertet wurden mit einem höheren Distanzwert belegt. Abbildung 4.5 veranschaulicht den Zusammenhang.

$$(\hat{c}-c)^T \sum^{-1} (\hat{c}-c) \tag{4.2}$$

In den Diagrammen zur Mahalanobis-Distanz findet sich noch die Angabe des Quantils zum Signifikanzniveau 5%. Das Quantil beschreibt den Wert der Verteilung, unter dem 95% aller gemessenen Werte liegen sollten. Da die Mahalanobis-Distanz Chi-Quadrat-verteilt ist und die Anzahl der Freiheitsgrade drei ist, weil drei Parameter geschätzt werden (Kamerapostionen x,y,z), kann der Wert des Quantils aus der Tabelle der Quantile der Chi-Quadrat-Verteilung zu 7,81 abgelesen werden.

Eine weitere Messgröße war die Rotation. Die Abweichung der geschätzten Rotation $\hat{R}$ von der wahren Rotation R haben wir durch die Frobenius-Norm

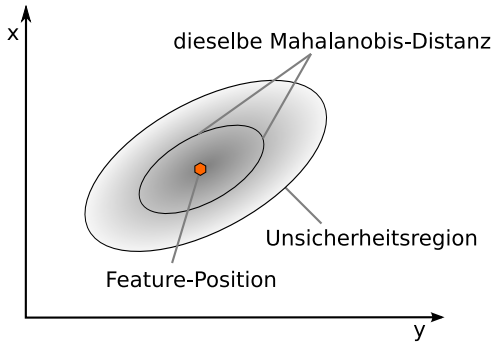


Abbildung 4.5: Visualisierung der Mahalanobis-Distanz im zweidimensionalen Raum. Es haben nicht mehr die Punkte mit dem gleichen Abstand dieselbe Distanz, sondern alle Punkte die sich auf einer Ellipse um das Feature herum befinden. Im dreidimensionalen Raum ist dies ein Ellipsoid.

bestimmt. (I ist dabei die Einheitsmatrix)

$$\|I - \hat{R}^T R\|_F \quad \text{mit} \tag{4.3}$$
$$\|A\|_F = \sqrt{\sum_{i,j} a_{i,j}^2}$$

Ein weitere Vergleich fand zwischen den Genauigkeiten der geschätzten Feature-Positionen statt. Dazu wurde der Einzelfehler einer geschätzten Feature-Position $\hat{y_i}$ gegenüber der wahren Feature-Position y_i durch die quadrierte euklidische Norm bestimmt. Für einen Durchschnittsgesamtfehler über die Anzahl n der Features haben wir dann alle Einzelfehler aufsummiert und durch die Anzahl n der Features geteilt.

$$\frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{3} (\hat{y_i} - y_i)_j^2 \tag{4.4}$$

Kapitel 5

Testergebnisse

Wir haben den Modellvergleich in drei verschiedenen Bereichen durchgeführt. Diese sind die Genauigkeit der Schätzung, die Unsicherheit der Positionsschätzung und die Genauigkeit der Features. Als weiteres haben wir noch die Auswirkung fester Feature-Positionen untersucht.

5.1 Genauigkeit

In diesem Abschnitt möchten wir auf den Vergleich der Genauigkeit zwischen nichtenkoppeltem und entkoppeltem Modell eingehen. Es hat sich gezeigt, dass Translation und Rotation eng verbunden sind, so dass die Auswertung der Rotation kaum neue Erkenntnisse gebracht hat. Wir beschränken uns daher größtenteils auf die Angabe der Translationsgenauigkeiten.

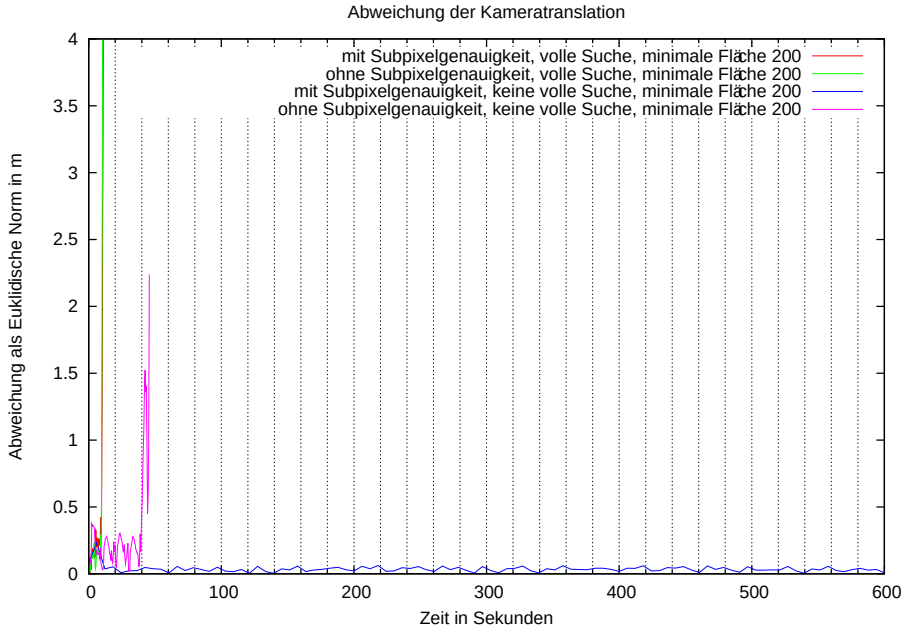
5.1.1 Würfelsequenz

Die Würfelseqeunz stellte sich als eine der schwierigsten Sequenzen heraus. Keines der Modelle konnte über alle Parametereinstellungen hinweg stabile Verfolgungen liefern. Die Ursache dafür ist der Schleifenschluss nach einer Umdrehung um den Würfel herum. Bereits beobachtete Features sind eine ganze Umdrehung nicht sichtbar, in dieser Zeit kann die geschätzte Kameraposition merkbar von der vorgegebenen Postion abweichen. Bei einer erneuten Beobachtung bereits bekannter Features wurde das System instabil. Eine mögliche Ursache könnten eine zu starke Korrektur der Kameraposition und der Features sein, um den Abweichungsfehler zu korrigieren.

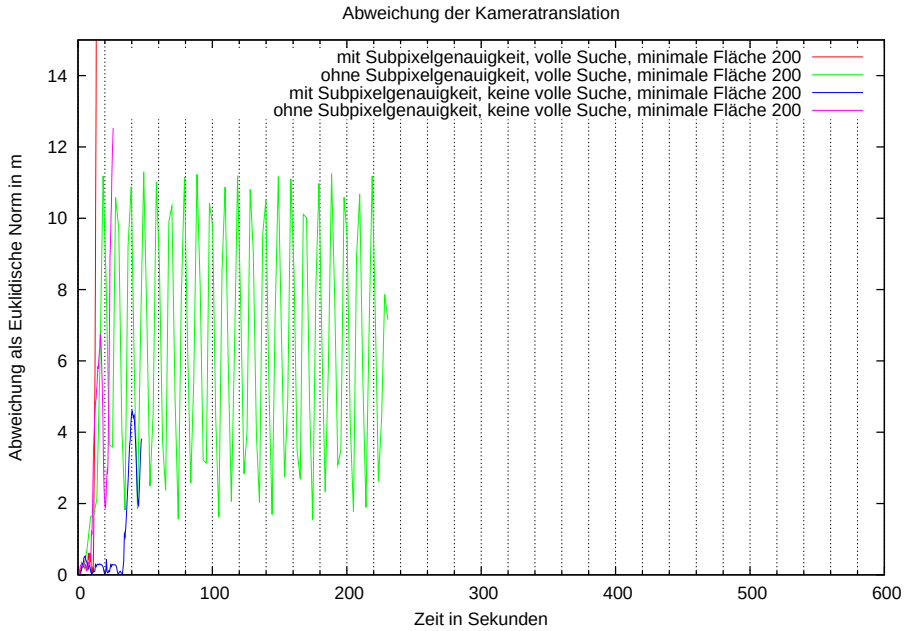
Der Vergleich zwischen den Modellen beschränkt sich daher auf die unterschiedliche Anzahl der erfolgreich absolvierten Durchläufe beziehungsweise die Zeitdauer bis zum instabil werden.

In Abbildung 5.1 sieht man, dass beide Modelle die Bahnkurve nicht lange verfolgen können. Es gibt nur einen Fall (IDFB-Modell mit Subpixelgenauigkeit ohne volle Suche) wo die Sequenz erfolgreich durchgelaufen ist. Die Ergebnisse scheinen hier sehr zufällig zu sein und lassen keine klaren Aussagen über die Modellqualitäten zu.

Die starke Schwankung beim Testdurchlauf des entkoppelten IDFB-Modells ohne Subpixelgenauigkeit mit voller Suche ist darauf zurückzuführen, dass das



(a) nichtentkoppeltes IDFB-Modell



(b) entkoppelte IDFB-Modell

Abbildung 5.1: Würfelsequenz
Beide Modelle versagen fast immer bei der Kameraschätzung. Nur in einem Fall läuft die Testsequenz erfolgreich durch.

VSLAM-Programm die Bahnkurve nach der ersten Umrundung zwar noch einige Zeit verfolgen konnte. Aber diese Bahnkurve war verschoben und verdreht zur vorgegebenen Bahnkurve. Auch in diesem Testfall waren die Schätzungen nach 10 Sekunden in ihren Abweichungen zu groß und der Durchlauf wurde abgebrochen.

5.1.2 Einfache Würfelsequenz

Da die normale Würfelsequenz zu unklaren Ergebnissen geführt hat, haben wir sie in ihrer Komplexität reduziert. Dadurch das der Blick immer auf eine Würfelspitze gerichtet ist, werden die beobachteten Features immer erkannt und ihre Anzahl bleibt konstant. Dies führt zu einem besonders guten Verfolgen der Bahnkurve.

In dieser Sequenz waren die Abweichungen in der Kameraverfolgung zwischen beiden Modellen minimal. Weder in der Translation (Abbildung 5.3) noch Rotation (Abbildung 5.4) zeigten sich signifikante Unterschiede. Eine mögliche Ursache ist, dass gerade die Einfachheit der Testsequenz dafür sorgt, dass beide Modelle ähnliche Ergebnisse zeigen. Die weiteren Freiheitsgrade, die das entkoppelte IDFB-Modell erlaubt sind in dieser Testsequenz nicht notwendig. Der höhere Aufwand der Berechnung ist damit auch nicht erforderlich.

Die Subpixelgenauigkeit führte zu einer besseren Genauigkeit bei beiden Modellen, bevorzugte dabei aber weder das eine noch das andere Modell.

5.1.3 Wandsequenz

In der Wandsequenz waren die Unterschiede zwischen den Modellen sehr deutlich. Allerdings zeigen die Ergebnisse in keine klare Richtung. Als entscheidender Parameter hat sich die Subpixelgenauigkeit herausgestellt. Abbildung 5.5 zeigt, dass das entkoppelte IDFB-Modell bei ausgeschalteter Subpixelgenauigkeit eine höhere Genauigkeit besitzt. Allerdings kehrt sich diese Beobachtung komplett um, wenn Subpixelgenauigkeit eingeschaltet wird (Abbildung 5.6).

5.1.4 Tunnelsequenz

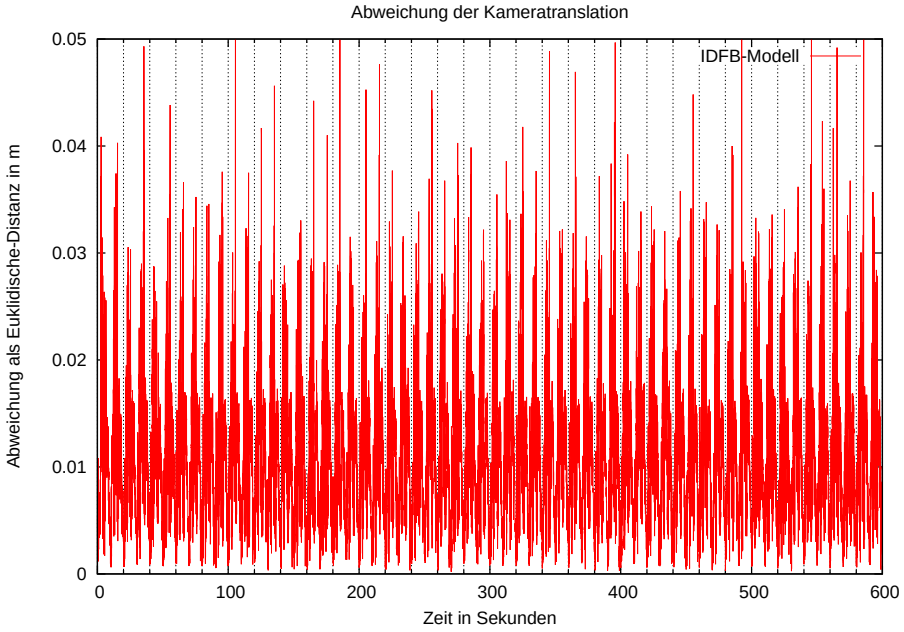
Wie in der Würfelsequenz waren hier erfolgreiche Durchläufe selten. Die Experimente in Abbildung 5.9 zeigen aber bessere Resultate für das entkoppelte IDFB-Modell. Dieses ermöglichte zwei erfolgreich absolvierte Testläufe im Gegensatz zu einem beim normalen IDFB-Modell. Auch die vorzeitig abgebrochen Testläufe liefen jeweils mindestens 80 Sekunden länger bevor sie instabil wurden.

Der Vergleich der Genauigkeit über die gesamte Durchlaufzeit zwischen den beiden Modellen ist nur in einer Parametereinstellung (Abbildung 5.11) möglich. Das entkoppelte IDFB-Modell schätzt die Kamerabewegung hierbei besser.

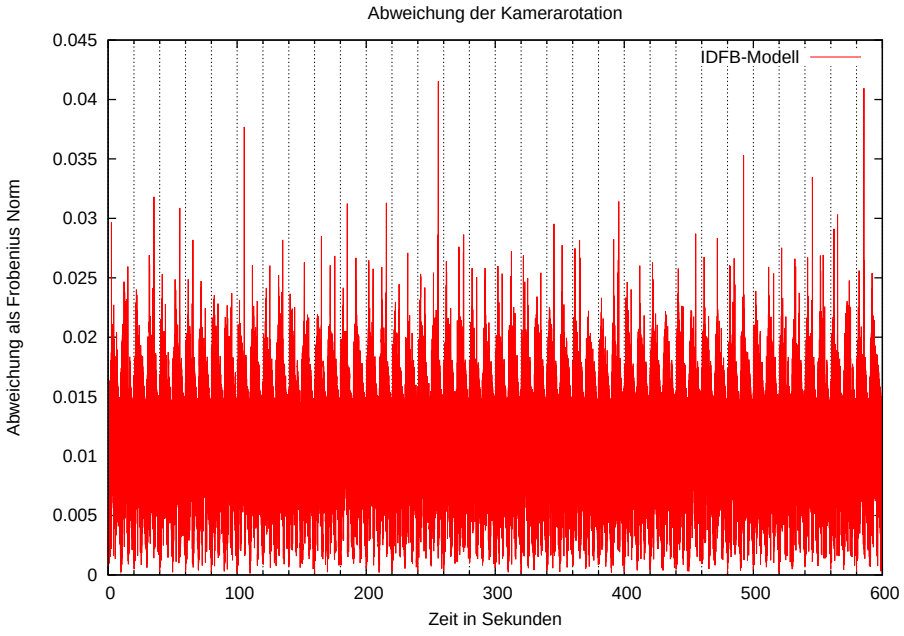
Die Tunnelsequenz ist somit die einzige Testsequenz in der das entkoppelte IDFB-Modell bei der Schätzung der Kameraposition klar besser ist.

5.2 Unsicherheit

Die Unsicherheit haben wir in Form der Mahalanobis-Distanz gemessen (Abschnitt 4.1). Die einfache Würfelsequenz (Abbildungen 5.13) zeigt wieder die

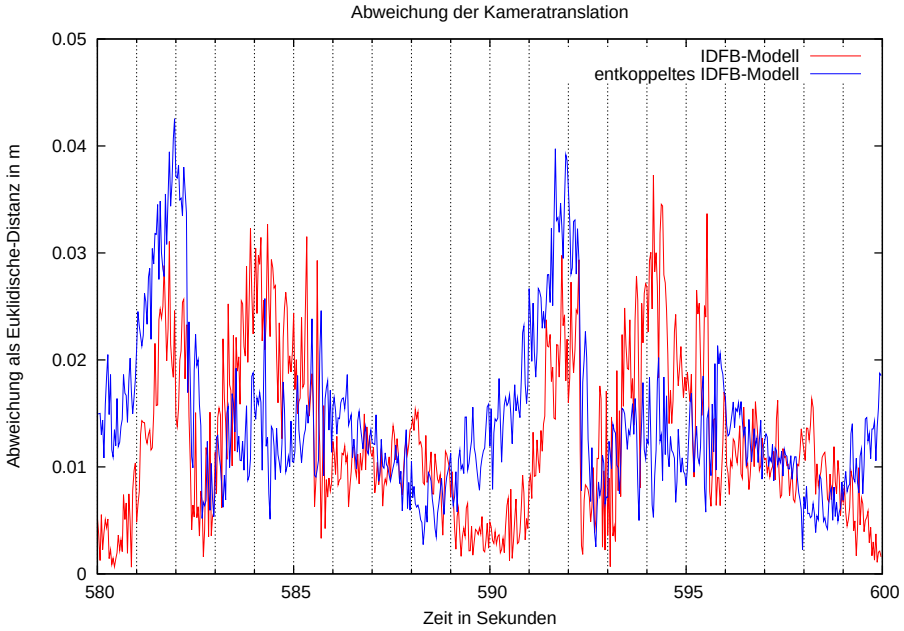


(a) Translation

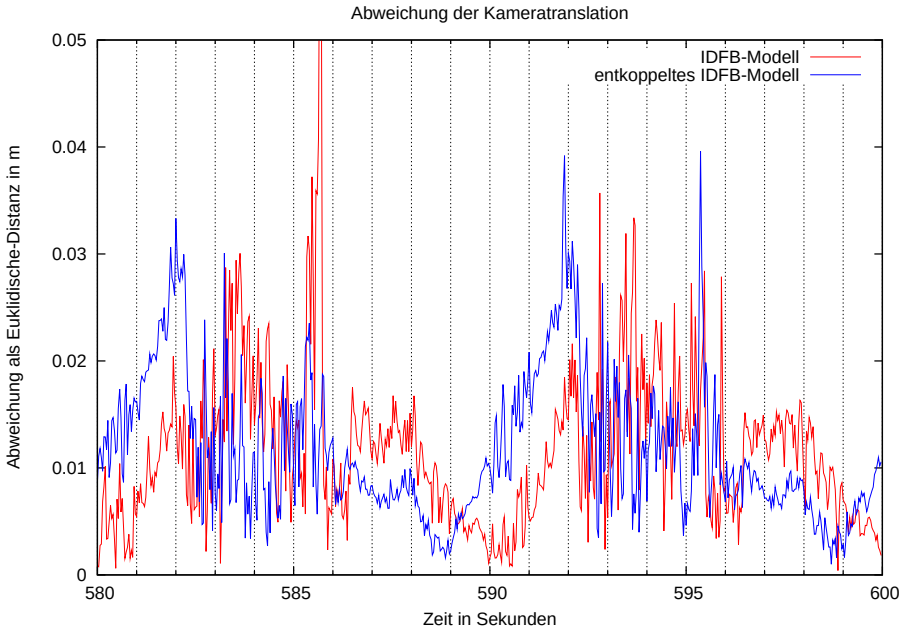


(b) Rotation

Abbildung 5.2: Einfache Würfelsequenz mit dem nichtentkoppelten Modell bei aktivierter Subpixelgenauigkeit und voller Suche
Die anderen Testläufe zeigen ein ähnliches Bild. Ein Vergleich der Modelle ist bei Darstellung der Messgrößen über die gesamte Länge schwer. Deswegen haben wir die letzten 20 Sekunden vergrößert dargestellt.

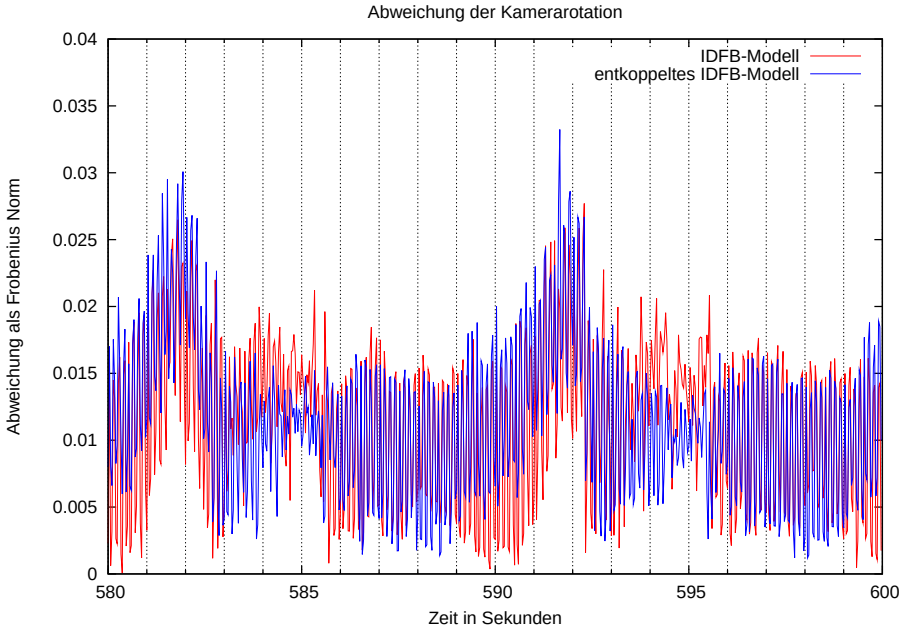


(a) ohne Subpixelgenauigkeit

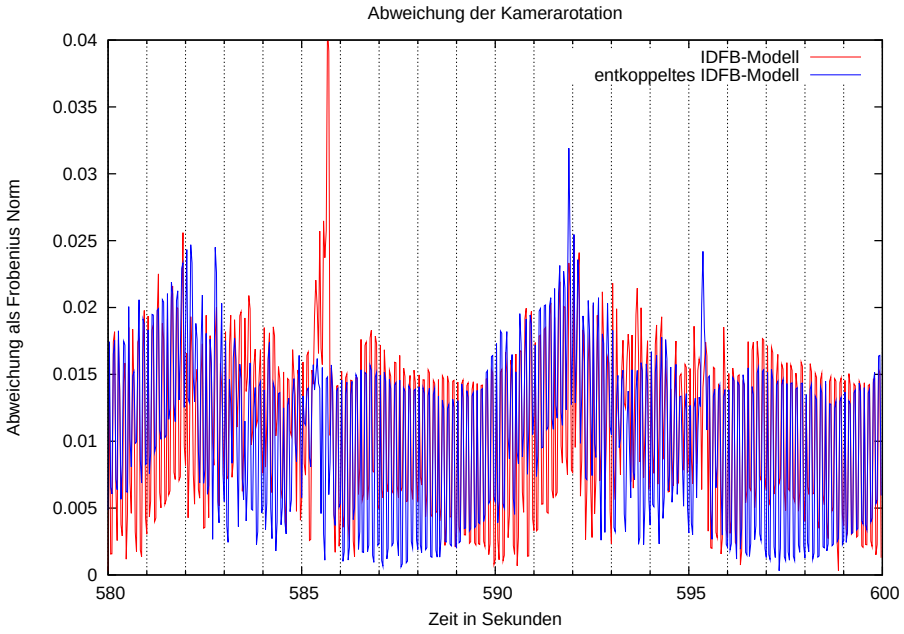


(b) mit Subpixelgenauigkeit

Abbildung 5.3: Einfache Würfelsequenz mit voller Suche, vergrößerte Ansicht
Die Abweichung der Kameratranslation zwischen den Modellen ist minimal, auch die Subpixelgenauigkeit verändert kaum etwas an dem Ergebnis. Die Kamera wird jeweils sehr gut geschätzt.

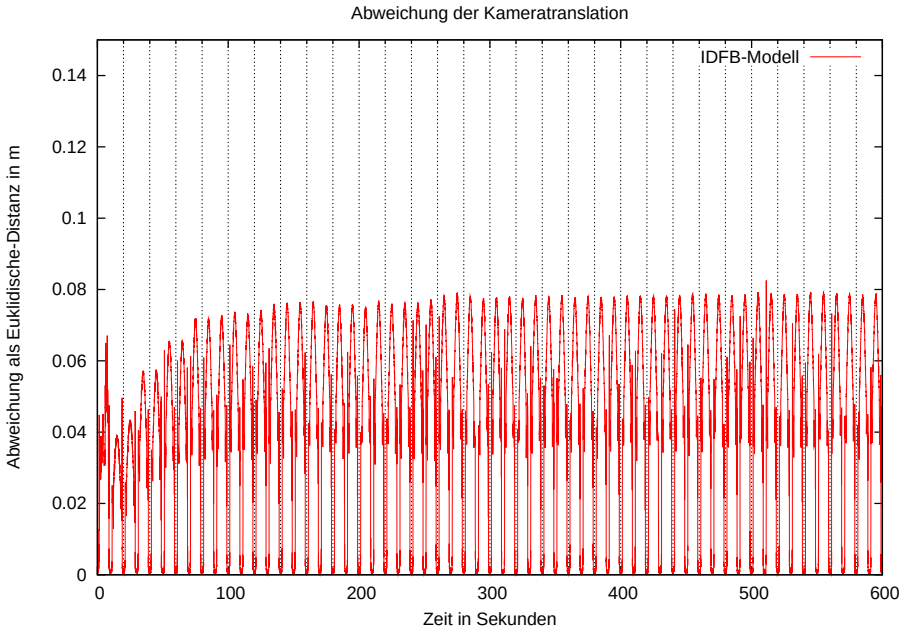


(a) ohne Subpixelgenauigkeit

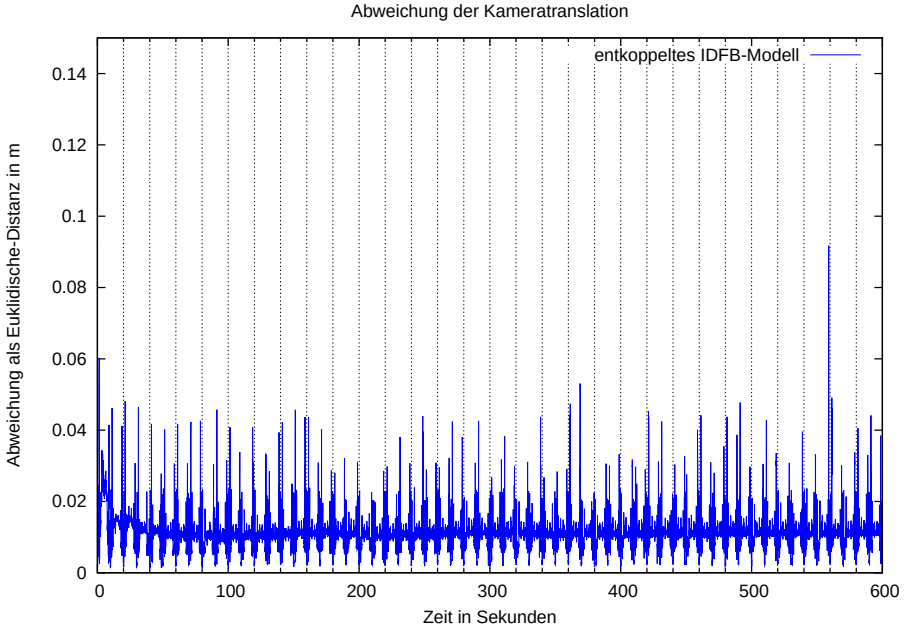


(b) mit Subpixelgenauigkeit

Abbildung 5.4: Einfache Würfelsequenz mit voller Suche, vergrößerte Ansicht
Die Abweichung der Kamerarotation liefert keine neuen Informationen gegenüber der Abweichung der Kameratranslation. Die Kameraschätzung ist sehr gut, unabhängig vom verwendeteten Modelle oder den verwendeten Parametereinstellungen.

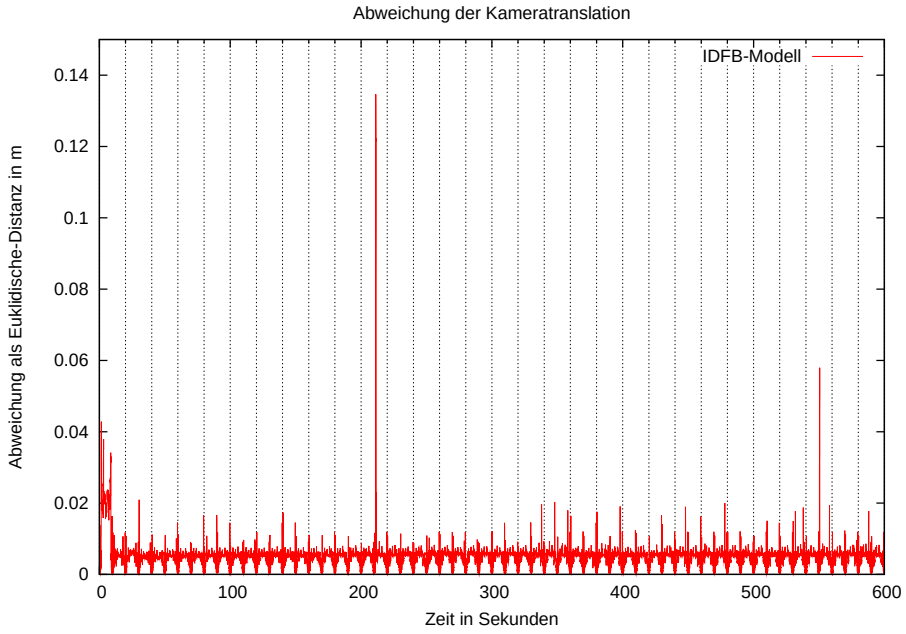


(a) nichtentkoppeltes IDFB-Modell

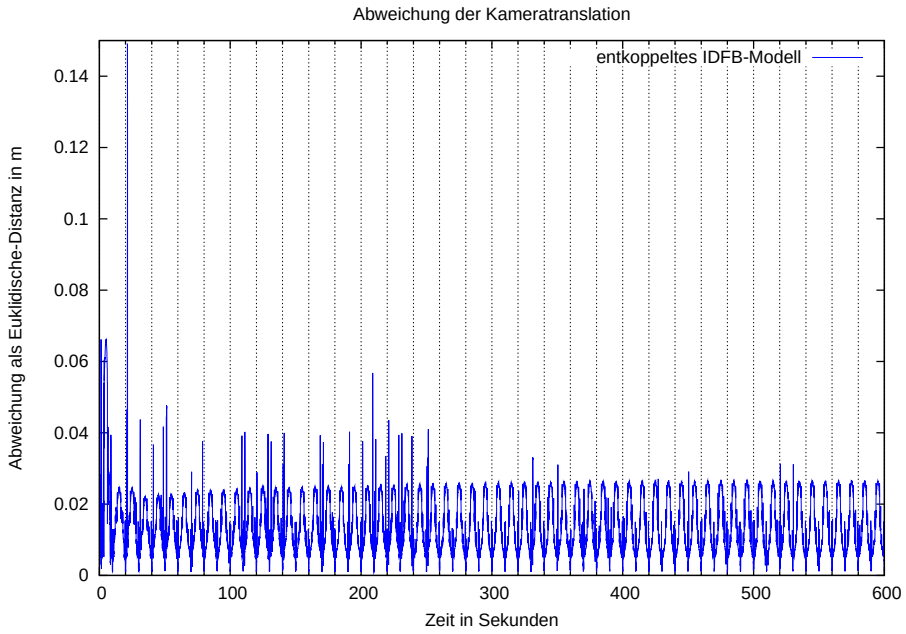


(b) entkoppeltes IDFB-Modell

Abbildung 5.5: Wandsequenz ohne Subpixelgenauigkeit mit voller Suche
Die Abweichungen des nichtentkoppelten Modells sind stärker.

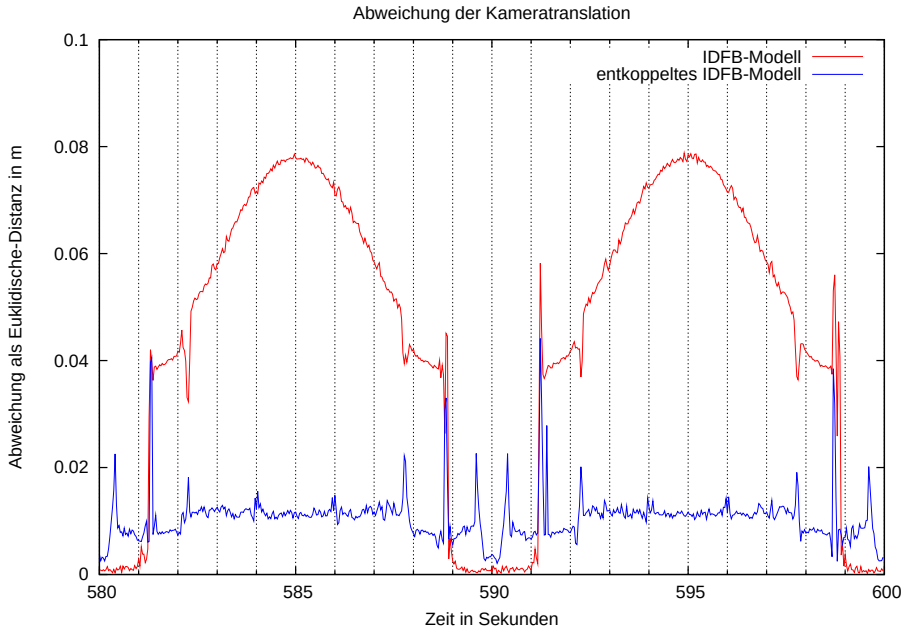


(a) nichtentkoppeltes IDFB-Modell

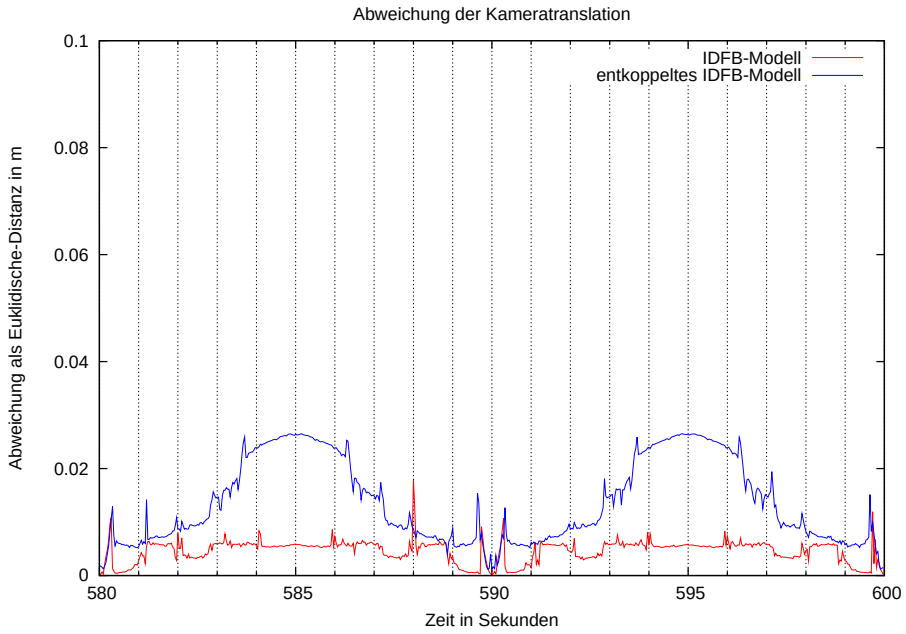


(b) entkoppeltes IDFB-Modell

Abbildung 5.6: Wandsequenz mit Subpixelgenauigkeit mit voller Suche
Bei aktivierter Subpixelgenauigkeit ist die Genauigkeit des nichtentkoppelten Modells besser.

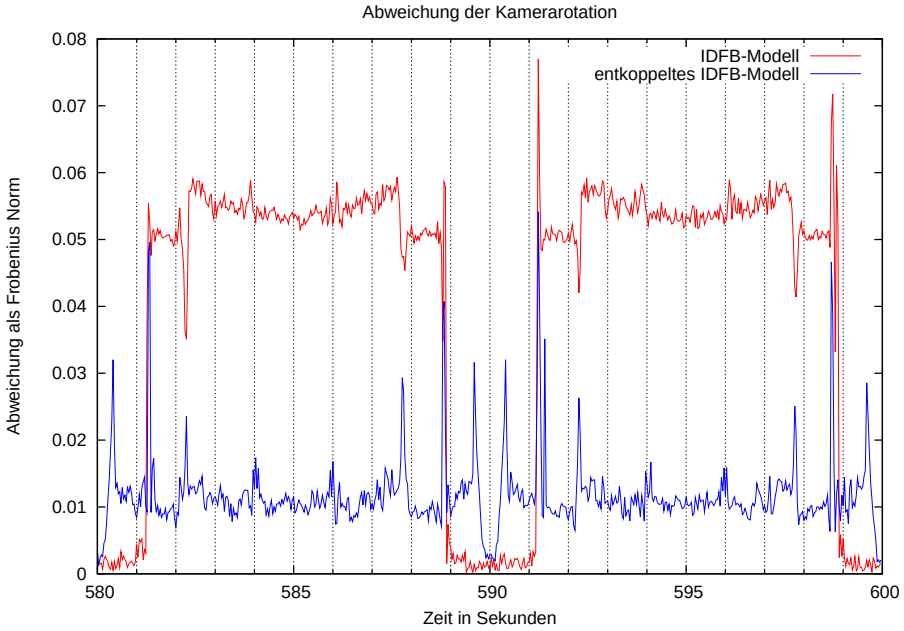


(a) ohne Subpixelgenauigkeit

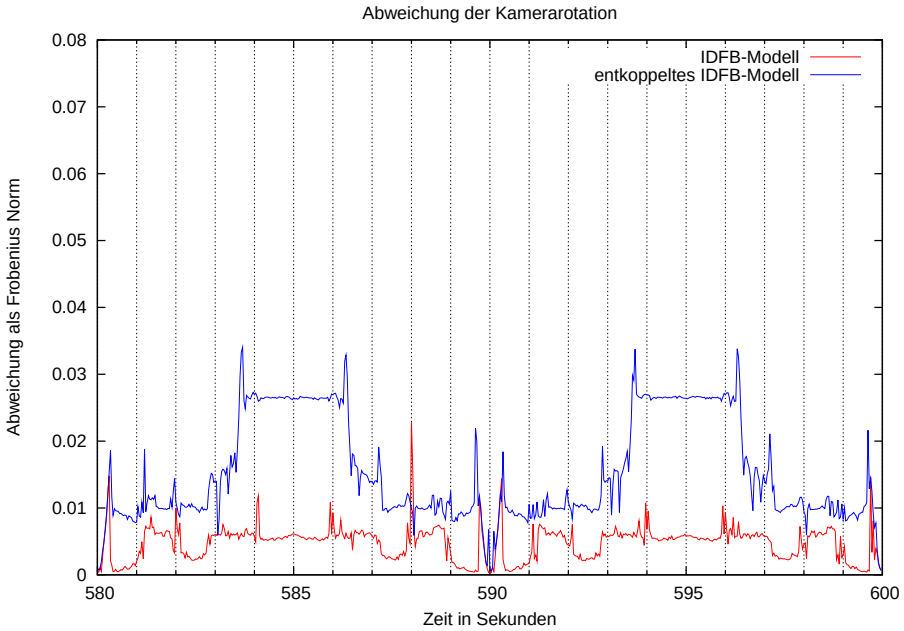


(b) mit Subpixelgenauigkeit

Abbildung 5.7: Wandsequenz Translation mit voller Suche, vergrößerte Ansicht In der vergrößerten Ansicht sieht man im direkten Vergleich den Einfluss der Subpixelgenauigkeit. Das nichtenkoppelte Modell profitiert durch die Aktivierung enorm, während das entkoppelte Modell sogar etwas an Gesamtgenauigkeit verliert.

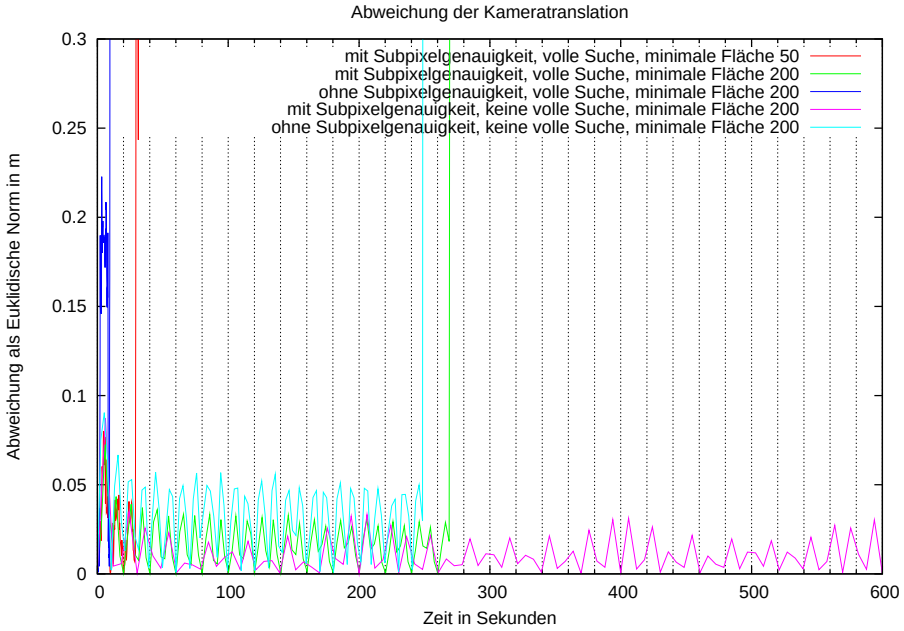


(a) ohne Subpixelgenauigkeit

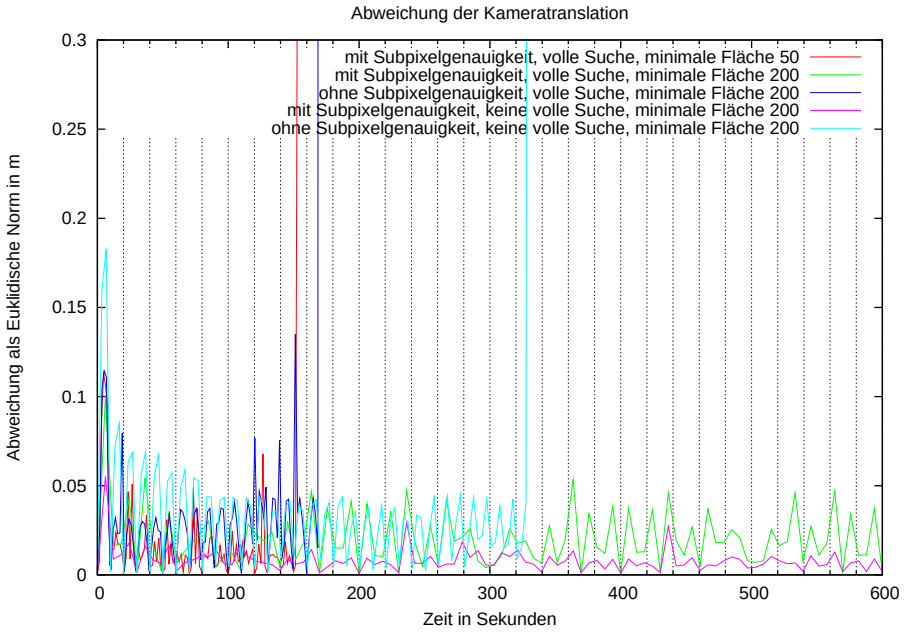


(b) mit Subpixelgenauigkeit

Abbildung 5.8: Wandsequenz Rotation mit voller Suche, vergrößerte Ansicht
Die Rotation zeigt ein ähnliches Bild wie die Translation. Wiederum ist bei Subpixelgenauigkeit das nichtentkoppelte Modelle genauer.

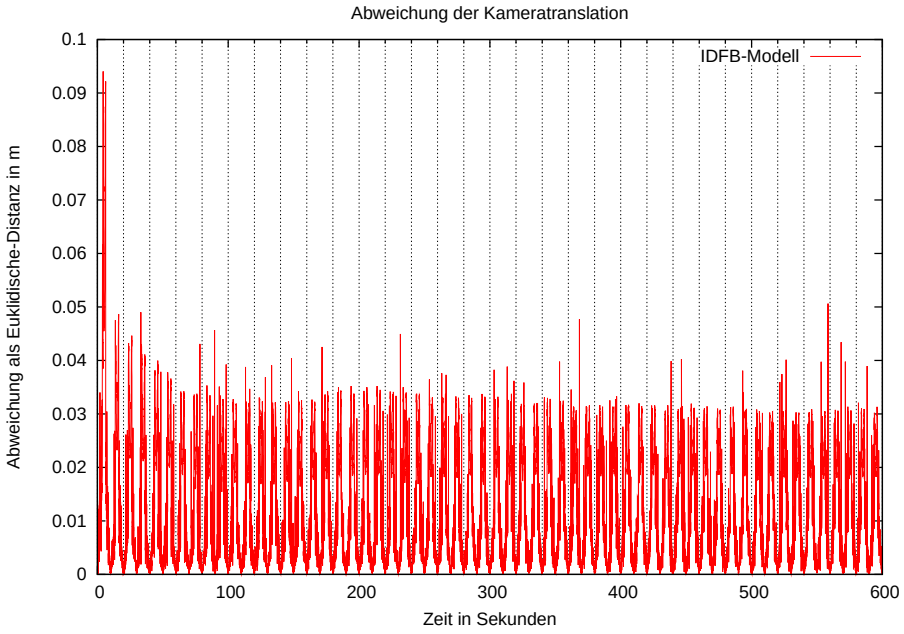


(a) nichtentkoppeltes IDFB-Modell

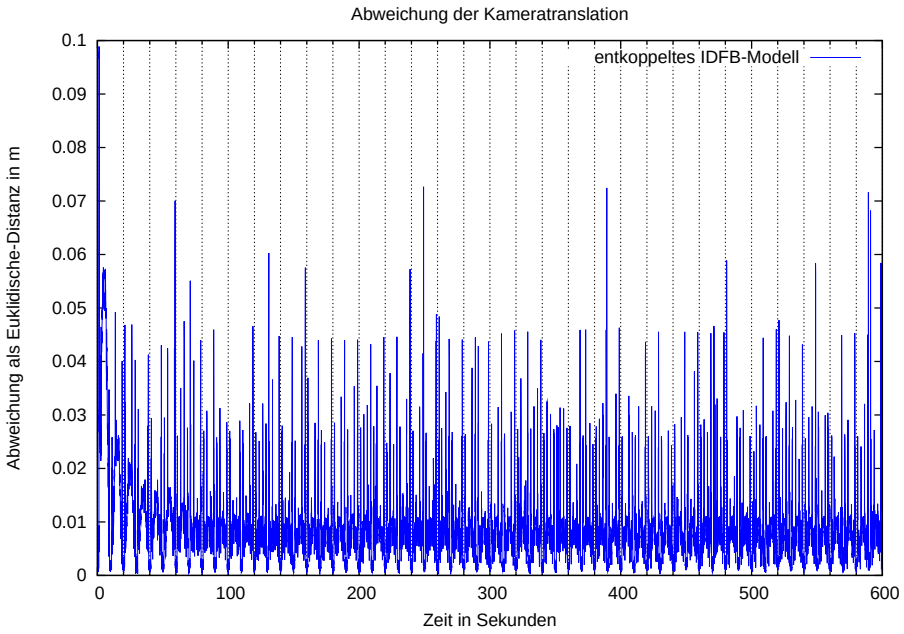


(b) entkoppeltes IDFB-Modell

Abbildung 5.9: Tunnelsequenz
Die Tunnelsequenz ist eine anspruchsvolle Testsequenz. Nur wenige Testdurchläufe sind erfolgreich. Das entkoppelte Modell kann die Kameraverfolung immer länger gegenüber dem nichtentkoppelten Modell durchführen.



(a) nichtentkoppeltes IDFB-Modell



(b) entkoppeltes IDFB-Modell

Abbildung 5.10: Tunnelsequenz mit Subpixelgenauigkeit ohne volle Suche
Das entkoppelte IDFB-Modell zeigt Vorteile in der Genauigkeit. In der vergrößerten Ansicht 5.11 ist dies besser zu erkennen.

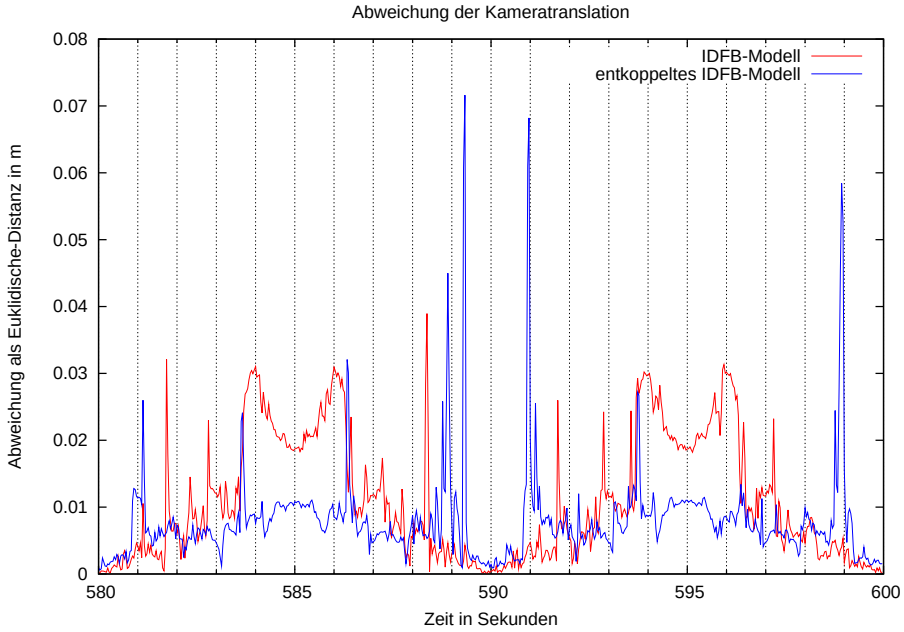


Abbildung 5.11: Tunnelsequenz mit Subpixelgenauigkeit ohne volle Suche
Trotz höherer einzelner Ausschläge ist insgesamt das entkoppelte Modell genauer

besten Ergebnisse. Die Mahalanobis-Distanz war hier stabil, lag aber oft über dem Quantil zum Signifikanzniveau 5%.

Entgegensätzliche Aussagen liefert die Wandsequenz. Ohne Subpixelgenauigkeit (Abbildung 5.14) steigt bei dem nichtentkoppelten IDFB-Modell die Mahalanobisdistanz während des Durchlaufes stark an. Demgegenüber zeigt die Positionsmessungen aber kein Ansteigen (Abbildung 5.5). Das deutet daraufhin, dass sich das Programm immer sicherer über die Position wird. Das entkoppelte Modell ist hier überlegen, das Quantil wird aber auch hier in mehr als 5% der Fälle überschritten. Bei eingeschalteter Subpixelgenauigkeit (Abbildung 5.15) kann die Abweichung der Mahalanobis-Distanz bei dem nichtentkoppelten IDFB-Modell reduziert werden und liegt sogar unter der vom entkoppelten IDFB-Modell. Beide Modelle sind aber immer noch nicht vertrauenswürdig, da sie den Signifikanztest nicht bestehen.

In der einzig auswertbaren Tunnelsequenz (Abbildungen 5.17 und 5.18) erfüllt das entkoppelte IDFB-Modell nur knapp nicht den Signifikanztest. Insgesamt sind 94,8% der Werte unter dem Quantil von 7,81. Im Gegensatz dazu steigt beim nichtentkoppelten IDFB-Modell die Mahalanobis-Distanz öfter über das Quantil.

5.3 Feature-Genauigkeit

Die Feature-Genauigkeit, wie in der Formel 4.4 beschrieben, ist ein sehr wichtiger Indikator für die Güte der Schätzung. Denn sie ist ein Maß für die Genauigkeit der erstellten Karte.

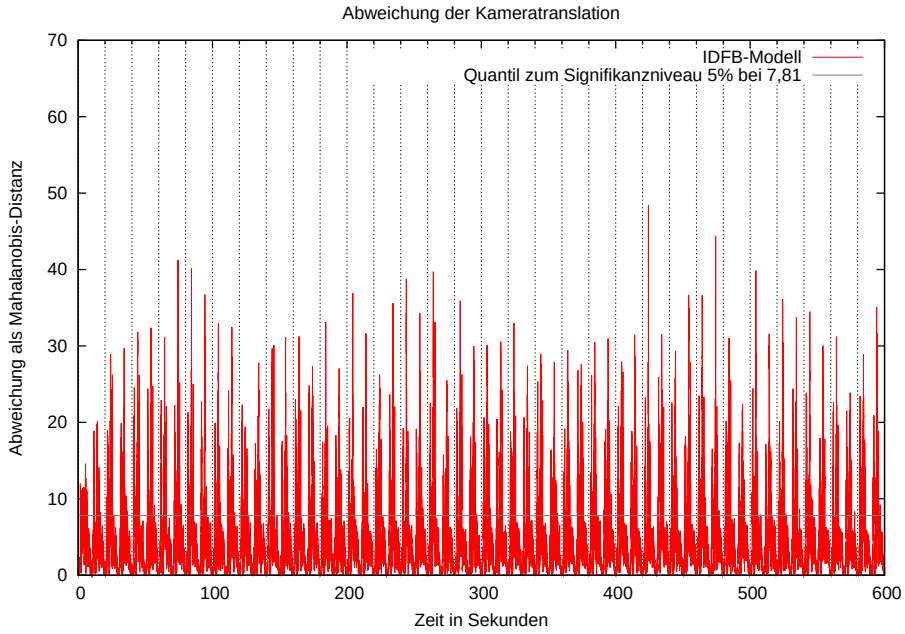
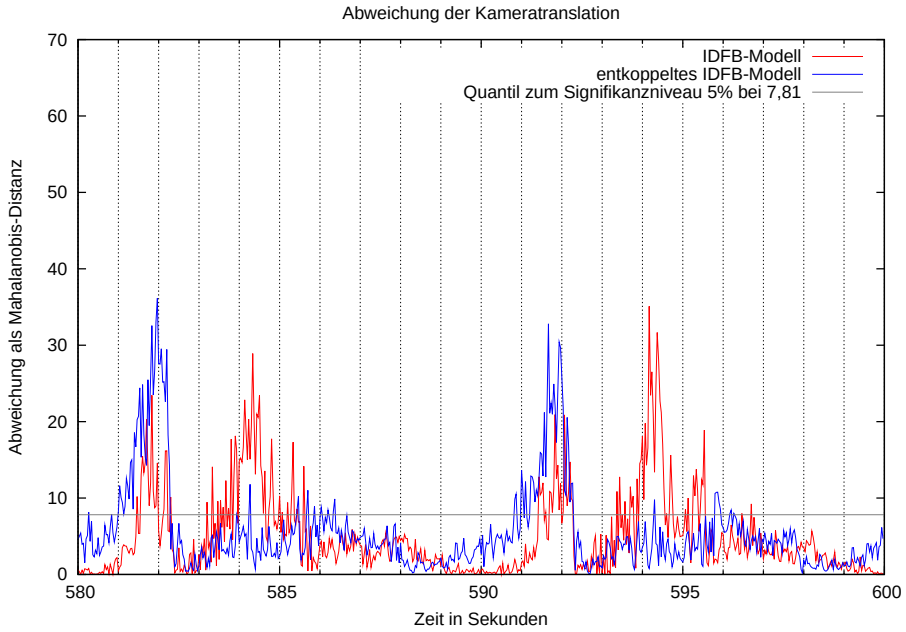
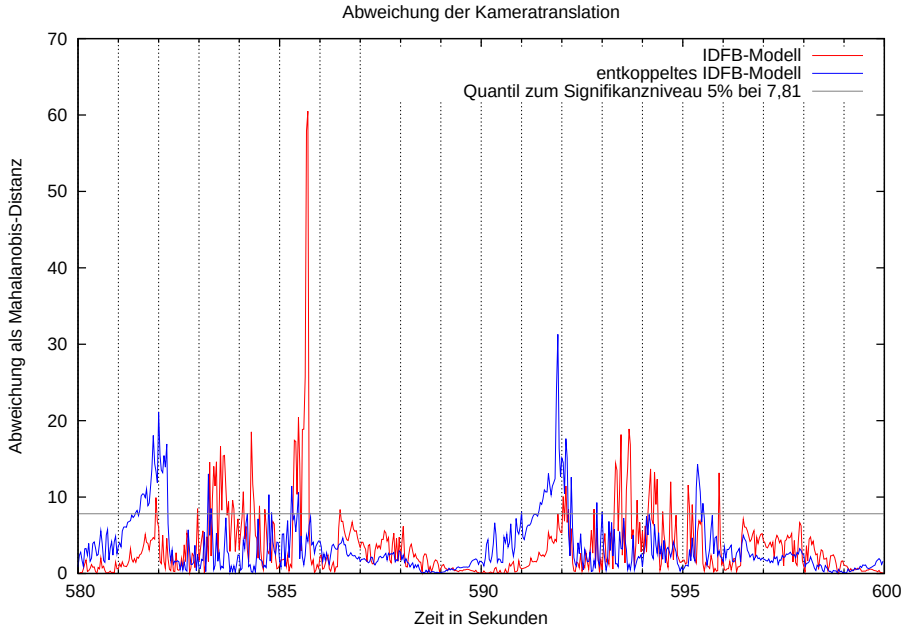


Abbildung 5.12: Einfache Würfelsequenz ohne Subpixelgenauigkeit mit voller Suche

Wie schon bei der Euklidischen Distanz sind auch hier die Darstellungen über die gesamte Länge unübersichtlich. Die andere Abbildungen sehen wiederum ähnlich aus. Die vergrößerten Ansichten lassen einen besseren Vergleich zu.

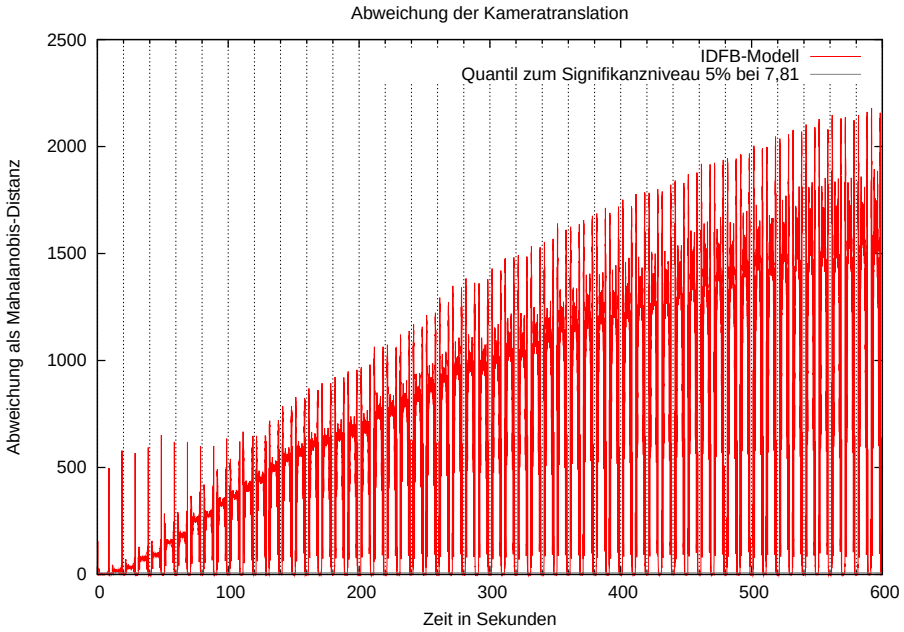


(a) ohne Subpixelgenauigkeit

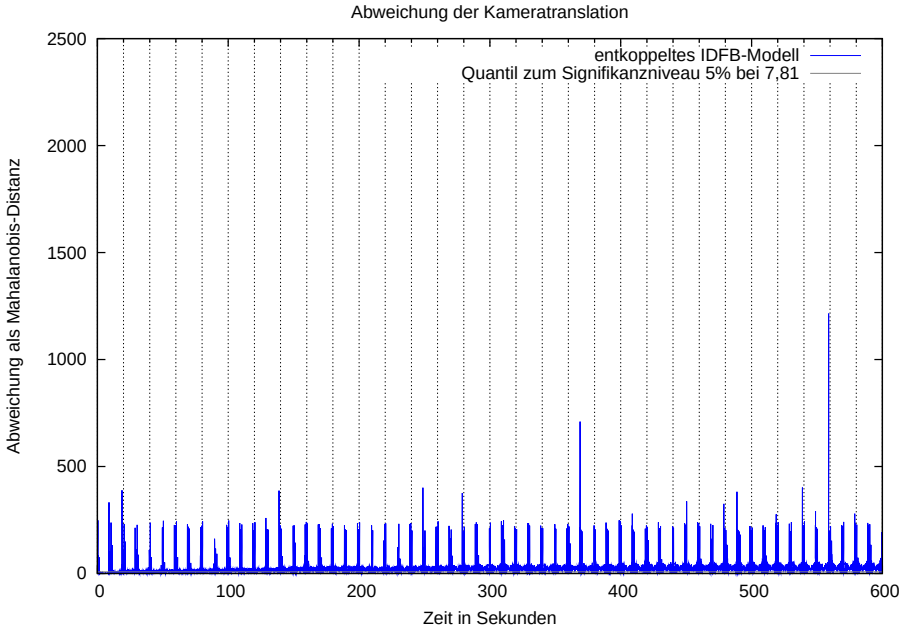


(b) mit Subpixelgenauigkeit

Abbildung 5.13: Einfache Würfelsequenz mit voller Suche, vergrößerte Ansicht
Zwischen den Modellen ist der Unterschied in der Mahalanobis-Distanz minimal. Durch das Einschalten der Subpixelgenauigkeit verringert sich bei beiden Modellen die Mahalanobis-Distanz.

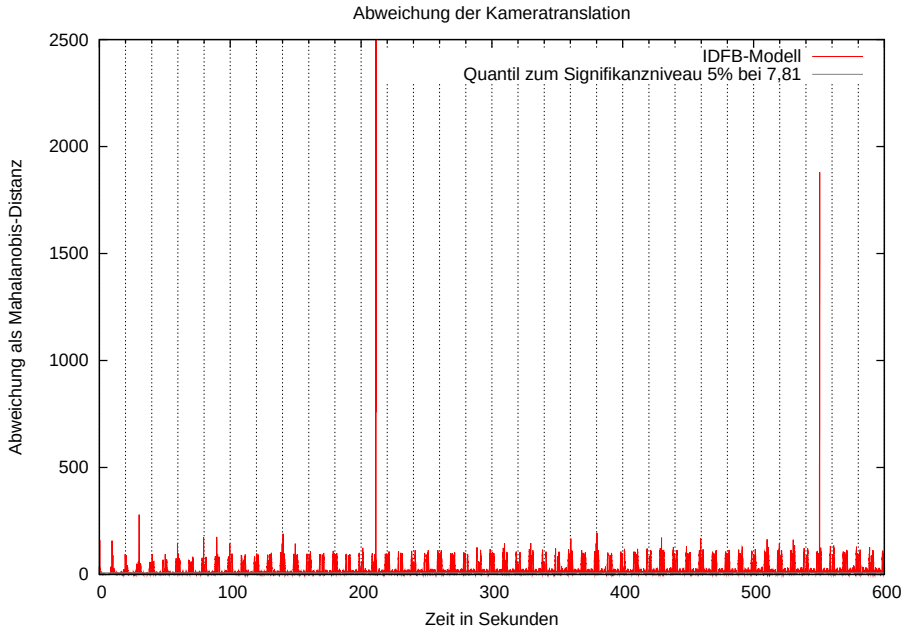


(a) nichtentkoppeltes IDFB-Modell

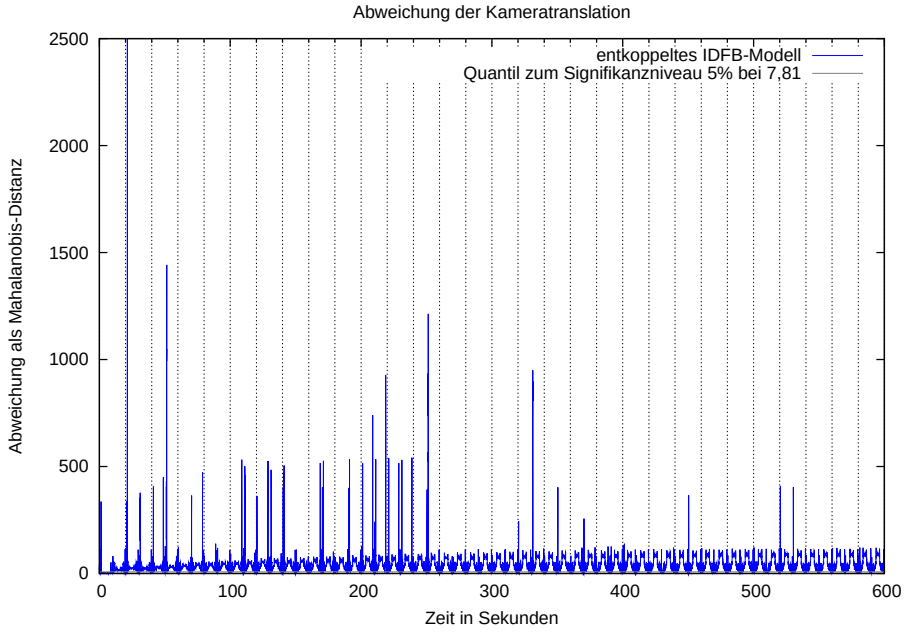


(b) entkoppeltes IDFB-Modell

Abbildung 5.14: Wandsequenz ohne Subpixelgenauigkeit mit voller Suche
Zwischen den Modellen gibt es große Unterschiede in der Unsicherheit. Auffällig ist der starke Anstieg der Mahalanobis-Distanz bei dem nichtentkoppelten Modell. Beide Modelle sind sich zu sicher über ihre Position.

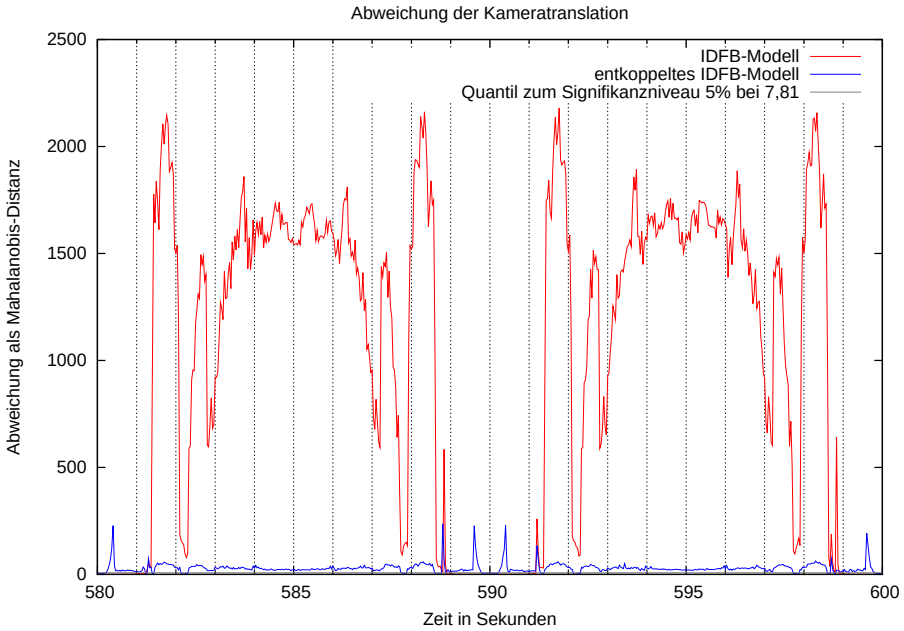


(a) nichtentkoppeltes IDFB-Modell

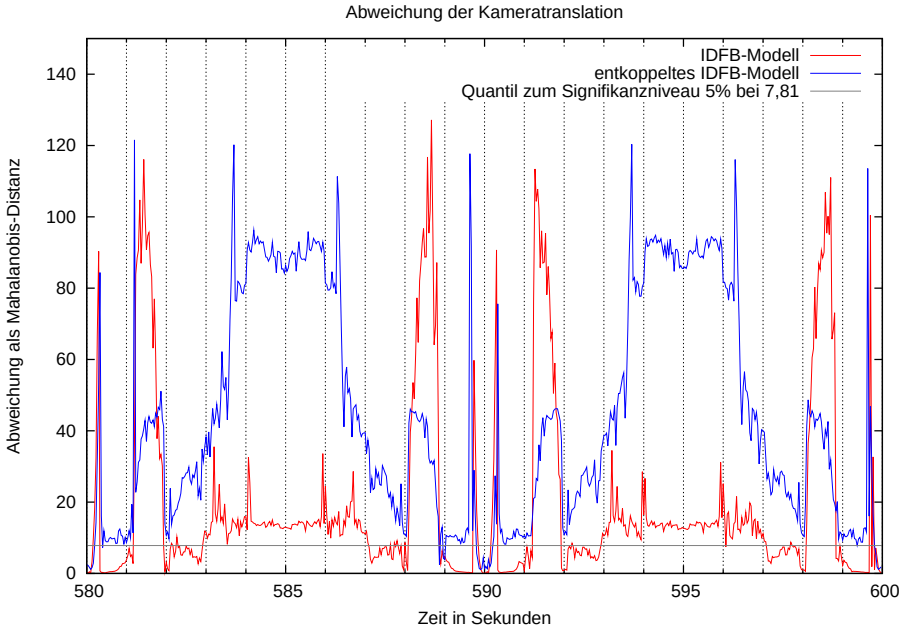


(b) entkoppeltes IDFB-Modell

Abbildung 5.15: Wandsequenz mit Subpixelgenauigkeit
Wie bei ausgeschalteter Subpixelgenauigkeit haben beide Modelle wiederum hohe Mahalanobis-Distanzen. Das nichtentkoppelte Modell zeigt keinen Anstieg mehr über den Zeitverlauf.

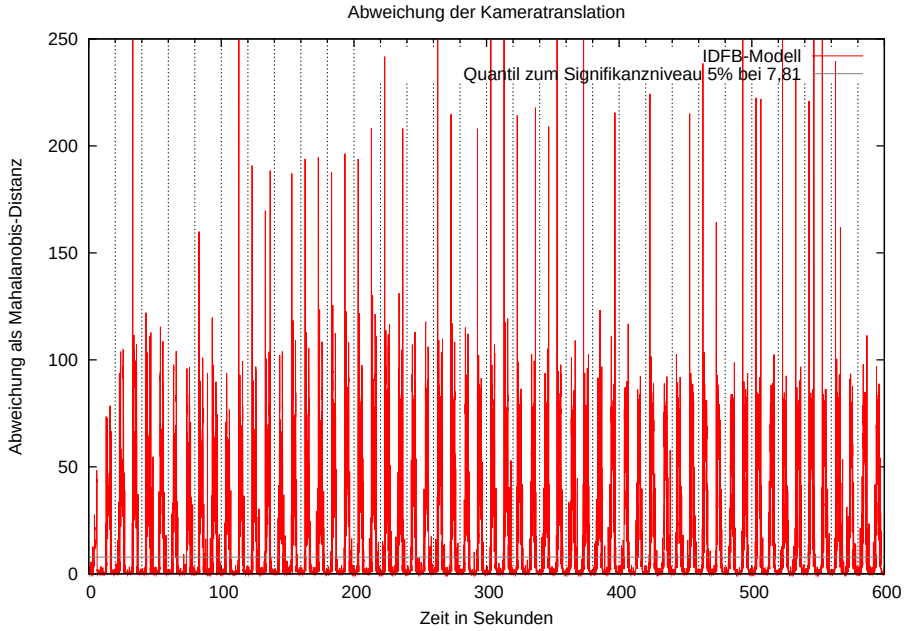


(a) ohne Subpixelgenauigkeit

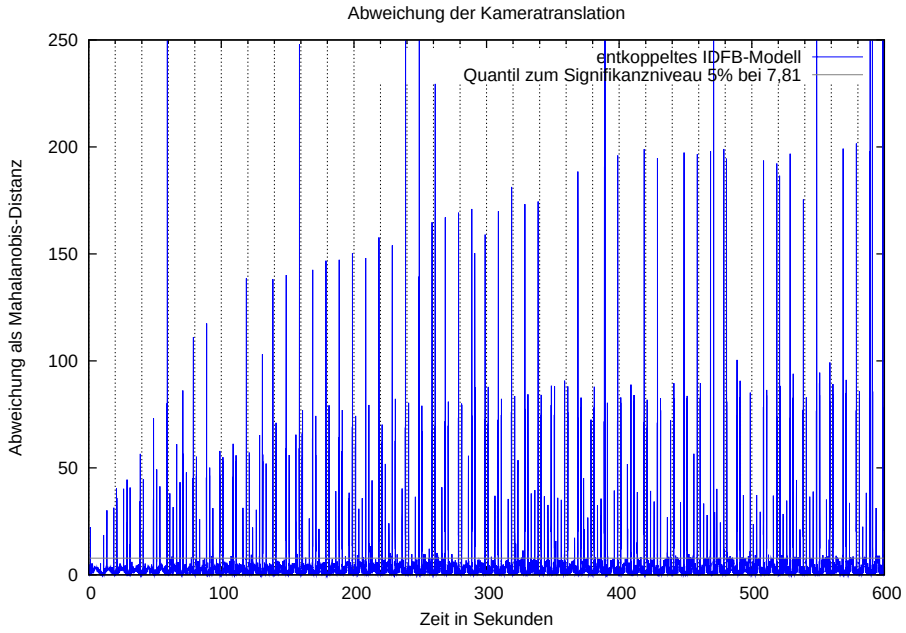


(b) mit Subpixelgenauigkeit

Abbildung 5.16: Wandsequenz mit voller Suche, vergrößerte Ansicht
In der vergrößerten Ansicht ist gut zu erkennen, dass das Quantil sehr oft überschritten wird.



(a) nichtentkoppeltes IDFB-Modell



(b) entkoppeltes IDFB-Modell

Abbildung 5.17: Tunnelsequenz mit Subpixelgenauigkeit ohne volle Suche
Bei beiden Modellen sind höhere Ausschläge erkennbar. Diese sind aber beim entkoppeltem Modell wesentlich geringer in ihrer Anzahl.

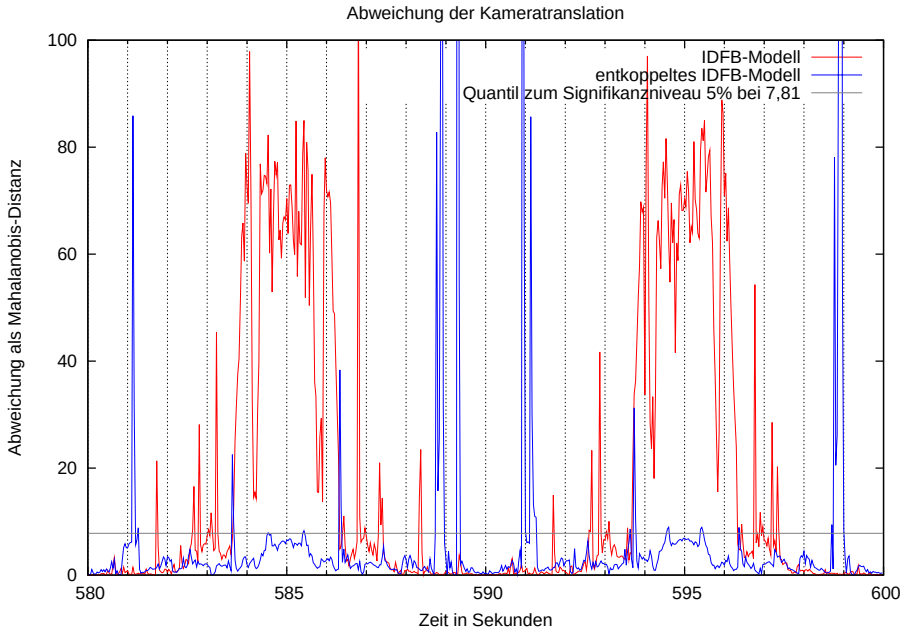


Abbildung 5.18: Tunnelsequenz mit Subpixelgenauigkeit ohne volle Suche, vergrößerte Ansicht
Das entkoppelte Modell bleibt fast immer unter dem Quantil. Wenn das Quantil überschritten wird, das sind die Werte oft sehr groß.

Die einfache Würfelsequenz lieferte wiederum gute Schätzungen. So lag nach einer Einschwungphase die Abweichung, gemessen als Euklidische Distanz, immer unter 0,15 cm. Die Stabilität der Sequenz ist auch daran gut erkennbar, dass die Anzahl der Features sich schon nach dem ersten Durchlauf nicht mehr verändert. Sie beträgt je nach Modell 33 bzw. 35 Features. Bei ausgeschalteter Subpixelgenauigkeit war das entkoppelte Modell wieder etwas besser. Andersherum schätzt bei eingeschalteter Subpixelgenauigkeit das nichtentkoppelte IDBF-Modell über die gesamte Zeit die Features besser. Zum Ende ist aber eine Annährung beider Modelle zu beobachten. Auffällig ist die leichte Schwingung der Genauigkeit, diese erfolgt in einem 10 Sekunden Abstand, also genau der Länge einer Drehung um die Würfelspitze. Auch nach mehreren Umrundungen werden die Positionen der Features neu geschätzt, erkennbar an den Genauigkeitsänderungen.

In der Wandsequenz fallen die Abweichungen der Features mit knapp 0,35 cm kaum größer aus als in der einfachen Würfelsequenz. Auch bei der wesentlich höherer Anzahl von über 150 Features. Eine Ausnahme stellt die Messung mit entkoppeltem IDBF-Modell und ausgeschalteter Subpixelgenauigkeit dar. Mit 3,5 cm Abweichung ist dieser Wert 10-mal größer, bei eingeschalteter Subpixelgenauigkeit kann der Wert aber reduziert werden. Interessant ist dabei, dass sich die Kameraschätzung genau umgedreht verhalten hat. Dort war das entkoppelte Modell genauer und sicherer.

Die einzige Tunnelsequenz die durchlief zeigt wesentlich höhere Genauigkeitsabweichungen. So liegt die Genauigkeit bei 15 bis 20 cm. Trotz ähnlicher Größenordnungen in der Anzahl der Features von 150 wie in der Wandsequenz. Die Stabilität der Features über den gesamten Zeitverlauf ist gut.

5.4 Auswirkung fester Feature-Initialisierungen

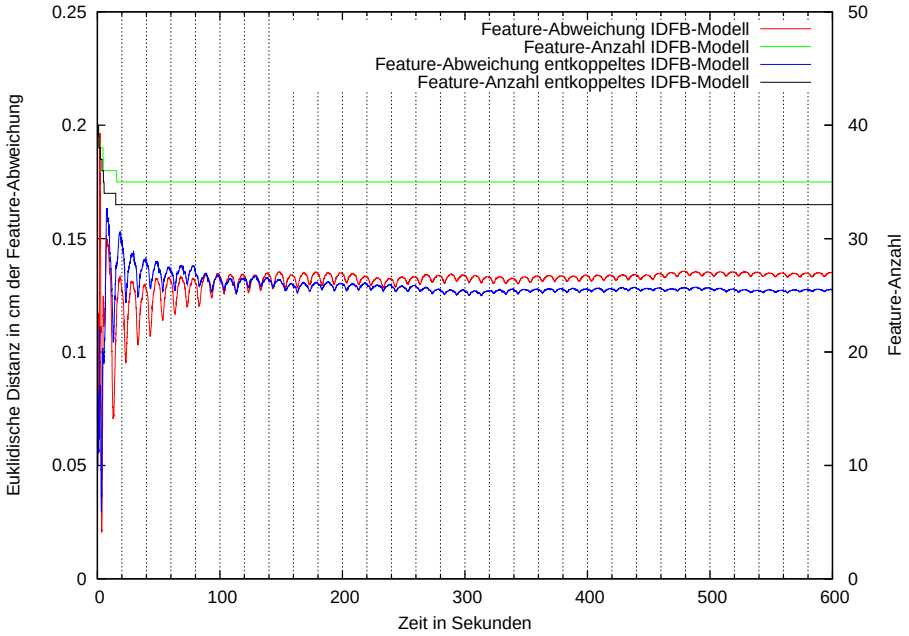
Um den Einfluss der Feature-Initialsierungen zu untersuchen, haben wir die beobachteten initialen Features einzelner ausgewählter Testdurchläufe gespeichert. Und diese dann dem anderen Modell fest vorgegeben, also die Feature-Initialsierung extern vorgenommen.

In der einfachen Würfelsequenz (Abbildung 5.22) war, wie zu erwarten, keine Änderung erkennbar. Beide Modelle haben ja bereits im Vorhinein gute Schätzungen geliefert.

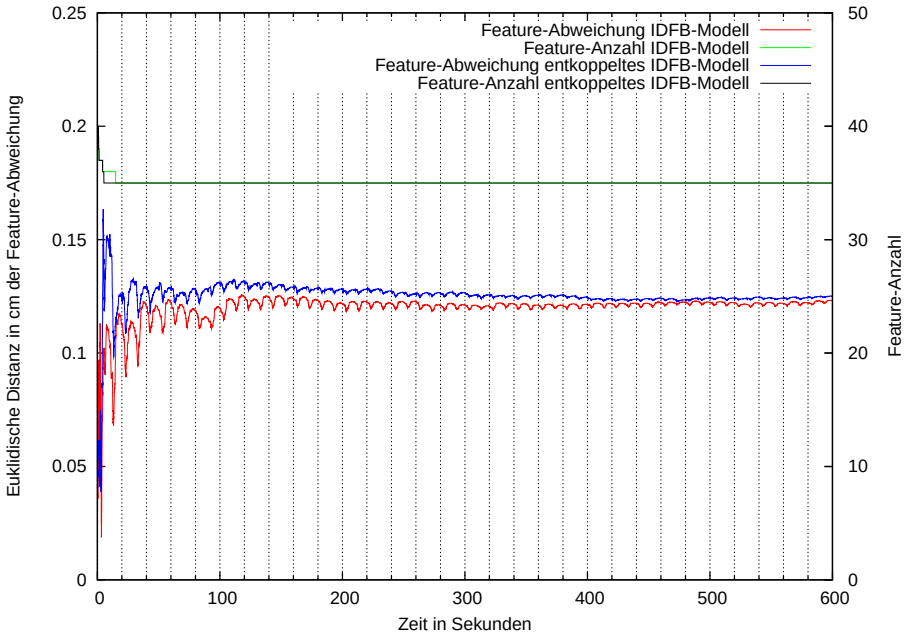
Interessanter ist der Vergleich in der Wandsequenz (Abbildung 5.23). Die Genauigkeit des nichtentkoppelten Modells kann durch die Feature des entkoppelten Modells ungefähr um den Faktor 3 verbessert werden. Es wird aber nicht die Genauigkeit des entkoppelten Modells ohne externe Initialisierung erreicht. Andersherum verschlechtert sich die Genauigkeit beim entkoppelten Modell durch die Feature des nichtentkoppelten Modells ebenfalls um den Faktor 3.

Eine analoges Verhalten zeigt sich bei der Mahalanobis-Distanz. Das starke Ansteigen dieser, wie noch in Abbildung 5.14(a), ist in Abbildung 5.25 nicht mehr zu beobachten.

Die Feature-Positionen haben also einen großen Einfluss auf das Ergebnis, sind aber nicht alleine entscheidend.

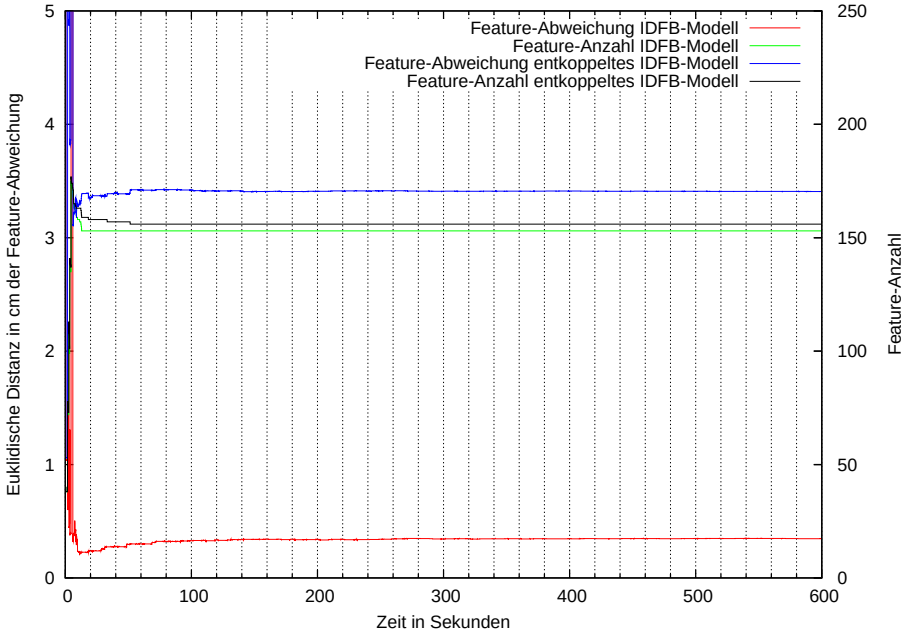


(a) ohne Subpixelgenauigkeit

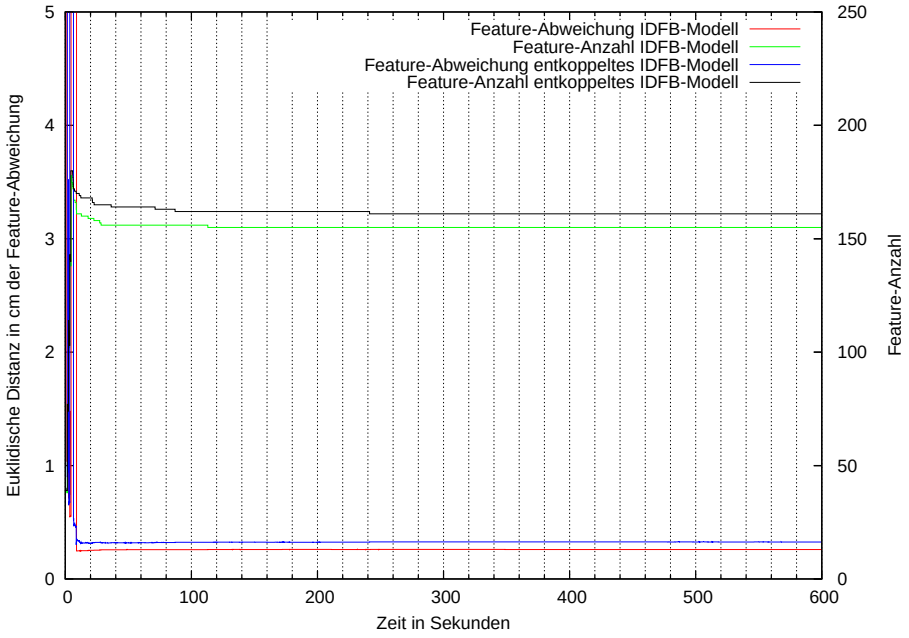


(b) mit Subpixelgenauigkeit

Abbildung 5.19: Einfache Würfelsequenz mit voller Suche
Die Feature-Anzahl ist nach dem ersten Durchlauf konstant. Die Genauigkeiten der Feature-Abweichungen unterscheiden sich in beiden Modellen kaum.



(a) ohne Subpixelgenauigkeit



(b) mit Subpixelgenauigkeit

Abbildung 5.20: Wandsequenz mit voller Suche
Die Feature-Anzahl pendelt sich nach einigen Durchläufen auf einen festen Wert ein. Ohne Subpixelgenauigkeit schätzt das entkoppelte Modell die Features wesentlich ungenauer als das nichtentkoppelte Modell. Bei aktivierter Subpixelgenauigkeit sind die Unterschiede marginal.

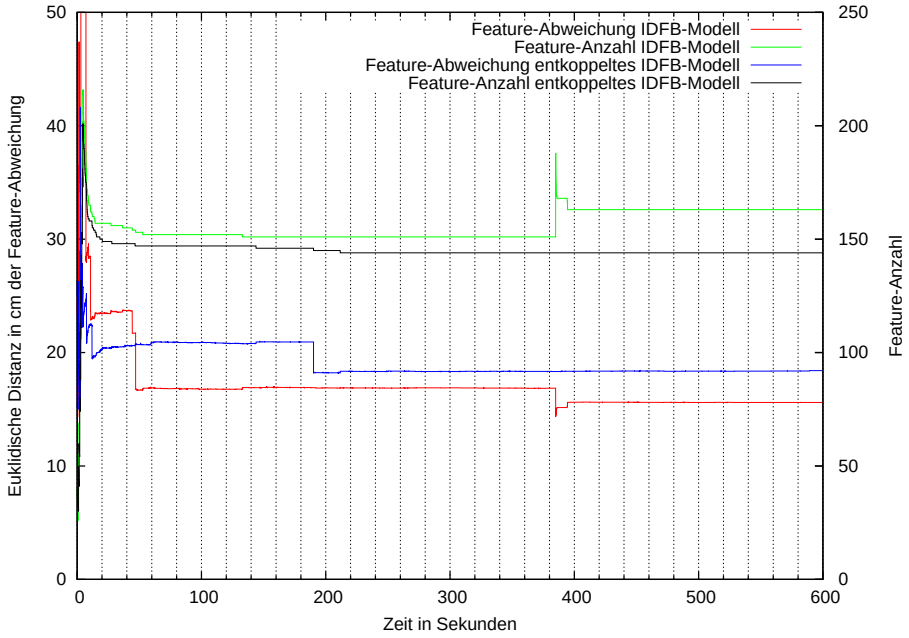


Abbildung 5.21: Tunnelsequenz mit Subpixelgenauigkeit ohne volle Suche
Die Feature-Anzahlen sind relativ stabil über den gesamten Verlauf. Die Feature-Abweichung ist mit über 10 cm aber vergleichsweise groß.

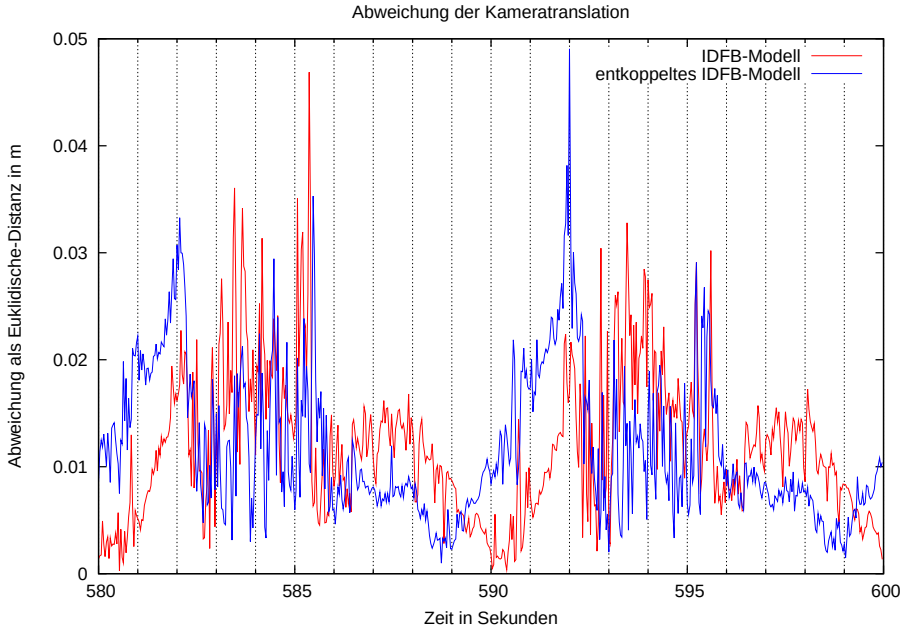
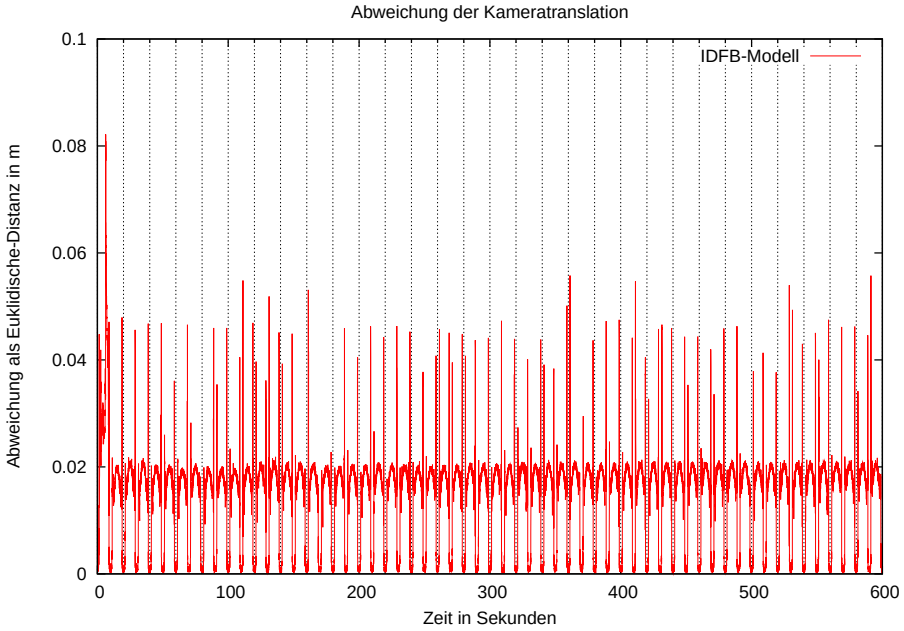
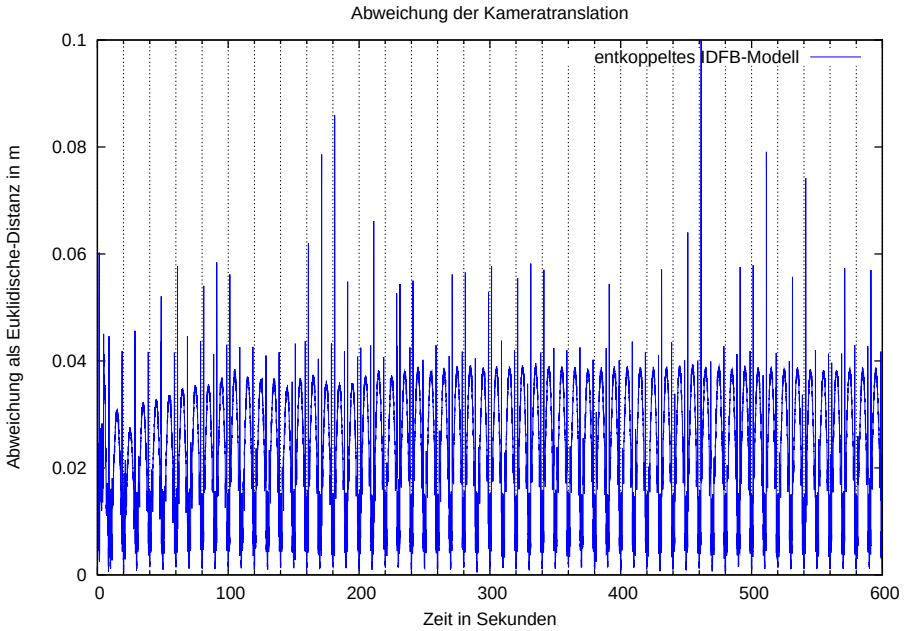


Abbildung 5.22: Einfache Würfelsequenz mit Subpixelgenauigkeit und voller Suche, Features initialisiert, vergrößerte Ansicht
Es sind keine großen Unterschiede zwischen den Modellen auszumachen.



(a) nichtentkoppeltes IDFB-Modell



(b) entkoppeltes IDFB-Modell

Abbildung 5.23: Wandsequenz ohne Subpixelgenauigkeit mit voller Suche, Features initialisiert
Die Maximalausschläge sind bei beiden Modellen ähnlich hoch. Allerdings treten diese beim entkoppelten initialisierten Modell häufiger auf.

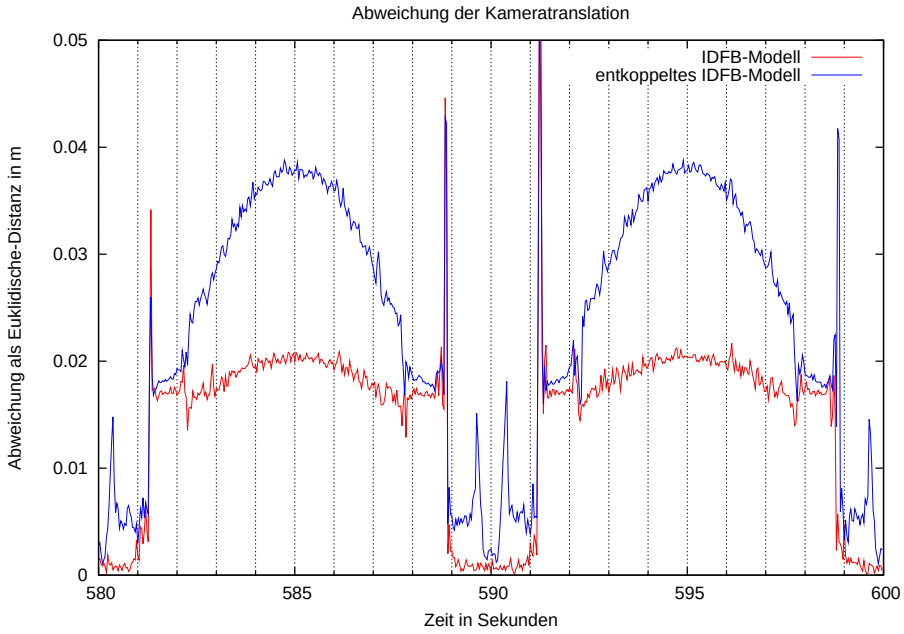


Abbildung 5.24: Wandsequenz ohne Subpixelgenauigkeit mit voller Suche, Features initialisiert, vergrößerte Ansicht
Durch die Initialisierung ist das nichtenkoppelte Modell genauer als das entkoppelte Modell. Die konkreten Feature-Positionen sind in dieser Testsequenz wichtiger für ein gutes Ergebnis, als die größere Freiheit durch die Entkopplung.

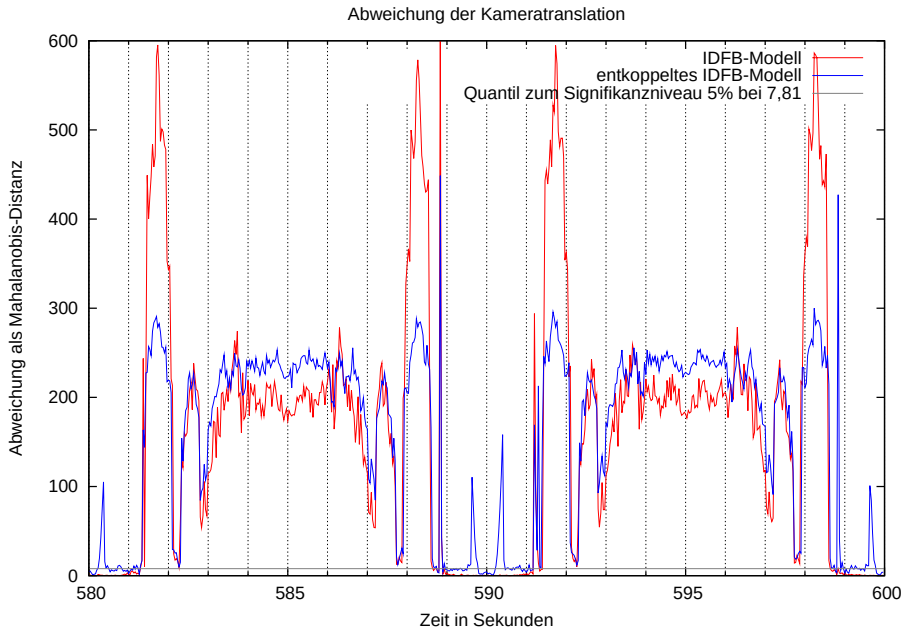


Abbildung 5.25: Wandsequenz ohne Subpixelgenauigkeit mit voller Suche, Features initialisiert, vergrößerte Ansicht
Die Mahalanobis-Distanzen weichen bei den initialisierten Testdurchläufen kaum voneinander ab.

Kapitel 6

Schlussfolgerungen

Die durchgeführten Experimente lassen insgesamt keine eindeutigen Schlussfolgerungen ziehen. Es ist nicht eindeutig, ob das entkoppelte IDFB-Modell generell besser geeignet ist als das nichtentkoppelte IDFB-Modell. In den einfachen Testsequenzen ist bei aktivierter Subpixelgenauigkeit das nichtentkoppelte IDFB-Modell gleichwertig (einfache Würfelsequenz) gegenüber dem entkoppelten IDFB-Modell oder sogar überlegen (Wandsequenz). Auch bei deaktivierter Subpixelgenauigkeit ist das entkoppelte IDFB-Modell nicht zwingend genauer (Tabelle 6.1). Auch wird in einem Fall zwar die Kameraposition und -rotation besser geschätzt, aber die Feature-Positionen sind dafür ungenauer. Die Würfelsequenz lässt kaum Aussagen zu, da fast alle Testdurchläufe fehlgeschlagen sind. Auch die Zeitdauern bis zur Instabilität weisen keine großen Unterscheide auf. Die Tunnelsequenz zeigt als einzige Testsequenz eine eindeutige Bevorzugung des nichtentkoppelten IDFB-Modells. Sie liefert bessere Resultate bei der Verfolgung der Kamera, da mehr Testdurchläufe erfolgreich beendet wurden bzw. die Testsequenzen länger stabil blieben. Auch war im Vergleich, des einzigen Testdurchlaufes der mit beiden Modellen funktioniert hat, die Abweichung von der Originaltrajektorie für das entkoppelte IDFB-Modell geringer. Allerdings gilt das nicht für die Feature-Genauigkeit. Dort war das nichtentkoppelte IDFB-Modell etwas besser. Beide Modelle hatten eine hohe Feature-Abweichung von über 15 cm im Vergleich zu den anderen Sequenzen. Dies kann aber auch an dem Aufbau der Tunnelsequenz liegen. Da hier Features im Sichtbereich wesentlich weiter entfernt sein können im Gegensatz zu den anderen Testsequenzen.

Dieser Eindruck vom entkoppelten IDFB-Modell wird durch die Einbeziehung der Unsicherheit in Form der Mahalanobis-Distanz bestätigt. Wiederum sind in den einfachen Testsequenzen bei aktivierter Subpixelgenauigkeit keine wesentlichen Unterschiede erkennbar. Während die Tunnelsequenz erneut leicht das entkoppelte IDFB-Modell bevorzugt. In keinem Testdurchlauf kam es zu einem Bestehen des Signifikanztestes (Tabelle 6.3). Dies kann zwei Gründe haben. Die Unsicherheiten wurden vom VSLAM-Programm zu optimistisch geschätzt oder die Genauigkeiten waren zu hoch.

Bei den fehlgeschlagenen Testdurchläufe kam es immer innerhalb kürzester Zeit zu einem Versagen. Es gab also keinen leichten Anstieg der Abweichung über mehrere Schleifen, sondern das VSLAM-Programm konnte an einem bestimmten Zeitpunkt die Trajektorienverfolgung nicht mehr erfolgreich durchführen. Dies kann zum Beispiel an falsch erkannten Features liegen. In den Sequenzen gab es

Sequenz	A	B	C	D
Würfelsequenz nichtentkoppelt	x	x	3.56	x
Würfelsequenz entkoppelt	x	x	x	x
einfache Würfelsequenz nichtentkoppelt	-	1.25	-	1.14
einfache Würfelsequenz entkoppelt	-	1.40	-	1.11
Wandsequenz nichtentkoppelt	-	4.32	-	0.48
Wandsequenz entkoppelt	-	1.02	-	1.38
Tunnelsequenz nichtentkoppelt	x	x	1.15	x
Tunnelsequenz entkoppelt	x	x	0.84	2.07

Tabelle 6.1: Gemittelte Euklidische Distanz über alle Messzeitpunkte in cm. Anzahl der Werte in Prozent, die unter dem Quantil zum Signifikanzniveau 5% lagen. Für ein Bestehen des Signifikanztestes sind 95% erforderlich. Im Mittel waren die Schätzungen in der einfachen Würfelsequenz am besten und in der Wandsequenz am schlechtesten. Legende in Tabelle 6.2.

Symbol	Erläuterung
A	ohne Subpixelgenauigkeit ohne volle Suche
B	ohne Subpixelgenauigkeit mit voller Suche
C	mit Subpixelgenauigkeit ohne volle Suche
D	mit Subpixelgenauigkeit mit voller Suche
-	Testdurchlauf nicht durchgeführt
x	Testdurchlauf fehlgeschlagen

Tabelle 6.2: Legende für die Testdurchläufe

Sequenz	A	B	C	D
Würfelsequenz nichtentkoppelt	x	x	40,26	x
Würfelsequenz entkoppelt	x	x	x	x
einfache Würfelsequenz nichtentkoppelt	-	81,51	-	90,32
einfache Würfelsequenz entkoppelt	-	83,58	-	90,66
Wandsequenz nichtentkoppelt	-	24,46	-	48,76
Wandsequenz entkoppelt	-	14,16	-	6,97
Tunnelsequenz nichtentkoppelt	x	x	69,41	x
Tunnelsequenz entkoppelt	x	x	94,78	53,48

Tabelle 6.3: Anzahl der Werte in Prozent, die unter dem Quantil zum Signifikanzniveau 5% lagen. Für ein Bestehen des Signifikanztestes sind 95% erforderlich. Im Mittel waren die Schätzungen in der einfachen Würfelsequenz am besten und in der Wandsequenz am schlechtesten. Legende in Tabelle 6.2.

immer eine kritische Stelle an der dies auftrat. Dies war in der Würfelsequenz der Schleifenschluss und in der Tunnelsequenz in der Mitte des Tunnels, also bei maximaler Geschwindigkeit. Die Schätzung der Features von ihren Originalpositionen war in diesen beiden Sequenzen tendentiell eher höher (oft über 10 cm), dies könnte fehlerhafte Zuordnungen von Features begünstigen. Das entkoppelte IDFB-Modell zeigt trotz der erhöhten Flexibilität im Umgang mit der Rotation nur teilweise bessere Ergebnisse. Auch ist nicht ganz klar ob dies auf reine Zufälligkeit oder wirklich an einer Überlegenheit des Modelles liegt.

In dem Paper von Chiuso et al. [CFM+02] wird die Reduktion des Feature-Vektors auf einen Wert als subminimales Modell beschrieben. Durch die Reduktion hat das normalverteilte Modellrauschen nicht mehr den Erwartungswert 0. Dies ist eine Verletzung der Annahmen des Erweiterten Kalman-Filters (siehe Abschnitt 2.2.2). Chiuso et al. haben daher postuliert und durch Experimente gezeigt, dass bei längerer Laufzeit der Sequenzen das Modell instabil wird. Diese Verhalten konnten wir nicht eindeutig reproduzieren. Nur in der Tunnelsequenz schien das nichtminimale (entkoppelte IDFB-Modell) besser geeignet zu sein. Ein Kritikpunkt war die Länge der Sequenzen, diese waren aber mit 600 Sekunde über 50-mal länger als in dem Paper von Chiuso et al.. Bei Betrachtung der Testergebnisse 5.1 und 5.9 fällt auf, dass es Testfälle gibt in denen das subminimale Modell (nichtentkoppelte IDFB-Modell) innerhalb kurzer Zeit instabil wird. Das nichtminimale (entkoppelte IDFB-Modell) aber nur kurze Zeit später, so dass bei ungünstiger Auswahl der Testfälle und Zeitdauern die Ergebnisse von Chiuso et al. zustande kommen können.

Sequenz	ohne Sub	mit Sub
einfache Würfelsequenz nichtentkoppelt	1.25	1.14
einfache Würfelsequenz entkoppelt	1.40	1.11
Wandsequenz nichtentkoppelt	4.32	0.48
Wandsequenz entkoppelt	1.02	1.38

Tabelle 6.4: Durchschnittliche Abweichung von der idealen Kameraposition in cm angegeben in der euklidischen Distanz
Parametereinstellungen: volle Suche war aktiviert, minimale Fläche auf 200 Pixel gestellt, Sub gibt die Subpixelgenauigkeit an

6.1 Parametereinfluss auf die Modelle

Wir möchten in diesem Abschnitt einen Überblick über die Auswirkungen unterschiedlicher Parametereinstellungen geben.

6.1.1 Subpixelgenauigkeit

Auffällig war der Einfluss der Subpixelgenauigkeit auf die Testsequenzen. Wie in den Testergebnissen beschrieben, konnte durch das Einschalten der Subpixelgenauigkeit oft eine höhere Genauigkeit erreicht werden. In der Wandsequenz (Abbildung 5.5 und 5.6) ist dies besonders deutlich sichtbar. Dies deutet daraufhin, dass die Subpixelgenauigkeit die Unsicherheiten zwischen den initialen Pixelpositionen korrigieren kann. Zwar war in der einfachen Würfelsequenz (Abbildung 5.3) die Auswirkung geringer. Aber in dieser Sequenz sind in keiner Parametereinstellung gravierende Unterschiede in der Genauigkeit zwischen beiden Modellen beobachtbar. In den Tabellen 6.4 und 6.5 sind die Ergebnisse noch einmal zusammengefasst. In den komplexeren Testsequenzen waren nur Testläufe mit eingeschalteter Subpixelgenauigkeit erfolgreich. Selbst bei nicht erfolgreichen Durchläufen, dauerte es merkbar länger bis sie instabil wurden, als in den Testläufen mit ausgeschalteter Subpixelgenauigkeit.

Die Aktivierung der Subpixelgenauigkeit führt in der Mehrzahl der Testdurchläufe zu einer besseren Genauigkeit und sollte als Standardoption gesetzt werden.

6.1.2 Volle Suche

Die volle Suche sollte eigentlich keine großen Auswirkungen auf den Ablauf des VSLAM-Programmes haben, da es eine Maßnahme zur Geschwindigkeitsoptimierung ist. Insbesondere müsste die volle Suche bessere Ergebnisse liefern als die eingeschränkte Suche, da ein größeres Gebiet abgesucht wird. Aber in den komplizierteren Sequenzen 5.1 und 5.9 erwies sich eine eingeschränkte Suche sogar als vorteilhafter. Mit dieser Parametereinstellung gab es mehr erfolgreiche Durchläufe. Wir vermuten, dass dies an Zufälligkeiten in der Feature-Beobachtung lag. Denn diese Sequenzen sind nicht stabil durchgelaufen und

Sequenz	ohne Sub	mit Sub
einfache Würfelsequenz nichtentkoppelt	4.80	3.29
einfache Würfelsequenz entkoppelt	5.37	3.16
Wandsequenz nichtentkoppelt	622.65	20.53
Wandsequenz entkoppelt	23.05	40.83

Tabelle 6.5: Durchschnittliche Mahalanobis-Distanz von der idealen Kameraposition in m
Parametereinstellungen: volle Suche war aktiviert, minimale Fläche auf 200 Pixel gestellt, Sub gibt die Subpixelgenauigkeit an

zeigen signifkante Änderungen im Testdurchlauf schon bei leichten Änderungen der Testumgebung. Eine andere Erklärung ist, dass der Eckendetektor fehlerhafte Patches aussselektiert. Durch die Beschränkung des Patch-Vergleichs auf durch den Eckendetektor gefundenen markanten Punkten werden seltener inkorrekte Patches wiedererkannt.

6.1.3 Minimale Fläche eines Features

Die minimale Fläche eines Features erwies sich als nützlich um ein Wiedererkennen von Features zu ermöglichen. Durch die Kamerabewegung wird die Fläche des Patches kleiner und damit die erneute Beobachtung zu unzuverlässig. Als Folge davon wurden die Features nicht mehr erkannt und dementsprechend gelöscht. Bei der nächsten Schleife in der Sequenz war deswegen das Feature nicht mehr vorhanden und durch die fehlende erneute Beobachtung von Features konnte das VSLAM-Programm einmal gemachte Fehler nicht korrigieren. Als Ergebnis wiesen die Trajektorien sehr starke Abweichungen auf. Durch Anheben des Wertes für die minimale Fläche konnte dieses Problem behoben werden. Ab einem sequenzabhängigen Schwellwert tritt keine weitere Verbesserung ein.

Kapitel 7

Zusammenfassung

Diese Arbeit beinhaltete die Ausarbeitung des entkoppelten IDFB-Modells (Abschnitt 3.3) und dessen Implementierung und Integration in ein bestehendes VSLAM-Programm. Wir haben dann das Modell evaluiert, indem wir es gegen das nichtentkopplte IDFB-Modell verglichen haben. Dazu haben wir das Test-Framework von Pietzsch und Funke [PF09] genutzt und erweitert. Sowie die entsprechenden Tests konzipiert. Ziel war es nachzuprüfen, ob durch die Reduktionen des nichtentkoppelten Modells signifikante Fehlschätzungen entstehen.

Zusammendfassend kann man folgende Schlüsse ziehen. Die Verwendung von Subpixelgenauigkeit reduziert bis auf wenige Ausnahmen immer die Ungenauigkeit und sollte standardmäßig eingeschaltet sein. In der Wand-Testsequenz ist damit das nichtenkoppelte IDFB-Modell auch gleichwertig oder überlegen gegenüber dem entkoppelten IDFB-Modell. Die einfache Würfelsequenz liefert in jeder Einstellung gute Ergebnisse. Komplexere Testsequenzen zeigen ein inhomogenes Verhalten. Sie bergen viele Möglichkeiten an denen das VSLAM-Programm scheitern kann. Ein eindeutiger Zusammenhang zwischen Wirkung und Ursache ist deswegen nicht herzustellen. Für die Tunnelsequenz ist aber das entkoppelte IDFB-Modell vorzuziehen, da damit die Testläufe stabiler sind. Eventuell sind auch Verbesserungen die nicht das Feature-Modell betreffen zielführender um die Gesamtstabilität zu erhöhen, wie die Parametereinstellung minimale Fläche eines Features zeigt. Das nichtentkoppelte IDFB-Modell zeigt damit in einer von vier Testsequenzen Vorteile, die restlichen Testsequenzen zeigen für beide Modelle ähnliche Ergebnisse. Dieses Resultat lässt unserer Ansicht nach keine klare Aussage zu. Es ist ein Anhaltspunkt dafür, dass in bestimmten Situationen eine Entkopplung ratsam sein könnte.

Für zukünftige Untersuchungen könnten daher folgende Bereiche näher beleuchtet werden. Die Abänderung oder das Hinzufügen von Testsequenzen ist zu bearbeitendes Aufgabengebiet. Es liefert vermutlich neue Erkenntnisse über Situationen in denen das entkoppelte IDFB-Modell überlegen ist. Einfachere Sequenzen helfen eventuell bestimmte Teilprobleme mit einem Modell zu isolieren, da dadurch zufällige Faktoren wie die Positionen der Features besser kontrolliert werden können. In diesen Bereich fällt auch die Wahl der Texturen. Das VSLAM-Programm initialisiert die Features je nach Textur an andere Positionen. Der Einfluss der Texturen ist also bedeutsam. Weiterhin sind aktuell die Testsequenzen sehr periodisch aufgebaut, diese äußert sich in zyklisch wiederkehrenden Mustern bei den Testergebnissen. Eine Dynamisierung in Form von neuen

Szenen oder eine Trajektorie, die sich über die gesamte Testzeit verändert, würde zu weniger Wiederholungen über den gesamten Testlauf hinweg führen. Vielleicht erlaubt dies mögliche eingeführte Abweichungen durch das nichtenkoppelte IDFB-Modell sichtbar werden zu lassen. Da das entkoppelte Modell besonders bei hohem Rauschen seine Vorzüge ausspielen sollte, wäre eine Untersuchung des Rauschverhaltens interessant. Dies haben wir in der vorliegenden Arbeit aus Zeitgründen nicht weiter überprüft. Es erscheint uns aber wichtig, da das Rauschen eine offensichtliche Modellverletzung darstellt. Dabei könnte man einen Vergleich von Sequenzen mit unterschiedlich starkem Rauschen durchführen. Je höher das Rauschen ist, umso schlechter müssten die Schätzungen der Modelle werden. Wobei die Erwartung ist, dass das entkoppelte IDFB-Modell länger bessere Schätzungen liefert. Bisher haben wir uns darauf beschränkt die Features nur auf ihre Positionierung zu vergleichen. Jedes Feature besitzt aber auch eine Unsicherheit. Eine Schätzung die nahe der Originalposition liegt aber mit einer zu hohen Sicherheit behaftet ist, kann als schlechter eingestuft werden als eine ungenauere Schätzung bei der aber die wahre Position innerhalb der Unsicherheitsregion liegt. Eine Berücksichtigung dieser Information kann weitere Erkenntnisse über die Glaubwürdigkeit eines Modelles liefern.

Anhang A

Quaternionen

Wir nutzen Quaternionen zur Angabe der Rotationen. Für unsere Berechnungen sind dabei einige Rechenoperationen notwendig.

Die Quaternionen-Multiplikation ist folgendermaßen definiert:

$$\mathbf{q}_1 \circ \mathbf{q}_2 = \begin{pmatrix} w_1 \\ x_1 \\ y_1 \\ z_1 \end{pmatrix} \circ \begin{pmatrix} w_2 \\ x_2 \\ y_2 \\ z_2 \end{pmatrix} = \begin{pmatrix} w_1 w_2 - x_1 x_2 - y_1 y_2 - z_1 z_2 \\ w_1 x_2 + w_2 x_1 + y_1 z_2 - z_1 y_2 \\ w_1 y_2 + w_2 y_1 + z_1 x_2 - x_1 z_2 \\ w_1 z_2 + w_2 z_1 + x_1 y_2 - y_1 x_2 \end{pmatrix} \tag{A.1}$$

Die Jacobi-Matrixen des Produktes der Quaternionen-Multiplikation abgeleitet nach $\mathbf{q}_1$ bzw. $\mathbf{q}_1$ sind

$$\frac{\partial\, \mathbf{q}_1 \circ \mathbf{q}_2}{\partial \mathbf{q}_1} = \begin{bmatrix} w_2 & -x_2 & -y_2 & -z_2 \\ x_2 & w_2 & z_2 & -y_2 \\ y_2 & -z_2 & w_2 & x_2 \\ z_2 & y_2 & -x_2 & w_2 \end{bmatrix} \tag{A.2}$$

und

$$\frac{\partial\, \mathbf{q}_1 \circ \mathbf{q}_2}{\partial \mathbf{q}_2} = \begin{bmatrix} w_1 & -x_1 & -y_1 & -z_1 \\ x_1 & w_1 & -z_1 & y_1 \\ y_1 & z_1 & w_1 & -x_1 \\ z_1 & -y_1 & x_1 & w_1 \end{bmatrix}. \tag{A.3}$$

Um die Ableitung von $\frac{\partial \log(\mathbf{q})}{\partial \mathbf{q}}$ zu bestimmen, nutzen wir die *exponential map* von Grassia [Gra98] und die Umkehrregel. Die Rotation $\omega = (\omega_x, \omega_y, \omega_z)^\top$ kann damit wie folgt als Quaternion dargestellt werden

$$\mathbf{q} = \begin{pmatrix} q_w \\ q_x \\ q_y \\ q_z \end{pmatrix} = \begin{pmatrix} \cos(\frac{1}{2}\theta) \\ \frac{\sin(\frac{1}{2}\theta)}{\theta}\omega \end{pmatrix} \tag{A.4}$$

mit $\theta = |\omega|$. Die partiellen Ableitungen ergeben sich zu

$$\frac{\partial q_w}{\partial \omega_i} = -\frac{1}{2}\omega_i \frac{\sin(\frac{1}{2}\theta)}{\theta} \tag{A.5}$$

$$\frac{\partial q_i}{\partial \omega_i} = \frac{1}{2}\omega_i^2 \frac{\cos(\frac{1}{2}\theta)}{\theta^2} - \omega_i^2 \frac{\sin(\frac{1}{2}\theta)}{\theta^3} + \frac{\sin(\frac{1}{2}\theta)}{\theta} \tag{A.6}$$

$$\frac{\partial q_j}{\partial \omega_i} = \frac{1}{2}\omega_j\omega_i \frac{\cos(\frac{1}{2}\theta)}{\theta^2} - \omega_j\omega_i \frac{\sin(\frac{1}{2}\theta)}{\theta^3} \tag{A.7}$$

mit $i, j \in \{x, y, z\}$ und $i \neq j$. In der Nachbarschaft von $\theta \to 0$ kann man diese mittels Taylor-Näherung vereinfachen.

$$\frac{\partial q_w}{\partial \omega_i} = -\frac{1}{2}\omega_i \left(\frac{1}{2} - \frac{\theta^2}{48}\right) \tag{A.8}$$

$$\frac{\partial q_i}{\partial \omega_i} = \frac{\omega_i^2}{24}\left(\frac{\theta^2}{40} - 1\right) + \left(\frac{1}{2} - \frac{\theta^2}{48}\right) \tag{A.9}$$

$$\frac{\partial q_j}{\partial \omega_i} = \frac{\omega_j\omega_i}{24}\left(\frac{\theta^2}{40} - 1\right) \tag{A.10}$$

Für die Einheitsrotation $\mathbf{q} = (1, 0, 0, 0)^T$ erhält man damit $\frac{\partial q_i}{\partial \omega_i} = \frac{1}{2}$. Die anderen Werte sind 0. Mithilfe der Umkehrregel $\frac{\partial q_i}{\partial \omega_i}\frac{\partial \omega_i}{\partial q_i} = 1$ erhält man damit die gewünschte Ableitung.

$$\frac{\partial \log(\mathbf{q})}{\partial \mathbf{q}} = \begin{pmatrix} 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{pmatrix} \tag{A.11}$$

Literaturverzeichnis

[AP94] A. Azarbayejani and A. Pentland. Recursive estimation of motion, structure, and focal length. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17:562–575, 1994.

[CFM+02] Ro Chiuso, Paolo Favaro, Student Member, Hailin Jin, and Stefano Soatto. Structure from motion causally integrated over time. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:523–535, 2002.

[DRMS07] Andrew J. Davison, Ian D. Reid, Nicholas D. Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 29, 2007.

[Gra98] Sebastin F. Grassia. Practical parameterization of rotations using the exponential map. *J. Graph. Tools*, 3(3):29–48, March 1998.

[Low99] David Lowe. Object recognition from local scale-invariant features, 1999.

[MCD06] J. M. M. Montiel, Javier Civera, and Andrew J. Davison. Unified inverse depth parametrization for monocular slam. In *Robotics: Science and Systems*, 2006.

[PF09] Tobias Pietzsch and Jan Funke. A framework for evaluating visual slam, 2009.

[Pie08] T. Pietzsch. Efficient feature parameterisation for visual slam using inverse depth bundles. In *BMVC08*, 2008.

[Pie09] Tobias Pietzsch, 2009.

[RD05] Edward Rosten and Tom Drummond. Fusing points and lines for high performance tracking. In *IEEE International Conference on Computer Vision*, volume 2, pages 1508–1511, October 2005.

[RD06] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *European Conference on Computer Vision*, volume 1, pages 430–443, May 2006.

[ST94] Jianbo Shi and C. Tomasi. Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on*, pages 593–600, 1994.

[TBF05] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. The MIT Press, 2005.