INSTITUT FÜR INFORMATIK
AUTONOME INTELLIGENTE SYSTEME

RHEINISCHE FRIEDRICH-WILHELMS UNIVERSITÄT BONN

Bachelorarbeit

RGB-D-Bild-Registrierung durch dichtes Tiefen- und spärliches Bildpunkt-Matching

Arno Gutt

Erstgutachter:	Prof. Dr. Sven Behnke
Zweitgutachter:	Dr. Dirk Schulz
Betreuer:	Dipl.-Inf. Jörg Stückler

12. August 2013

Eidesstattliche Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus anderen Quellen übernommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Arno Gutt, Bonn, 12. August 2013

Danksagung

Danken möchte ich in erster Linie meinem Betreuer Jörg Stückler für den Ansporn, die Unterstützung und die Zeit, die er in dieses Projekt investiert hat.
Ebenfalls bedanken möchte ich mich bei den Mitarbeitern der Abteilung sowie bei Prof. Dr. Sven Behnke für die Möglichkeit meine Bachelorarbeit in diesem Bereich schreiben zu dürfen.

Kurzfassung

Herkömmliche Videosequenzen bestehen lediglich aus 3 Farbkanälen für rot, grün und blau. RGB-D-Sequenzen hingegen, wie sie die Microsoft Kinect liefert, enthalten zusätzlich einen Kanal, der zu jedem Pixel einen Tiefenmessung enthält. Projiziert man diese Pixel in den 3-dimensionalen Raum lässt sich daraus eine Surfelmap erstellen. Im RGB-Bild lassen sich charakteristische Punkte, sogenannte Features oder Bildmerkmale, detektieren. Diese werden in der Regel von charakteristischen Punkten oder „Landmarken“ im 3-dimensionalen Raum erzeugt.

In dieser Bachelorarbeit wird ein Verfahren implementiert und ausgewertet, welches RGB-D-Bilder paarweise über Bildmerkmalen und Surfel aufeinander registriert und so die Bewegung zwischen den Positionen, von denen die Bilder aufgenommen wurden, schätzt und schrittweise verbessert. Dafür werden die Informationen aus Surfel- und Bildpunktmatching miteinander kombiniert, um in Situationen, in denen eine der beiden Techniken schlechte Ergebnisse liefert, immernoch ein robustes Ergebnis zu erzielen. Zudem werden die geschätzten Landmarkenpositionen mehrfach nachjustiert, um Messfehler zu minimieren und so die Transformation zu verbessern.

INHALTSVERZEICHNIS

1. EINLEITUNG

Das Aufgabenspektrum von Robotern hat mittlerweile viele Anwendungsbereiche erreicht. Schwierigkeiten bei der Aufgabenbewältigung ergeben sich vor allem aus variablen Umgebungen. Dieses Problem stellt sich unter anderem Service- und Haushaltsrobotern, die Gegenstände benutzen oder transportieren und dabei Hindernissen beim Greifen oder Navigieren ausweichen müssen. Hierfür ist es wichtig ein möglichst genaues Modell der Umgebung des Roboters aufzubauen und die Position des Roboters darin exakt zu bestimmen. Dafür ist es nicht unbedingt nötig relativ teure Sensoren, wie z.B. Laserscanner, zu verwenden. Unter anderem Verfahren von Richard A. Newcombe et al. [6] und Albert S. Huang et al. [5] haben gezeigt, dass sich mit günstigen Alternativen, in Form von RGB-D-Kameras, gute Ergebnisse erzielen lassen.

Bei RGB-D-Aufnahmen, wie mit der Microsoft Kinect, gibt es ein paar unangenehme Nebeneffekte. Ein Problem, welches hier aber nicht weiter behandelt wird, ist die leicht asynchrone Aufnahme von RGB- und Tiefenbild. Interessant ist jedoch das Problem der Tiefenmessungen, welche in manchen Bildbereichen sogar ungültig sein können. Mit zunehmender Tiefe werden die Messungen quadratisch ungenauer und die Tiefenwerte diskreter, sodass man die diese mit größerer Unsicherheit betrachten muss. Tiefensprünge an geraden Kanten (zum Beispiel an Tischkanten) sind oft unsauber.
Die Überlegung liegt nahe, dass diese Messungen durch ein Bündelausgleichsverfahren, auch Bundle Adjustment, nachjustiert werden müssen. Durch den Abgleich mit anderen Frames einer Aufnahmesequenz lassen sich neue Landmarkenkoordinaten und relative Kameraposen errechnen, die eher der Realität entsprechen.

Aktuelle Kamera-Tracking-Verfahren, welche mit Bildregistrierung arbeiten, rekonstruieren die Kameraposition und -rotation mit Abweichungen unter 10cm und wenigen Grad in Echtzeit, falls ausreichend Textur für Bildpunktmerkmale bzw. Form für dichte Tiefe vorhanden ist. Mit einem Bündelausgleichsverfahren, welches sowohl Farbbild- als auch Tiefenmes-

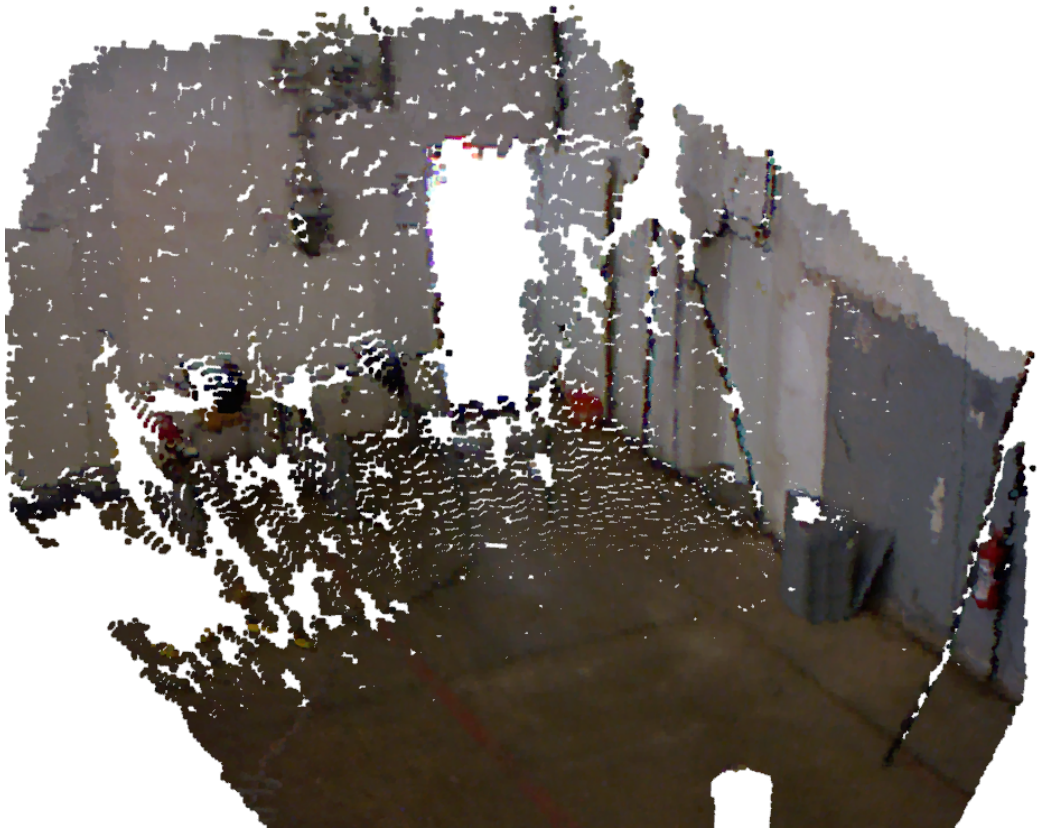

Fig. 1.1: Diskretisierungseffekt bei zunehmender Tiefe

sungen einbezieht, könnte es möglich sein, die Rekonstruktionsgenauigkeit und die Robustheit zu erhöhen, wenn entweder geringe Textur oder Form vorhanden sind.

2. GRUNDLAGEN

2.1 Multi-resolution surfel map

Die Implementierung dieses Ansatzes basiert auf der bereits existierenden Software der „Multi-resolution surfel maps“ [7] und erweitert deren Funktionalität um das Bundle Adjustment-Verfahren auf Bildmerkmalen. Die Software erlaubt es durch Einbeziehen von Bildregionen und Tiefenunsicherheit der Messungen effizient RGB-D-Bilder in kompakte Maps zu konvertieren. Die dabei entstehende Surfelmap wird in Octrees gespeichert.

2.1.1 Surfel

Surfel, die Kurzform für surface element, sind von einander unabhängige Punkte und bestehen im Wesentlichen aus Koordinaten und einer Textur. Mit einer Anhäufung von Surfels lassen sich so Oberflächen und Objekte repräsentieren.

2.1.2 Octrees

Als Octree bezeichnet man eine Baumstruktur mit einer Wurzel, bei der alle Knoten entweder genau 8 oder genau 0 Kinder haben. Diese Struktur wird in 3-dimensionalen Räumen verwendet, um Objekte zu unterteilen. Jeder Knoten entspricht daher einem Würfel, der von seinen Kindern in 8 gleich große Teile geteilt wird.

3. VERWANDTE ARBEITEN

3.1 RGB-D-SLAM (Endres et al.) [2]

RGB-D-SLAM extrahiert aus dem RGB-Bild Bildmerkmale und gibt ihnen die jeweilige Tiefe aus dem Tiefenbild der Kinect. Die Extraktion wurde mit 3 verschiedene Deskriptoren (SURF, SIFT und ORB) getestet und die Ergebnisse anhand Erfolg, durchschnittliche Laufzeit, Translations- und Rotationsfehler verglichen. Mit RANSAC wird aus den Featuren mit Tiefeninformationen durch paarweise Schätzung der Kamera-Transformation ein Posengraph erstellt, welcher in mehreren Iterationen mit g^2o optimiert wird. Die global konsistente Punktwolke wird abschließend per OctoMap in eine 3D-Belegungs-Grid-Map konvertiert. Zur Auswertung werden auch hier die „Freiburg“-Datensätze benutzt.

3.2 Integrating Depth and Color Cues for Dense Multi-Resolution Scene Mapping Using RGB-D Cameras (Jörg Stückler, Sven Behnke) [7]

Dieser Ansatz implementiert eine effiziente ICP-Tiefenbildregistrierung mit Farbhinweisen bereits auf CPU und erstellt dichte 3D-Maps von Innenräumen in Echtzeit. Das Umgebungsmodell wird inkrementell in Multi-Resolution Surfel Maps via Octrees gespeichert. Dadurch ist es möglich bei der Surfel-Assoziierung redundante Berechnungen zu sparen. Die Posen-Rekonstruktion der Kamera wird über Key Views mit jeweiliger Posenunsicherheit in einem probabilistischen Framework optimiert. Auch hier wird das Verfahren auf den „Freiburg“-Datensätzen ausgewertet.

3.3 RGB-D Mapping: Using Depth Cameras for Dense 3D Modeling of Indoor Environments (Peter Henry, et. al.) [4]

RGB-D Mapping erzeugt dichte 3D Modelle von Innenräumen. Aus dem Kamera-Input werden spärlich RGB-D-Feature extrahiert und mit aufeinanderfolgenden Bildern gematcht. Die Feature werden mit RANSAC verarbeitet und die entstehenden Assoziationen zur Initialisierung der folgenden ICP-Berechnung verwendet, was sich signifikant auf die Laufzeit von ICP auswirkt. ICP matcht die dichte Punktwolke, welche aus dem Tiefenbild der Kamera erstellt wird, mit dem bisherigen Modell. Unter berücksichtigung von loop-closure wird die entstehende Punktwolke in eine Surfel Map konvertiert.

3.4 KinectFusion: Real-Time Dense Surface Mapping and Tracking (Richard A. Newcombe, et. al.) [6]

Das Verfahren ist in der Lage Maps von komplexen Innenräumen unter sich ändernder Beleuchtung in Echtzeit zu erstellen. Aus den rohen Tiefenmessungen der Kinect werden zunächst Maps erstellt. Die Kamerapose wird nicht über Bildregistrierung, sondern über einen grob-zu-fein Iterative Closest Point (ICP) Algorithmus zwischen dem live-Tiefenbild und dem vorausberechneten Tiefenbild, welches aus dem globalen Umgebungsmodell erstellt wird, rekonstruiert. Mit der Pose lässt sich das Umgebungsmodell, welches über eine volumetrische „truncated signed distance function“ (TSDF) dargestellt wird. Mit der TSDF lässt sich, durch Projektion in eine virtuelle Kamera, ein Tiefenbild für den ICP-Algorithmus vorausberechnen. Damit dieses Tiefenbild möglichst vollständig ist, speichert die TSDF die Umgebung mit dichten Oberflächen.

4. EIGENER ANSATZ

Dieser Ansatz orientiert sich analog zu „Probabilistic Robotics“ [9] an der Maximierung der Wahrscheinlichkeitsfunktion

$$p(x^A, x^B, m_1, ..., m_N \mid I^A, I^B), \tag{4.1}$$

die nach den 3D-Landmarkenpositionen $m_1, ..., m_N \in \mathbb{R}^3$ und Kameraposen $x^A \in \mathbb{R}^6$ und $x^B \in \mathbb{R}^6$ für die beiden zugehörigen Kameraaufnahmen I^A und I^B optimiert werden soll. Die Landmarken m_i ergeben sich aus Merkmalen, die zwischen Frame A und B assoziiert wurden und werden relativ zu Frame B angegeben. Die Kameraposen x^A und x^B setzen sich aus 3D-Position sowie Orientierung zusammen. I^A und I^B werden dabei mit ihren Punktmerkmalen $z_1^A, ..., z_N^A \in \mathbb{R}^3$ bzw. $z_1^B, ..., z_N^B \in \mathbb{R}^3$ identifiziert und in 2.5D dargestellt, das heißt, dass sie neben ihren Pixelkoordinaten auch Tiefeninformationen enthalten. Dieser Ansatz wird mit inverser Tiefe arbeiten. Definiere zur kompakteren Darstellung der folgenden Formeln abkürzend $z^A := z_1^A, ..., z_N^A$, $z^B := z_1^B, ..., z_N^B$, $s := x^A, x^B, m_1, ..., m_N$

4.1 Konstruktion einer Fehlerfunktion

Über die Bayes'sche Regel lässt sich diese Formel sehr einfach umformen:

$$\begin{aligned} &p(x^A, x^B, m_1, ..., m_N \mid I^A, I^B) \\ &\quad = \frac{p(z^A \mid z^B, x^A, x^B, m_1, ..., m_N) \cdot p(x^A, x^B, m_1, ..., m_N \mid z^B)}{p(z^A \mid z^B)} \\ &\quad = \frac{p(z^A \mid x^A, x^B, m_1, ..., m_N) \cdot p(x^A, x^B, m_1, ..., m_N \mid z^B)}{p(z^A)} \end{aligned} \tag{4.2}$$

Da die Messungen z^A und z^B konstant sind ist auch $p(z^A \mid z^B)$ konstant und wird daher im Folgenden durch $\eta_A := (p(z^A \mid z^B))^{-1}$ abgekürzt. η_B wird analog definiert.

Wegen Unabhängigkeit der einzelnen Merkmale $z_1^A, ..., z_N^A$ und Unabhängigkeit von x^B lässt sich der linke Teil des Zählers wie folgt vereinfachen:

$$\begin{aligned} p(z_1^A, ..., z_N^A \mid x^A, x^B, m_1, ..., m_N) &= \prod_{i=1}^{N} p(z_i^A \mid x^A, x^B, m_1, ..., m_N) \\ &= \prod_{i=1}^{N} p(z_i^A \mid x^A, m_1, ..., m_N) \end{aligned} \tag{4.3}$$

Der rechte Teil des Zählers lässt sich ähnlich wie in Gleichung 4.2 umformen und analog zu Gleichung 4.3 über Unabhängigkeit der $z_1^B, ..., z_N^B$ und Unabhängigkeit von x^A vereinfachen:

$$\begin{aligned} &p(x^A, x^B, m_1, ..., m_N \mid z^B) \\ &\quad = \frac{p(z^B \mid x^A, x^B, m_1, ..., m_N) \cdot p(x^A, x^B, m_1, ..., m_N)}{p(z^B)} \\ &\quad = \eta_B \cdot \prod_{i=1}^{N} p(z_i^B \mid x^B, m_1, ..., m_N) \cdot p(x^A, x^B, m_1, ..., m_N) \\ &\quad = \eta_B \cdot \prod_{i=1}^{N} p(z_i^B \mid x^B, m_1, ..., m_N) \cdot p(x^A) \cdot p(x^B) \cdot \prod_{i=1}^{N} p(m_i) \end{aligned} \tag{4.4}$$

Im letzten Teil der Gleichung wurde ausgenutzt, dass Posen und Merkmale unabhängig von einander sind, sodass sich ihre Wahrscheinlichkeiten separat voneinander betrachten lassen. Die Pose x^B und die Merkmale m_i sind uniform verteilt, weswegen diese Terme irrelevant sind. Ähnliches gilt für x^A, welches im Ursprung „festgenagelt“ wird und somit weggelassen werden kann. Zusammengefasst ergibt sich die Gleichung:

$$\begin{aligned} &p(x^A, x^B, m_1, ..., m_N \mid z^A, z^B) = \\ &\quad \eta_A \cdot \eta_B \cdot \prod_{i=1}^{N} p(z_i^A \mid x^A, m_1, ..., m_N) \cdot \prod_{i=1}^{N} p(z_i^B \mid x^B, m_1, ..., m_N) \end{aligned} \tag{4.5}$$

Die Wahrscheinlichkeiten der einzelnen Merkmale werden ebenso analog zu „Probabilistic Robotics“ [9] durch die Gleichungen

$$\begin{aligned} p(z_i^A \mid x^A, m_1, ..., m_N) &= \eta \cdot exp(-\frac{1}{2}(z_i^A - \hat{z}_i^A)^T (\Sigma_i^A)^{-1} (z_i^A - \hat{z}_i^A)) \\ p(z_i^B \mid x^B, m_1, ..., m_N) &= \eta \cdot exp(-\frac{1}{2}(z_i^B - \hat{z}_i^B)^T (\Sigma_i^B)^{-1} (z_i^B - \hat{z}_i^B)) \end{aligned} \tag{4.6}$$

berechnet. $\hat{z}_i$ ist die **erwartete Messung** der Landmarke i für das jeweilige Frame. Mehr zur Berechnung von $\hat{z}_i$ später. Die Unsicherheit der Messung z_i wird über die Kovarianzmatrix Σ_i modelliert. Durch anwenden des negative Logarithmus auf Gleichung 4.5 wird aus dem Maximierungs- ein Minimierungsproblem. Mit den Gleichungen 4.6 ergibt sich die Fehlerfunktion:

$$\begin{aligned}
&-log(p(x^A, x^B, m_1, ..., m_N \mid z^A, z^B)) \\
&\quad = const - \sum_{i=1}^{N} log(p(z_i^A \mid x^A, m_1, ..., m_N)) \\
&\qquad\quad - \sum_{i=1}^{N} log(p(z_i^B \mid x^B, m_1, ..., m_N)) \\
&\quad = const + \sum_{i=1}^{N} \frac{1}{2}(z_i^A - h(x^A, m_i))^T (\Sigma_i^A)^{-1} (z_i^A - h(x^A, m_i)) \\
&\qquad\quad + \sum_{i=1}^{N} \frac{1}{2}(z_i^B - h(x^B, m_i))^T (\Sigma_i^B)^{-1} (z_i^B - h(x^B, m_i))
\end{aligned} \tag{4.7}$$

Die Funktion evaluiert damit den Statusvektor $x := (s^T, m^T)^T \in \mathbb{R}^{6+N}$, bestehend aus der Pose $s := x^B$ und den vermuteten Landmarkenpositionen $m := (m_1^T, ..., m_N^T)^T \in \mathbb{R}^{3 \cdot N}$. Für die Minimalität dieser Fehlerfunktion spielt die Konstante *const* und der Faktor $\frac{1}{2}$ innerhalb der Summen keine Rolle und werden im weiteren Vorgehen weggelassen. Daraus ergibt sich die **Fehlerfunktion E**:

$$\begin{aligned}
E(x) := &\sum_{i}^{N} (\hat{z}_i^A - z_i^A)^T \cdot (\Sigma_i^A)^{-1} \cdot (\hat{z}_i^A - z_i^A) + \\
&\sum_{i}^{N} (\hat{z}_i^B - z_i^B)^T \cdot (\Sigma_i^B)^{-1} \cdot (\hat{z}_i^B - z_i^B)
\end{aligned} \tag{4.8}$$

Erwartete Messungen: Die erwarteten Messungen lassen sich über die jeweiligen Landmarkenpositionen und eine Transformation zwischen den beiden Kameraposen A und B berechnen. Die Transformation einer 3D-Koordinaten p^B aus Frame B in Frame A ergibt sich aus der Pose s und berechnet sich mit Rotation $R(s) \in \mathbb{R}^{3\times3}$ und Translation $Tr(s) \in \mathbb{R}^3$ über die Formel $p^A = R(s) \cdot p^B + Tr(s)$.

Um Landmarken in Messungen umzurechnen, wird eine **Projektionsfunktion** φ benötigt, welche eine Messung in 2.5D-Darstellung, bestehend

aus den Pixelkoordinaten p_x und p_y sowie der inversen Tiefe $\delta := z^{-1}$, unter Berücksichtigung der Kamerakalibrierung, auf ihre 3D-Koordinate $(x, y, z)^T$ projiziert.

$$\begin{aligned} &\varphi: \ 2.5D \to 3D \\ &(p_x, p_y, \delta) \mapsto (x, y, z) \end{aligned} \tag{4.9}$$

Im Vergleich zu ordinären Projektionsfunktionen, welche auf Pixelkoordinaten ohne Tiefeninformationen abbilden, wird bei Hinzunahme der inversen Tiefe die 3. Komponente des Pixelvektors durch die Tiefe dividiert:

$$\begin{aligned} \begin{pmatrix} p_x \\ p_y \\ \delta \end{pmatrix} &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & z^{-1} \end{pmatrix} \begin{pmatrix} p_x \\ p_y \\ 1 \end{pmatrix} \\ &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & z^{-1} \end{pmatrix} \cdot K \cdot \begin{pmatrix} x/z \\ y/z \\ 1 \end{pmatrix} \\ &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & z^{-1} \end{pmatrix} \cdot K \cdot \Big(z^{-1} \begin{pmatrix} x \\ y \\ z \end{pmatrix} \Big) \\ &= \Psi_z \cdot K \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix} =: \varphi^{-1}\Big(\begin{pmatrix} x \\ y \\ z \end{pmatrix} \Big) \end{aligned} \tag{4.10}$$

K bezeichnet die Kalibrierungsmatrix und Ψ_a die Umkehrung der Projektion mit inverser Tiefe:

$$K = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 525 & 0 & 319,5 \\ 0 & 525 & 239,5 \\ 0 & 0 & 1 \end{pmatrix}, \Psi_a := \begin{pmatrix} a^{-1} & 0 & 0 \\ 0 & a^{-1} & 0 \\ 0 & 0 & a^{-2} \end{pmatrix} \tag{4.11}$$

Die Parameter f_x und f_y sind die jeweiligen Brennweiten der Kamera. $(c_x, c_y)^T$ ist die Bildmitte der Kamera, angegeben in Pixeln.
Für φ gilt dann:

$$\varphi\Big(\begin{pmatrix} p_x \\ p_y \\ \delta \end{pmatrix} \Big) = K^{-1} \cdot \Psi_\delta \cdot \begin{pmatrix} p_x \\ p_y \\ \delta \end{pmatrix} \tag{4.12}$$

Die Funktion $h(s, z_i)$ transformiert, unter Verwendung der durch s gegebenen Transformation, eine Messung z_i^B aus Frame B in eine erwartete Messung in Frame A und ergibt sich durch Projektion der Messung, Transformation des 3D-Punkts und Projektion ins Bild:

$$h(s, z_i) = \varphi^{-1}\big(R(s) \cdot \varphi(z_i) + Tr(s)\big) \tag{4.13}$$

Für eine gegebene Landmarke m_i und eine Pose s errechnet sich die erwartete Messung dann über die Gleichung:

$$\hat{z}_i := h\Big(s, \varphi^{-1}(m_i)\Big) \tag{4.14}$$

4.2 Minimierung der Fehlerfunktion

Über ein Gradientenabstiegsverfahren auf $E(x)$ ist es möglich den Statusvektor x zu optimieren. Hierfür wird $E(x)$, alternativ zur Summation über alle Landmarken in beiden Kameraposen, über Matrizen berechnet. Die Fehlerterme $f_i^A := \hat{z}_i^A - z_i^A$ bzw. $f_i^B := \hat{z}_i^B - z_i^B$ aus Gleichung 4.8 werden wie folgt zu Vektoren gestapelt:

$$f^A := \begin{pmatrix} f_1^A \\ \vdots \\ f_N^A \end{pmatrix}, f^B := \begin{pmatrix} f_1^B \\ \vdots \\ f_N^B \end{pmatrix}, f := \begin{pmatrix} f^A \\ f^B \end{pmatrix} \tag{4.15}$$

Die Kovarianzmatrizen Σ_i^A bzw. Σ_i^B werden zu blockdiagonalen Matrizen zusammengefasst:

$$\Sigma^A := \begin{pmatrix} \Sigma_1^A & & 0 \\ & \ddots & \\ 0 & & \Sigma_N^A \end{pmatrix}, \Sigma^B := \begin{pmatrix} \Sigma_1^B & & 0 \\ & \ddots & \\ 0 & & \Sigma_N^B \end{pmatrix}, \Sigma := \begin{pmatrix} \Sigma^A & 0 \\ 0 & \Sigma^B \end{pmatrix} \tag{4.16}$$

Das Inverse einer blockdiagonalen Matrix ist wiederum blockdiagonal und besteht aus den Inversen der jeweiligen Blöcke, sodass Σ^{-1} aus den $(\Sigma_i)^{-1}$ der jeweiligen Frames aus Gleichung 4.8 besteht. $E(x)$ berechnet sich nun durch:

$$\begin{aligned} E(x) &= (f^A)^T \cdot (\Sigma^A)^{-1} \cdot f^A + (f^B)^T \cdot (\Sigma^B)^{-1} \cdot f^B \\ &= f^T \cdot \Sigma \cdot f \end{aligned} \tag{4.17}$$

Um den Statusvektor x so anzupassen, dass die Fehlerfunktion minimiert wird, wird der Gradient von f genutzt. Hierbei kann es prinzipiell vorkommen, dass durch einen ungünstigen Startwert, hier der Statusvektor, ein schlechtes lokales Minimum gefunden und weiterverwendet wird. Da es aber in der Natur von Aufnahmesequenzen liegt, dass aufeinanderfolgende Aufnahmen örtlich sehr nah beieinander liegen, sodass der gesuchte Statusvektor bereits in der Nähe des aktuellen liegt, konvergiert das Verfahren mit großer Wahrscheinlichkeit gegen einen brauchbaren Wert.
f ist, wenn auch nicht explizit angegeben, abhängig vom Statusvektor und

wird von nun an als Funktion auf x betrachtet. Stapelt man die erwarteten und gemessenen Punktmerkmale wie in den folgenden Definitionen

$$h^A(x) := \begin{pmatrix} h(s, \varphi^{-1}(m_1)) \\ \vdots \\ h(s, \varphi^{-1}(m_N)) \end{pmatrix}, \; h^B(x) := \begin{pmatrix} h(0, \varphi^{-1}(m_N)) \\ \vdots \\ h(0, \varphi^{-1}(m_N)) \end{pmatrix}$$

$$h(x) := \begin{pmatrix} h^A(x) \\ h^B(x) \end{pmatrix} \in \mathbb{R}^{2 \cdot 3 \cdot N} \tag{4.18}$$

$$z^A := \begin{pmatrix} z_1^A \\ \vdots \\ z_N^A \end{pmatrix}, \; z^B := \begin{pmatrix} z_1^B \\ \vdots \\ z_N^B \end{pmatrix}, \; z := \begin{pmatrix} z^A \\ z^B \end{pmatrix} \in \mathbb{R}^{2 \cdot 3 \cdot N}$$

errechnet sich $f(x)$ aus der Differenz der beiden Vektoren.

$$f(x) = h(x) - z \tag{4.19}$$

Diese Darstellung splittet f in die von x abhängigen und unabhängigen Komponenten auf, was sie für den Gradienten von f interessant macht:

$$\begin{aligned} f(x + \Delta) &\approx f(x) + \frac{df}{dx}(x) \cdot \Delta \\ &= f(x) + \frac{dh}{dx}(x) \cdot \Delta \\ &= f(x) + J \cdot \Delta, \; J := \frac{dh}{dx}(x) \in \mathbb{R}^{2 \cdot 3 \cdot N \times 6 + N} \end{aligned} \tag{4.20}$$

Bei der Ableitung von h nach x wird die Funktion nach den Komponenten von x abgeleitet, sprich nach der Pose s und den geschätzen Landmarkenpositionen m, sodass sich die Jacobi-Matrix J in 4 Matrizen aufspaltet:

$$J = \begin{pmatrix} \frac{dh^A}{ds}(s) & \frac{dh^A}{dm}(m) \\ \frac{dh^B}{ds}(s) & \frac{dh^B}{dm}(m) \end{pmatrix} \tag{4.21}$$

4.3 Berechnung der Jacobi-Matrix

Die Projektion φ^{-1} in $h(x, z)$ nutzt die Tiefe der transformierten Landmarke, welche im Weiteren durch α definiert ist:

$$h(s, z) = \varphi^{-1}\big(\underbrace{R(s) \cdot \varphi(z) + Tr(s)}_{=:\alpha(s,z)=:\alpha} \big) = \varphi^{-1}(\alpha(s, z)) \tag{4.22}$$

Um $h(s, z)$ nach Pose und Merkmalen abzuleiten, wird jeweils die Kettenregel angewandt:

$$\begin{aligned} \frac{dh}{ds} &= \frac{d\varphi^{-1}}{d\alpha}(\alpha) \cdot \frac{d\alpha}{ds}(s, z) \\ \frac{dh}{dm} &= \frac{d\varphi^{-1}}{d\alpha}(\alpha) \cdot \frac{d\alpha}{dm}(s, z) \end{aligned} \tag{4.23}$$

Programmintern wird auf den 2.5D-Messungen der Landmarken gearbeitet, weswegen hier nicht $h(s, \varphi^{-1}(m))$ betrachtet werden muss.

Ableitung von $\varphi^{-1}(\alpha)$ nach α

$$\begin{aligned} \frac{d\varphi^{-1}}{d\alpha_x} &= \Psi_{\alpha_z} \cdot K \cdot \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = \Psi_{\alpha_z} \cdot \begin{pmatrix} f \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{f}{\alpha_z} \\ 0 \\ 0 \end{pmatrix} \\ \frac{d\varphi^{-1}}{d\alpha_y} &= \Psi_{\alpha_z} \cdot K \cdot \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} = \Psi_{\alpha_z} \cdot \begin{pmatrix} 0 \\ f \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ \frac{f}{\alpha_z} \\ 0 \end{pmatrix} \\ \frac{d\varphi^{-1}}{d\alpha_z} &= \begin{pmatrix} -\alpha_z^{-2} & 0 & 0 \\ 0 & -\alpha_z^{-2} & 0 \\ 0 & 0 & -2\alpha_z^{-3} \end{pmatrix} \cdot K \cdot \begin{pmatrix} \alpha_x \\ \alpha_y \\ \alpha_z \end{pmatrix} + \Psi_{\alpha_z} \cdot K \cdot \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \end{aligned} \tag{4.24}$$

Jede dieser Ableitungen ist ein Vektor im $\mathbb{R}^3$. Schreibt man diese Vektoren nebeneinander ergibt sich Ableitung nach α im Ganzen:

$$\frac{d\varphi^{-1}}{d\alpha} = \begin{pmatrix} \frac{d\varphi^{-1}}{d\alpha_x} & \frac{d\varphi^{-1}}{d\alpha_y} & \frac{d\varphi^{-1}}{d\alpha_z} \end{pmatrix} \in \mathbb{R}^{3\times 3} \tag{4.25}$$

Ableitung von $\alpha(s, m)$ nach s
Die Pose s setzt sich, wie anfangs bereits erwähnt, aus einem 3-dimensionalen Punkt und einer Orientierung zusammen und hat die Form $s = (s_x, s_y, s_z, \theta_x, \theta_y, \theta_z)$. Bei den θ-Komponenten handelt es sich um Quaternionen, die, wie der Name bereits vermuten lässt, eigentlich aus 4 Komponenten bestehen. Die vierte Komponente ist in diesem Fall redundant und kann für die Berechnung weggelassen werden.

$$\frac{d\alpha}{s_x} = \frac{dTr}{ds_x} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \tag{4.26}$$

$\frac{d\alpha}{s_y}$ und $\frac{d\alpha}{s_z}$ ergeben sich analog mit entsprechendem Eintrag im Einheitsvektor. Auf die Ableitungen nach den Quaternionen wird nicht im Detail eingegangen, diese berechnen sich durch:

$$\frac{d\alpha}{d\theta} = \frac{dR}{d\theta} \cdot \varphi(z) \tag{4.27}$$

Ableitung von $\alpha(s, m)$ nach m

$$\begin{aligned}
\frac{d\alpha}{dm} &= R(x) \cdot \frac{d\varphi}{dm} \\
&= R(x) \cdot [\frac{d\varphi}{dm_x},\ \frac{d\varphi}{dm_y},\ \frac{d\varphi}{dm_\delta}] \ \in \ \mathbb{R}^{3\times 3} \\
\frac{d\varphi}{dm_x} &= K^{-1} \cdot \Psi_\delta \cdot \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \\
\frac{d\varphi}{dm_y} &= K^{-1} \cdot \Psi_\delta \cdot \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \\
\frac{d\varphi}{dm_\delta} &= K^{-1} \cdot \begin{pmatrix} -\alpha_z^{-2} & 0 & 0 \\ 0 & -\alpha_z^{-2} & 0 \\ 0 & 0 & -2\alpha_z^{-3} \end{pmatrix} \cdot \begin{pmatrix} m_x \\ m_y \\ m_\delta \end{pmatrix} + K^{-1} \cdot \Psi_\delta \cdot \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}
\end{aligned} \tag{4.28}$$

Das Produkt der Ableitung entsprechend Gleichung 4.23 ergibt die jeweilige Ableitung nach s bzw. m. Die Ableitungen werden für alle Einträge des gestapelten Vektors $h(x)$ angewendet.
Viele dieser Einträge sind jedoch unabhängig von dem Term, nach dem abgeleitet wird, sodass in der Jacobi-Matrix viele 0-Einträge entstehen. Während $h^A(x)$ vom gesamten Statusvektor abhängig ist, benötigt $h^B(x)$ nur die Landmarken und nutzt für die Transformation statt s die Identität. So ist die gesamte, nach s abgeleitete Matrix $\frac{dh^B}{ds} = 0 \ \in \ \mathbb{R}^{3\cdot\mathbb{N}\times 6}$.
Ähnliches gilt für die Ableitung nach den Landmarken. Die Funktion $h(s, \varphi^{-1}(m_i))$ ist von allen Landmarken m_j mit $i \neq j$ unabhängig, sodass beim Ableiten von $h^A(x)$ bzw. $h^B(x)$ blockdiagonale Matrizen entstehen. Die

Jacobi-Matrix hat damit folgende Form:

$$J = \left(\begin{array}{c|ccc} & \frac{d\varphi^{-1}}{d\alpha}(\alpha)\cdot\frac{d\alpha}{dm_1}(s,\hat{z}_1^A) & & 0 \\ \frac{dh^A}{ds}(x) & & \ddots & \\ & 0 & & \frac{d\varphi^{-1}}{d\alpha}(\alpha)\cdot\frac{d\alpha}{dm_N}(s,\hat{z}_1^A) \\ \hline & \frac{d\varphi^{-1}}{d\alpha}(\alpha)\cdot\frac{d\alpha}{dm_1}(0,\hat{z}_1^B) & & 0 \\ 0 & & \ddots & \\ & 0 & & \frac{d\varphi^{-1}}{d\alpha}(\alpha)\cdot\frac{d\alpha}{dm_N}(0,\hat{z}_1^B) \end{array}\right) \tag{4.29}$$

$\hat{z}_i^A$ und $\hat{z}_i^B$ bezeichnen die ins jeweilige Frame projizierte geschätze Landmarke m_i, dh.:

$$\begin{aligned} \hat{z}_i^A &:= h\Big(s, \varphi^{-1}(m_i)\Big) \\ \hat{z}_i^B &:= h\Big(0, \varphi^{-1}(m_i)\Big) = \varphi^{-1}(m_i) \end{aligned} \tag{4.30}$$

4.4 Optimierung des Statusvektors

Mit dem Levenberg-Marquardt Verfahren aus [1] lässt sich für den Statusvektor x ein Update Δx berechnen, mit dem bei passender Schrittweite die Fehlerfunktion $E(x)$ minimiert wird.

$$\begin{aligned} (J^T\Sigma^{-1}J + \mu\cdot I)\Delta x = -J^T\cdot\Sigma^{-1}\cdot f \Leftrightarrow \\ \Delta x = -(J^T\Sigma^{-1}J + \mu\cdot I)^{-1}J^T\cdot\Sigma^{-1}\cdot f \end{aligned} \tag{4.31}$$

Das Update Δx wird, skaliert mit einem Wert $\alpha \in \mathbb{R}_+$, auf x aufaddiert.

$$x^{(j+1)} := x^{(j)} + \alpha\cdot\Delta x \tag{4.32}$$

Der entstehende Vektor x' wird beibehalten, sofern $E(x') < E(x)$. Ansonsten werden interne Parameter des Levenberg-Marquardt Verfahrens angepasst und eine neue Iteration gestartet. Durch ausreichend viele Durchläufe werden iterativ passendere Kamera- und Landmarkenposen errechnet, bis die Folge $E(x^{(j)})$ konvergiert oder die maximale Anzahl an Iterationen überschritten wird.
Auch wenn diese Berechnung $E(x)$ minimiert ist sie noch viel zu ineffizient, um sie auf einen Input-Stream anzuwenden. Um gute Ergebnisse zu erzielen werden viele Landmarken gebraucht. Viele Landmarken bedeuten aber auch gleichzeitig wesentlich mehr Berechnungen. Bei N gematchten Landmarken fasst die Matrix $(J^T\cdot\Sigma^{-1}\cdot J+\mu\cdot I)$ genau $(6+3\cdot N)\times(6+3\cdot N)$ Einträge. In

den hier verwendeten Datensätzen werden zwischen 2 Frames ca. $200 - 400$ gemeinsame Landmarken gefunden. In jeder der oben genannten Iterationen müsste demnach für beispielsweise 300 Landmarken eine Matrix mit 820.836 Einträge berechnet und invertiert werden.

4.5 Effizientere Berechnung

g^2o macht sich zur Lösung dieses Problems die Struktur der zu invertierenden Matrix zu Nutze. Dafür sei die Matrix H und der Vektor b definiert durch:

$$\begin{aligned} H &:= \begin{pmatrix} H_{pp} & H_{pl} \\ H_{lp} & H_{ll} \end{pmatrix} = J^T \cdot \Sigma^{-1} \cdot J + \mu \cdot I \\ b &:= \begin{pmatrix} b_p \\ b_l \end{pmatrix} = J^T \cdot \Sigma^{-1} \cdot f \end{aligned} \tag{4.33}$$

Hierbei steht p für die Pose mit jeweils 6 und l für die Landmarken mit jeweils $3 \cdot N$ Einträgen. Die Teilmatrizen und -vektoren sind definiert durch:

$$\begin{aligned} H_{pp} &:= \frac{dh}{ds}(x)^T \cdot \Sigma^{-1} \cdot \frac{dh}{ds}(x) + \mu \cdot I \\ H_{lp} &:= \frac{dh}{ds}(x)^T \cdot \Sigma^{-1} \cdot \frac{dh}{dm}(x) \\ H_{pl} &:= \frac{dh}{dm}(x)^T \cdot \Sigma^{-1} \cdot \frac{dh}{ds}(x) \\ h_{ll} &:= \frac{dh}{dm}(x)^T \cdot \Sigma^{-1} \cdot \frac{dh}{dm}(x) + \mu \cdot I \\ b_p &:= \frac{dh}{ds}(x)^T \cdot \Sigma^{-1} \cdot f \\ b_l &:= \frac{dh}{dm}(x)^T \cdot \Sigma^{-1} \cdot f \end{aligned} \tag{4.34}$$

Die oben vorgestellte Gleichung 4.31 zur Berechnung von Δx hat dann die Form:

$$\begin{pmatrix} H_{pp} & H_{pl} \\ H_{lp} & H_{ll} \end{pmatrix} \cdot \begin{pmatrix} \Delta x_p \\ \Delta x_l \end{pmatrix} = - \begin{pmatrix} b_p \\ b_l \end{pmatrix} \tag{4.35}$$

Mit dem Schur-Komplement von H ist es nun möglich ein „äquivalentes reduziertes System“ zu erzeugen, mit dem sich Posen- und Landmarken-Updates getrennt voneinander berechnen lassen.

$$(H_{pp} - H_{pl} \cdot H_{ll}^{-1} \cdot H_{pl}^T)\Delta x_p = -b_p + H_{pl} H_{ll}^{-1} b_l \tag{4.36}$$

Ausgehend von Δx_p ergibt sich das Landmarken-Update Δx_l dann aus:

$$(H_{ll} + \mu \cdot I)\Delta x_l = -b_l - H_{pl}^T \Delta x_p \tag{4.37}$$

Für die Matrix H ist es nicht nötig beide Teilmatrizen H_{pl} und H_{lp} zu speichern, da H wegen Symmetrie von Σ selbst symmetrisch ist. Statt H_{lp} wird deshalb H_{pl}^T verwendet. Die Matrix $(H_{pp} - H_{pl} \cdot H_{ll}^{-1} \cdot H_{pl}^T)$ fasst lediglich 6×6 Einträge, sodass die Inverse dieser Matrix schnell berechnet ist. Gleiches gilt für die Invertierung der Teilmatrix H_{ll}, welche zwar mit $3 \cdot N \times 3 \cdot N$ wesentlich größer, aber durch die Struktur der Jacobi-Matrix J blockdiagonal und deshalb, genau wie Σ, leicht zu invertieren ist.

5. IMPLEMENTIERUNG

Die Berechnung aus Kapitel 4 wird zusammen mit den Surfeltermen im Levenberg-Marquardt-Verfahren [3] untergebracht. Hier werden in mehreren Iterationen die Landmarkenpositionen sowie die Kamerapose optimiert.

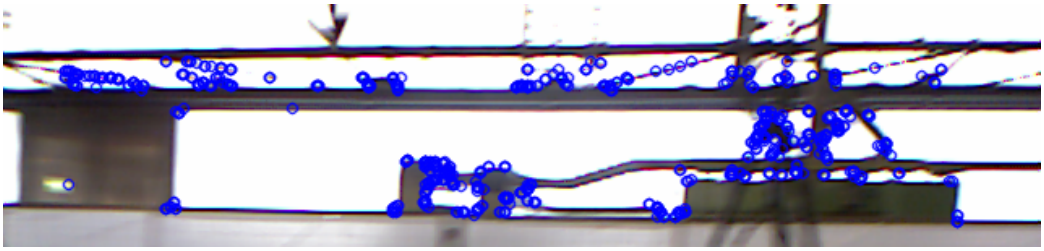

Fig. 5.1: Geschätzte Landmarkenpositionen vor der Optimierung in Fr2 Large with loop

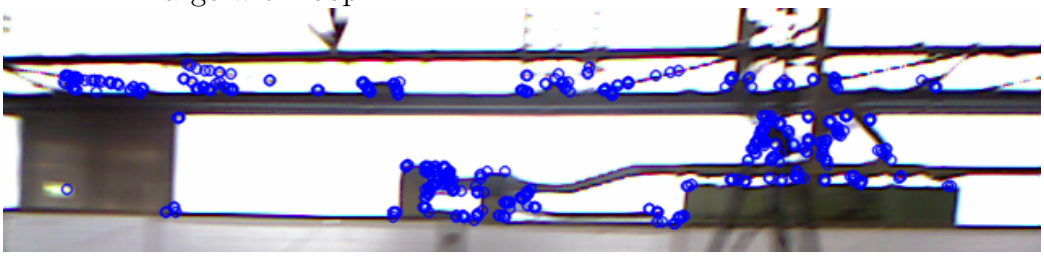

Fig. 5.2: Geschätzte Landmarkenpositionen während der Optimierung in Fr2 Large with loop

5.1 Globale Features

Die durch das Bundle Adjustment optimierten Landmarkenpositionen werden nach der Schätzung der Kameratransformation verworfen und in der nächsten Iteration nichtmehr berücksichtigt. Wenn die optimierten Positionen erhalten bleiben, könnte damit eine genauere Transformation in den folgenden Iterationen geschätzt werden.

5.2 Grid

Nicht selten kommt es vor, dass sich sehr viele Merkmale auf engstem Raum befinden. Homogenfarbene Flächen sind davon nicht betroffen. Dieses Verhalten tritt vor allem an chaotisch wirkenden Bildregionen, wie zum Beispiel Pflanzen, auf. Solche Anhäufungen von Merkmalen scheinen die Transformationsschätzung nicht zu verbessern. Viel mehr scheint es sinnvoller die Merkmale stattdessen gleichmäßig über das Bild zu verteilen, dafür schwächere Merkmale in Kauf zu nehmen und dadurch die Trackinggenauigkeit zu erhöhen.

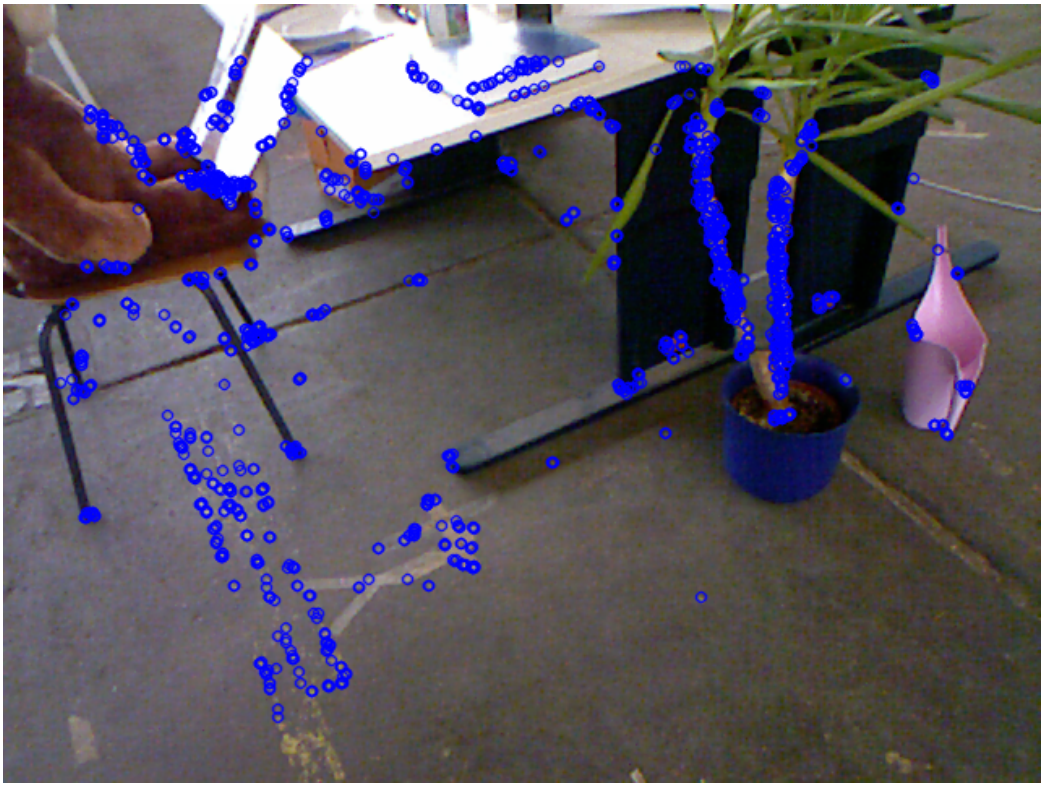

Fig. 5.3: Anhäufung von Bildmerkmalen

Zu diesem Zweck wird auf das Bild ein Raster bzw. eine Grid gelegt, sodass die im Bild gefundenen Merkmale darin aufgeteilt werden. Die Anzahl an Merkmalen pro Zelle wird nach oben limitiert und so die Ansammlung von Merkmalen reduziert. Dabei werden die Merkmale innerhalb ihrer Zelle nach ihrer Stärke sortiert und die schwächsten ggf. verworfen. Gleichzeitig wird

die Anzahl der zu detektierenden Merkmale im Bild wesentlich erhöht, um die verworfenen Merkmale zu kompensieren.

Anzahl Zeilen und Spalten der Grid sowie die maximale Anzahl der Merkmale pro Zelle und die Gesamtzahl aller im Bild zu detektierenden Merkmale können durch Parameter beim Aufruf festgelegt werden.

5.3 Unsicherheiten

Die im Verfahren extrahierten Bildpunkte sind Messungen, die prinzipiell Unsicherheiten mit sich bringen. Diese Unsicherheiten finden sich in den Covarianzmatrizen der einzelnen Landmarkenmessungen wieder und werden mit entsprechenden Werte skaliert. Davon ist nicht nur die Tiefenschätzung, sondern auch die Pixelposition betroffen.

Die Implementierung von Unsicherheiten für Pixelposition und Tiefe wurden so angepasst, dass sie beim Aufruf des Programms via Parameter modifiziert werden können. Der Parameter für das Rauschen der Pixel geht quadratisch in die ersten beiden Diagonaleinträge der Covarianzmatrix ein. Der Parameter für das Tiefenrauschen ist linear im 3. Diagonaleintrag.

6. EVALUATION

In diesem Kapitel wird eine Reihe von Experimenten ausgeführt und deren Ergebnisse mit denen anderer Verfahren verglichen.

6.1 Datensätze

Die Auswertung erfolgt auf öffentlich zugänglichen RGB-D-Datensätzen von Jürgen Sturm et al. [8], welche von der Technischen Universität München zur Verfügung gestellt werden. Da auch andere Ansätze mit diesen Datensätzen arbeiten, können die Verfahren direkt miteinander verglichen werden.

Die Datensätze enthalten neben den RGB-D-Sequenzen eine Groundtruth-Datei, welche den Verlauf von Pose und Orientierung der RGB-D-Kamera beinhaltet. Der Bewegungsablauf der Kamera wurde mit einem hochpräzisen Motion-Capture-System, bestehend aus 8 Kameras mit jeweils 100 Hertz, erstellt.

Die Aufnahmen konzentrieren sich auf Kameraführung und verschiedene Umstände der Umgebung. Dazu gehören kleine und große Entfernungen der Objekte zur Kamera, Informationsgehalt der Tiefenbilder, gesonderte Rotations- und Translationstests, Bewegungsgeschwindigkeit der Kamera und Schleifen, das heißt, dass die Kamera zu einer bekannten Szenerie zurückkehrt. Mit Ausnahme von sich ggf. bewegenden Kabeln oder Personen sind alle Umgebungen statisch.

6.2 Bewertung der Trajektorie

Die durch die Software rekonstruierte Trajektorie der Kamera wird mit RPE (relative pose error) ausgewertet. Rotations- und Translationsfehler der relativen Posen werden dabei bezüglich rmse (root mean square error), arithmetischem Mittel bzw. Mean, Median, Standardabweichung, Minimum und Maximum betrachtet.

6.3 Grid

In diesem Teil wird das Verhalten der Datensätze für Grids verschiedener Größen, Zell-Limits und Gesamtfeature getestet. Surfelmatching und Unsicherheiten sind gewählt wie bei globalen Features. Zum Einsatz kommt unter Anderem ein 3×3 Raster mit maximal 222 Bildmerkmalen pro Zelle und 2000 detektierten Merkmalen auf dem gesamten Bild - in der Tabelle mit „3x3x222.2000“ abgekürzt. Weitere Grids sind „5x5x60.3000“, „7x7x30.3000“ und „11x11x13.3500“. Die Werte sind so gewählt, dass eine vergleichbare Anzahl Merkmale nach dem Aussortieren übrig bleibt.

Die Tabelle auf Seite 22 zeigt die Ergebnisse der Auswertung. Die besten Ergebnisse für Median und Maximum von Translation und Rotation sind **fett** markiert.
Im Allgemeinen fällt der Median kleiner aus, wenn die Szenerie viele nahe Objekten enthält und die Kamera sehr ruhig und kontrolliert geführt wird. Das Maximum verschlechtert sich schon bei wenigen schlechten Bildabläufen.

Unglücklicherweise gibt es keine Grid, die allen anderen überlegen ist. Dennoch zeigen sich Vorteile durch die Verwendung einiger Grids. Von den 52 Bestwerten fallen 48 Stück auf Grid-Varianten. Zudem verbessern sich alle bis auf einen Median, wenn die 3×3-Grid benutzt wird, statt garkeine zu verwenden. Die Grids 5×5 und 7×7 verhalten sich hierbei ähnlich gut. Die meisten Verschlechterungen der Mediane entstehen beim Anwenden der 11×11-Grid. Jedoch liefert diese Grid die besten Mediane für die Datensätze „fr1 plant“ und „fr1 teddy“ - beides Datensätze, bei denen sich die Kamera um 360° um ein Objekt bewegt, wobei die restlichen Objekte bis zu 3 Metern entfernt sind. Für „fr2 desk“, der einzige weitere Datensatz, bei dem sich die Kamera um 360° um ein Objekt bewegt, wird der Median jedoch leicht schlechter, während sich die anderen Werte etwas verbessern. Die Kamera hält hier einen größeren Abstand von 1-2m zum Objekt, 2 Schreibtische mit Accessoires, ein. Die sehr starke Verteilung der Bildpunkte auf fernere Objekte und den featurearmen Boden scheint hier die optimierenden Eigenschaften von Grids aufzuheben.

Datensatz	keine Grid	3x3x222.2000	5x5x60.3000	7x7x30.3000	11x11x13.3500
fr1 360	0.006939m (0.346633m)	0.006881m (0.346633m)	**0.006768m (0.346632m)**	0.007127 (0.346633)	0.007099m (0.346632m)
	0.365399° (11.858897°)	0.365306° (11.858901°)	**0.364242°** (11.858941°)	0.369002 (**11.858841**)	0.378903° (11.858906°)
fr1 desk	0.007727m (**0.050686m**)	0.007250m (0.108760m)	0.007333m (0.137076m)	**0.007248m** (0.198190m)	0.007391m (0.095277m)
	0.417614° (**4.034406°**)	0.405697° (8.038292°)	**0.402092°** (16.720771°)	0.415028° (8.963371°)	0.424153° (8.825490°)
fr1 desk2	0.007669m (0.275151m)	**0.007327m (0.059907m)**	0.007553m (0.323506m)	0.007466m (0.799979m)	0.007497m (0.316941m)
	0.417066° (15.201401°)	0.403695° (**4.680374°**)	**0.402086°** (13.175951°)	0.410260° (34.743336°)	0.415231° (15.500500°)
fr1 floor	0.002108m (0.412351m)	**0.002047m** (0.545241m)	0.002057m (0.412351m)	0.002072m (0.412351m)	0.002127m (0.505304m)
	0.186324° (21.039443°)	0.185031° (21.423700°)	0.184802° (**15.815136°**)	**0.184587°** (18.540545°)	0.184657° (18.066312°)
fr1 plant	0.004444m (0.034032m)	0.004090m (**0.029671m**)	0.004133m (0.032098m)	0.003975m (0.030943m)	**0.003903m** (0.031465m)
	0.287766° (3.209152°)	0.280224° (3.195704°)	0.278609° (3.203073°)	0.275904° (3.254741°)	**0.275350°** (**3.090089°**)
fr1 room	0.005237m (**0.037138m**)	**0.005076m** (0.037139m)	0.005228m (0.041488m)	0.005154m (0.046381m)	0.005450m (0.056989m)
	0.332792° (3.449693°)	**0.325244°** (3.510729°)	0.328093° (**3.449692°**)	0.329809° (3.449694°)	0.337989° (3.759971°)
fr1 rpy	0.004194m (0.150703m)	0.004083m (0.057434m)	0.004039m (0.065338m)	0.004034m (**0.049043m**)	**0.003959m** (0.051880m)
	0.400637° (15.576924°)	0.403584° (3.967310°)	0.405976° (**3.953873°**)	0.408182° (3.995841°)	0.401600° (5.550775°)
fr1 teddy	0.007122m (0.784027m)	0.006511m (0.784027m)	0.006409m (0.784025m)	0.006359m (0.768039m)	**0.006291m (0.269291m)**
	0.354766° (14.553854°)	0.347704° (14.553893°)	0.349177° (14.553825°)	0.349944° (24.289500°)	**0.346694°** (**7.811487°**)
fr1 xyz	0.004456m (0.030049m)	**0.004241m** (0.027136m)	0.004560m (0.022512m)	0.004474m (0.022317m)	0.004439m (**0.020807m**)
	0.272074° (1.699718°)	**0.267665°** (1.570613°)	0.270399° (1.341104°)	0.278439° (1.379643°)	0.272507° (**1.303238°**)
fr2 desk	0.002656m (0.019981m)	**0.002517m** (0.017307m)	0.002604m (0.014736m)	0.002588m (**0.013741m**)	0.002678m (0.016370m)
	0.199240° (1.354480°)	0.196084° (1.367999°)	**0.194533°** (1.346775°)	0.194666° (**1.328041°**)	0.194796° (1.342401°)
fr2 loop	0.013547m (0.373115m)	0.012482m (0.283043m)	**0.012286m** (0.204540m)	0.012717m (**0.191034m**)	0.013391m (0.253217m)
	0.210318° (1.676148°)	0.205148° (1.562046°)	**0.199800°** (1.399591°)	0.200408° (1.486852°)	0.207052° (**1.304682°**)
fr2 rpy	0.001634m (0.038336m)	**0.001598m** (0.023473m)	0.001630m (0.021738m)	0.001729m (0.023958m)	0.001791m (**0.017643m**)
	0.118852° (1.752327°)	**0.113184°** (1.035882°)	0.113913° (**0.935756°**)	0.114103° (1.056761°)	0.119569° (0.973292°)
fr2 xyz	0.001624m (0.017240m)	**0.001584m** (0.011249m)	0.001619m (**0.010643m**)	0.001640m (0.011039m)	0.001696m (0.014136m)
	0.153624° (1.244329°)	0.146704° (1.243644°)	0.146890° (**1.228907°**)	0.148018° (1.235992°)	**0.146500°** (1.237124°)

6.4 Pixel- und Tiefenunsicherheit

An dieser Stelle werden verschiedene Parameter für Pixel- und Tiefenunsicherheit in Kombination getestet. Wie vorher auch, werden die Surfel deaktiviert, um stabilisierende Effekte zu vermeiden. Bei allen Durchläufen wurden weder Grids, noch globale Features verwendet. Im gesamten Bild werden 1000 Bildmerkmale benutzt.

Zuvor wurden im Programm für Pixel- und Tiefenunsicherheit die Fixwerte 5 bzw. 4 benutzt. Für Pixelunsicherheit werden zunächst die diskreten Werte von 1 bis 7 in Kombination mit den diskreten Werten von 2 bis 6 für Tiefenunsicherheit verwendet. Der durchschnittliche Translationsmedian über alle Datensätze wurde dann für jede Parameterkombination geplottet, um Tendenzen für bessere Parameter zu finden. Der durschnittliche

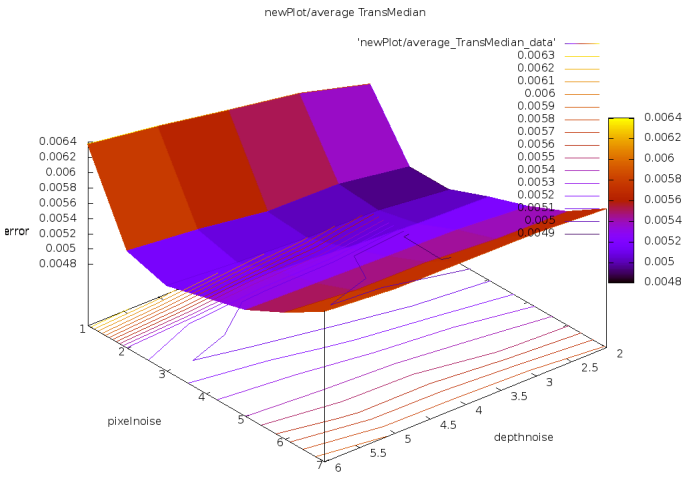

Fig. 6.1: Durchschnittlicher Translationsmedian für verschiedene Pixel- und Tiefenunsicherheiten

Translationsmedian in Bild 6.1 nimmt sein Minimum bei *depthnoise* 2 und *pixelnoise* 2 bzw. *pixelnoise* 3 an. Der durchschnittliche Rotationsmedian in Grafik 6.2 zeigt ein ähnliches Ergebnis mit *depthnoise* 2 und *pixelnoise* 3.

Die Grafiken legen die Vermutung nahe mit einer noch kleineren Tiefenunsicherheit den Fehler weiter reduzieren zu können. Der Wertebereich für Pixelunsicherheit kann noch weiter eingeschränkt werden. Im Weiteren werden neue Intervalle für die Unsicherheiten gewählt und die Schrittweite verkürzt, um genauere Ergebnisse zu bekommen.

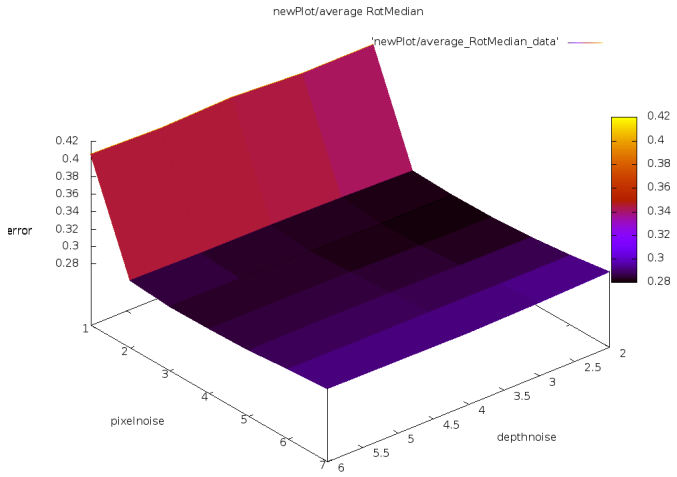

Fig. 6.2: Durchschnittlicher Rotationsmedian für verschiedene Pixel- und Tiefenunsicherheiten

Für *pixelnoise* wird nun das Intervall $[2,5;5]$ in $0,5$er-Schritten ausgewertet. *depthnoise* wird mit $0,153125$er-Schritten im Intervall $[0,1;5]$ wesentlich genauer betrachtet. Die Ergebnisse der globalen Feature wurde erst nach diesem Teil des Experiments ausgewertet, weshalb sie hier noch aktiviert sind. Prinzipiell können aber ohne globale Feature ähnliche Ergebnisse erwartet werden. Zudem wurden bei allen Durchläufen eine 5×5-Grid mit maximal 60 Bildmerkmalen pro Zelle und 3000 Merkmalen pro Bild benutzt, welche jedoch nur die Verteilung und nicht die Eigenschaften der jeweiligen Landmarken beeinflusst.

In Abbildung 6.3 ist gut zu erkennen, dass sich der Translationsmedian mit sinkender Tiefenunsicherheit, verglichen mit den umliegenden Werten, tatsächlich weiter verbessert. Sein Minimum liegt bei *depthnoise* 0.253125 und *pixelnoise* 2.5 mit $0.005348m$.
Leicht höhere *depthnoise*- und *pixelnoise*-Werte verbessern den Rotationsmedian. Das Minimum hier liegt bei *depthnoise* 0.406250 und *pixelnoise* 3 mit $0.340113°$.

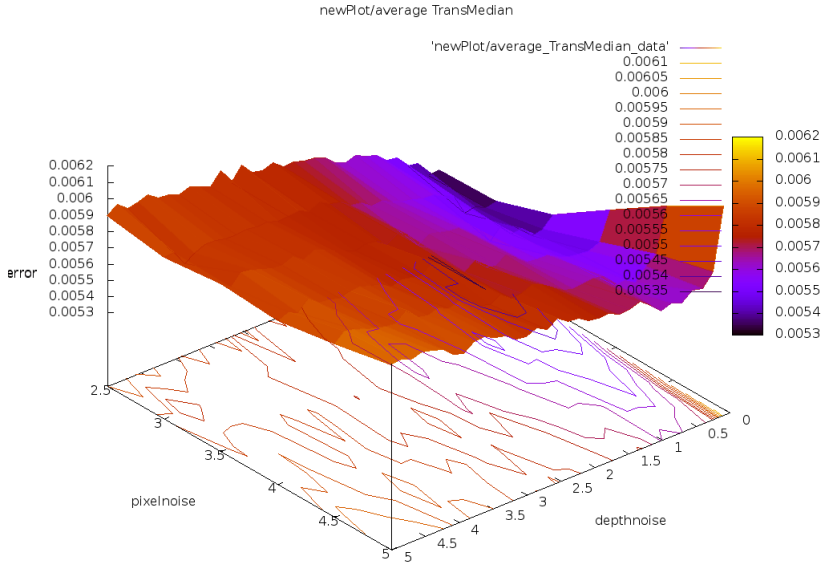

Fig. 6.3: Durchschnittlicher Translationsmedian für verschiedene Pixel- und Tiefenunsicherheiten

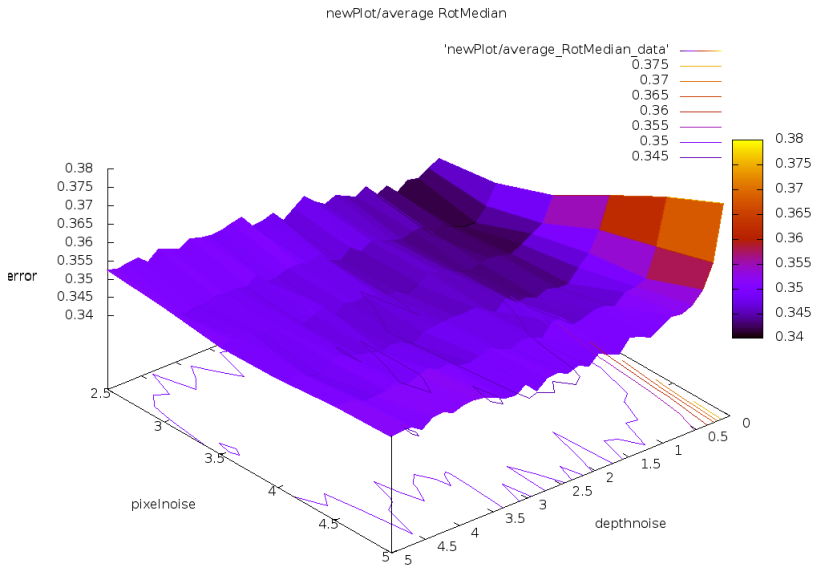

Fig. 6.4: Durchschnittlicher Rotationsmedian für verschiedene Pixel- und Tiefenunsicherheiten

6.5 Kombination mit Surfelmatching

Die Software, auf der das Bundle Adjustment-Verfahren aufbaut, besitzt bereits die Möglichkeit mit Surfeln zu arbeiten. Neben dem Surfelverfahren („Surfel“), welches sich mit diesem Ansatz kombinieren lässt, gibt es eine weitere reine Farb-Surfel-Implementierung („CSurfel“).
In diesem Teil der Evaluation wird die Kombination des Bundle Adjustment-Verfahrens mit dem kombinierbaren Surfelverfahren ausgewertet. Dafür werden die Verfahren einzeln ausgewertet und die Ergebnisse mit denen der Kombination verglichen.

Das Feature-Verfahren benutzt, sowohl bei der Einzelauswertung als auch in Kombination mit dem Surfelverfahren, keine globalen Feature, eine 3×3-Grid mit maximal 60 Bildmerkmalen pro Zelle und 2000 Merkmalen im gesamten Bild. Zudem werden die Parameter *depthnoise* 0.406250 und *pixelnoise* 3, welche aus dem Experiment zu Pixel- und Tiefenunsicherheit in Abschnitt 6.4 hervorgehen, verwendet.

Die Tabelle auf Seite 27 zeigt jeweils für Feature, Feature kombiniert mit Surfel, Surfel und CSurfel den relativen Positions- und Winkelfehler sowie die Laufzeit, die zum Verarbeiten von einem Frame benötigt wurde. Positions- und Winkelfehler sind in der Syntax *Median (Maximum)* angegeben und werden in Metern bzw. Grad gemessen. Die Laufzeit wird in der Form *Durchschnitt (Standardabweichung)* in Millisekunden angegeben.
Die besten Ergebnisse für Positions- und Winkelfehler sind **fett** markiert.

In der Tabelle ist gut zu erkennen, dass die Kombination der Feature und Surfel meistens bessere Mediane liefert, als beide Varianten im Einzelnen. Das Maximum verbessert sich durch die Kombination hingegen selten. Leider vergrößert sich hier die Laufzeit bei allen Datensätzen. Obwohl dies eine zunächst logische Konsequenz der Kombination beider Verfahren zu sein scheint, wäre es möglich gewesen, dass das Verfahren schneller konvergiert und dadurch weniger Iteration zur Transformationsschätzung nötig gewesen wären.
Auf Distanzen bis zu 3 Meter, wie es in den hier verwendeten „fr1“-Datensätzen der Fall ist, liefert CSurfel in den meisten Fällen die besten Translationsmediane. Bei RGB-D-Sequenzen mit mehr Tiefe, wie in „fr2“, bietet die Kombination der Feature und Surfel eindeutig bessere Mediane. Zudem ist der Rotationsmedian bei allen Datensätzen besser als bei CSurfel.

Datensatz	Feature		Feature & Surfel		Surfel		CSurfel	
	Position [m]	Winkel [°]	Position [m]	Winkel [°]	Position [m]	Winkel [°]	Position [m]	Winkel [°]
fr1 360	0.006427 (0.679051)	0.351089 (9.457757)	**0.005003** (0.083816)	**0.323030** (4.040763)	0.005717 (**0.045620**)	0.405765 (2.880246)	0.005433 (0.050544)	0.387846 (**2.108766**)
Laufzeit [ms]	126.70 (60.775)		97.353 (36.307)		81.537 (36.280)		79.605 (26.547)	
fr1 desk	0.006882 (1.032322)	0.389077 (36.987967)	0.006044 (0.038574)	**0.354493** (**3.678767**)	0.005927 (0.033395)	0.567947 (4.295610)	**0.004302** (**0.019885**)	0.537164 (3.679483)
Laufzeit [ms]	113.51 (18.169)		148.12 (21.006)		129.59 (50.200)		107.77 (20.577)	
fr1 desk2	0.006938 (0.380948)	0.401707 (9.807308)	0.006197 (**0.073158**)	**0.379401** (5.201032)	0.005159 (0.079626)	0.557250 (3.224869)	**0.004453** (0.023636)	0.545024 (**2.458795**)
Laufzeit [ms]	105.03 (22.026)		144.65 (28.108)		129.24 (52.591)		108.55 (24.105)	
fr1 floor	**0.001949** (0.412351)	**0.179413** (11.969039)	0.001953 (0.427718)	0.184144 (7.151052)	0.005847 (0.428143)	0.249349 (7.549752)	0.004754 (**0.387321**)	0.247640 (**5.156002**)
Laufzeit [ms]	100.12 (25.836)		158.16 (27.072)		141.46 (58.884)		122.53 (24.940)	
fr1 plant	0.003671 (0.031264)	0.280199 (3.215822)	**0.003157** (**0.020930**)	**0.277375** (3.039461)	0.003590 (0.024986)	0.432141 (**2.627921**)	0.003670 (0.030043)	0.429209 (2.725320)
Laufzeit [ms]	113.88 (16.038)		132.46 (16.518)		86.773 (15.864)		77.770 (16.227)	
fr1 room	0.004539 (0.596415)	0.315721 (21.898505)	0.003823 (**0.033639**)	**0.302048** (2.826357)	0.003894 (0.042368)	0.388295 (2.464060)	**0.003528** (0.035133)	0.376326 (**2.440270**)
Laufzeit [ms]	102.55 (21.892)		132.06 (31.961)		98.822 (36.298)		98.856 (24.500)	
fr1 rpy	0.003842 (0.075421)	**0.399053** (8.146528)	0.003262 (0.034607)	0.405801 (3.509572)	0.004141 (0.032689)	0.535046 (**2.216729**)	**0.003004** (**0.026140**)	0.494022 (2.217534)
Laufzeit [ms]	108.52 (17.476)		148.01 (20.380)		127.22 (55.346)		101.87 (23.811)	
fr1 teddy	0.005951 (0.724253)	0.349744 (19.800165)	0.004598 (0.072549)	**0.341853** (6.049857)	0.004788 (0.098804)	0.561372 (6.049877)	**0.004361** (**0.072400**)	0.539664 (**5.504226**)
Laufzeit [ms]	106.56 (23.750)		126.84 (27.627)		81.065 (22.549)		73.188 (15.919)	
fr1 xyz	0.004009 (0.019726)	0.255405 (**1.237682**)	0.003306 (0.018843)	**0.247438** (1.298979)	0.003332 (0.010755)	0.357526 (1.697445)	**0.002566** (**0.010026**)	0.340252 (1.796502)
Laufzeit [ms]	117.16 (11.447)		151.36 (14.259)		119.23 (30.399)		105.11 (16.247)	
fr2 desk	0.002281 (0.017694)	0.192932 (**1.372766**)	**0.001989** (0.019493)	**0.188116** (1.386590)	0.002171 (0.016313)	0.234313 (1.423136)	0.002125 (**0.015623**)	0.232470 (1.407245)
Laufzeit [ms]	113.72 (10.663)		127.90 (10.917)		78.746 (12.707)		78.870 (14.193)	
fr2 large w/ loop	0.009300 (**0.168577**)	0.185028 (4.785967)	**0.008687** (0.188965)	**0.180985** (**3.986622**)	0.043852 (1.615335)	0.793556 (28.284981)	0.027987 (5.268780)	0.642173 (135.227858)
Laufzeit [ms]	85.664 (19.730)		91.939 (21.444)		48.547 (7.6345)		37.647 (11.007)	
fr2 rpy	0.001316 (**0.010692**)	0.099960 (**0.988834**)	**0.001030** (0.017192)	**0.092045** (0.989181)	0.001426 (0.030466)	0.133797 (1.272965)	0.001528 (0.030444)	0.134616 (1.257548)
Laufzeit [ms]	109.25 (17.415)		123.49 (17.964)		73.633 (11.465)		68.650 (15.443)	
fr2 xyz	0.001409 (**0.006309**)	0.143418 (1.246366)	**0.001198** (0.006691)	**0.140634** (**1.224105**)	0.001506 (0.059085)	0.178640 (1.918257)	0.001426 (0.045231)	0.183441 (1.567041)
Laufzeit [ms]	118.05 (10.843)		136.61 (16.596)		81.368 (13.486)		77.384 (18.624)	

6.6 Skip Frames

Das Evaluationstool erlaubt es, k Frames zu überspringen, sodass Kameratransformationen nurnoch alle $k + 1$ Frames berechnet werden. Dadurch können die Auswirkungen größerer Transformationen auf die Trackinggenauigkeit beobachtet und Datensätze schneller ausgewertet werden, zum Beispiel um vorläufig neue Funktionen des Programms zu testen.
Ebenfalls interessant ist es, die Trackinggenauigkeit in Abhängigkeit von unterschiedlichen Tiefeninformationen zu untersuchen. Für dieses Experiment werden deshalb 3 möglichst unterschiedliche Datensätze verwendet:

- Freiburg1 Desk in einer Büroumgebung mit folglich hauptsächlich nahen Objekten.
- Freiburg2 Desk in einer Halle mit einem kleinen Büronachbau, bestehend aus 2 Tischen, Stühle, Monitor und sonstigen Accessoires. Die Kamera bewegt sich einmal um die Tischen herum und schließt die Schleife zur Ausgangsposition. Der Datensatz bietet hauptsächlich nahe und teilweise ferne Objekte.
- Freiburg2 Large With Loop in einer Halle und somit hauptsächlich fernen Objekten. Gegen Ende schließt sich die Schleife zur Szenerie vom Anfang der RGB-D-Sequenz.

Jedoch lässt sich ein Datensatz nicht für jeden beliebigen Skip-Wert auswerten. Je größer der Wert, desto höher die Wahrscheinlichkeit, dass die beiden betrachteten Frames ihren Bezug zueinander verlieren, nicht genug überlappen und damit eine Transformationsschätzung unmöglich ist. Der größtmögliche Skip-Wert hängt daher vom jeweiligen Datensatz ab.

Bei allen Durchläufen wurde keine Grid benutzt. Die Unsicherheitsparameter basieren an dieser Stelle noch auf den diskreten Daten des Pixel- und Tiefenunsicherheitsexperiments und wurden auf *depthnoise* 2 und *pixelnoise* 2.5 gesetzt.
Für die Auswertung wurden 4 verschiedene Optionen benutzt: Nicht-globale Feature („F nonGlobal“), globale Feature und Surfel in Kombination („FS Global“), nicht-globale Feature und Surfel („FS nonGlobal“) sowie nur Surfel („S“).

Dargestellt in den Bildern 6.5, 6.6 und 6.7 ist der Translationsmedian in Metern für den jeweiligen Skipwert.

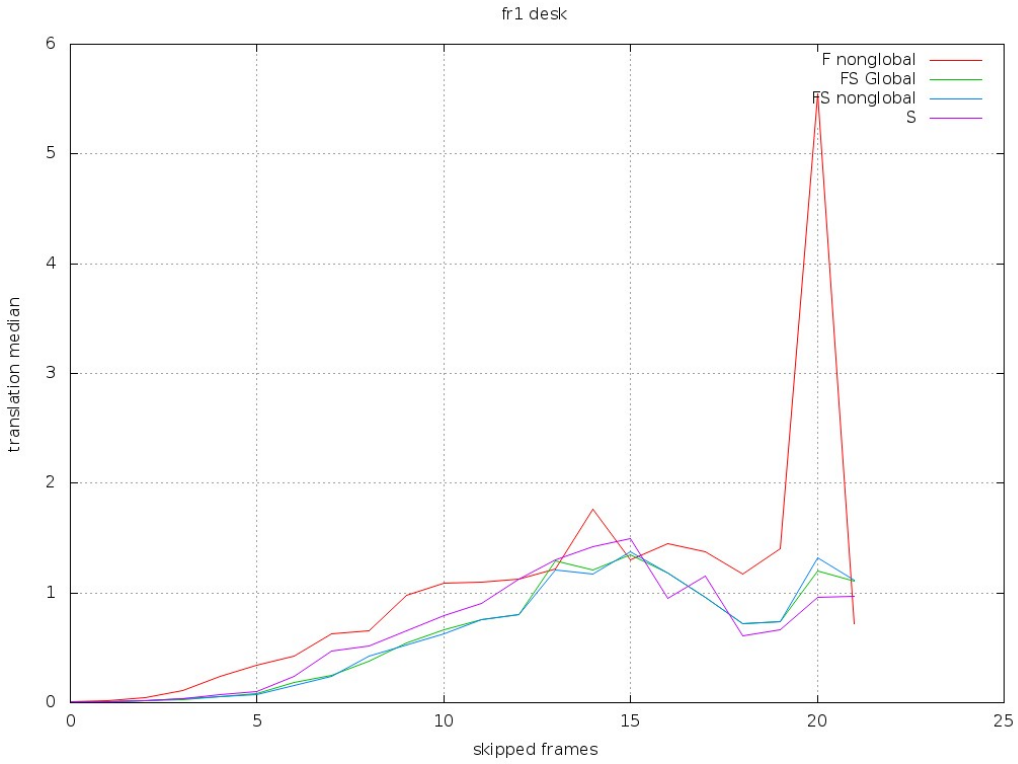

Fig. 6.5: Freiburg 1 Desk

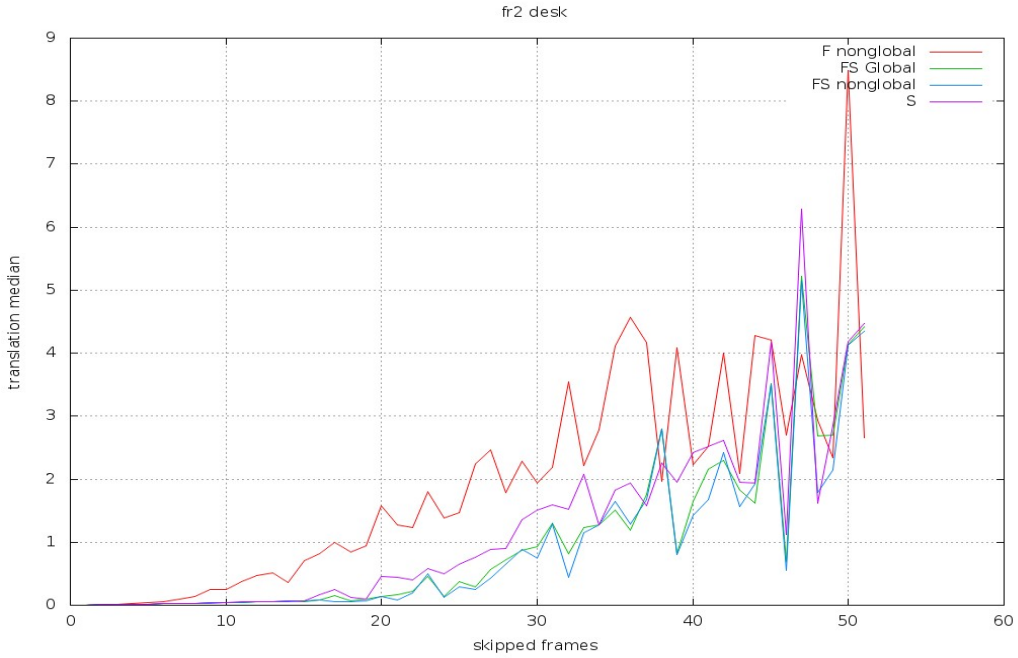

Fig. 6.6: Freiburg 2 Desk

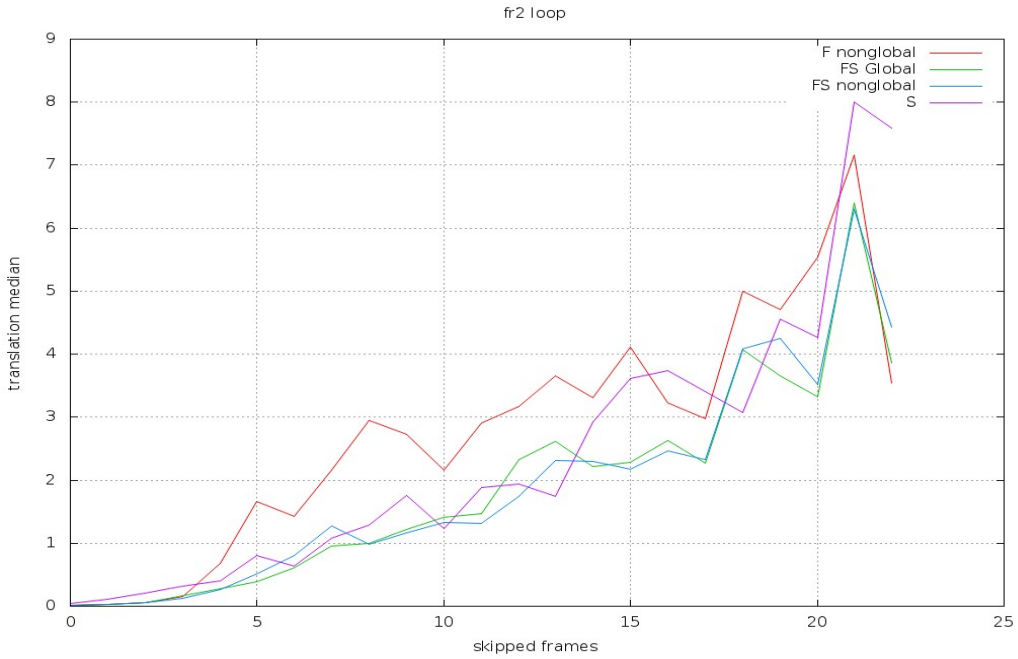

Fig. 6.7: Freiburg 2 Large with Loop

Auffällig ist, dass der Fehler bei allen Optionen und Datensätzen für „große“ Skipwerte unbrauchbar groß wird. Freiburg2 Desk weißt jedoch viel bessere Fehlerwerte und höhere Skipwerte auf, was dadurch zu erklären ist, dass die Kamera hier keine unruhigen Bewegungen macht und über den gesamten Datensatz die selbe Szenerie aus verschiedenen Winkeln gefilmt wurde, sodass die Überlappung zweier Frames mit großem Abstand ausreicht, um eine relativ gute Transformation zu schätzen, während Freiburg1 Desk und Freiburg2 Large With Loop stets nur Teile der Szenerie zeigen.
Da man mit großen Skipwerten scheinbar nicht arbeiten kann wäre ein Blick auf die kleineren Skipwerte interessant, da hier der Fehler noch sehr klein zu sein scheint. Zudem könnte sich ein Blick darauf lohnen, wie sich die Verhältnisse der Fehler zueinander verhalten, also ob sie sich um einen vergleichbaren Faktor erhöhen.
Die folgenden Graphiken konzentrieren sich auf die kleineren Skip-Werte. Die einzelnen Funktionen wurden durch den jeweiligen Fehler bei *Skip* 0 geteilt.

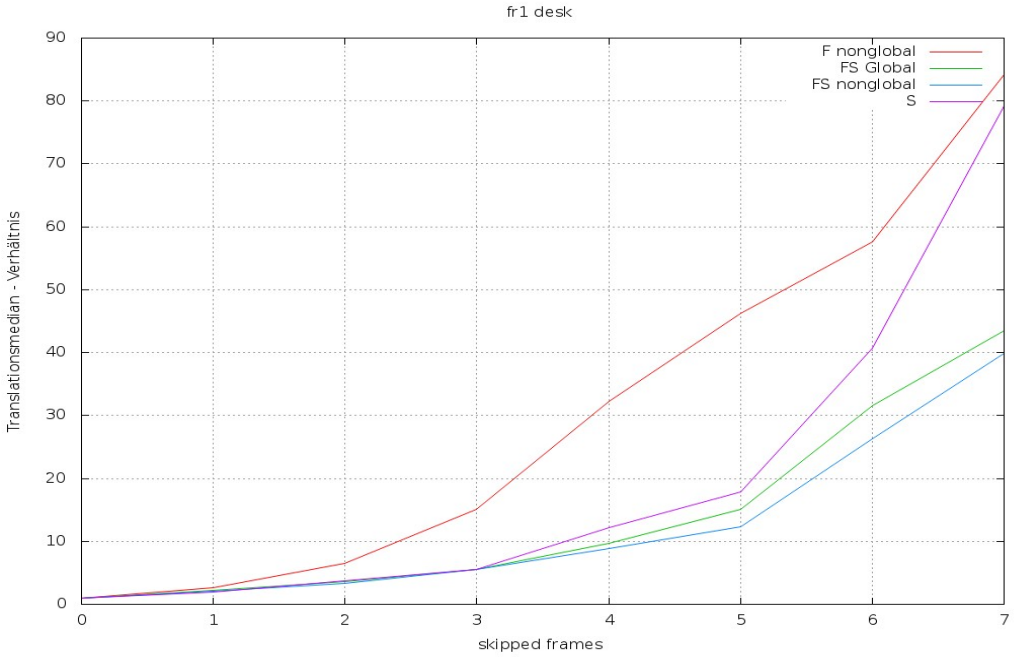

Fig. 6.8: Skipframes Verhältnis Freiburg 1 Desk

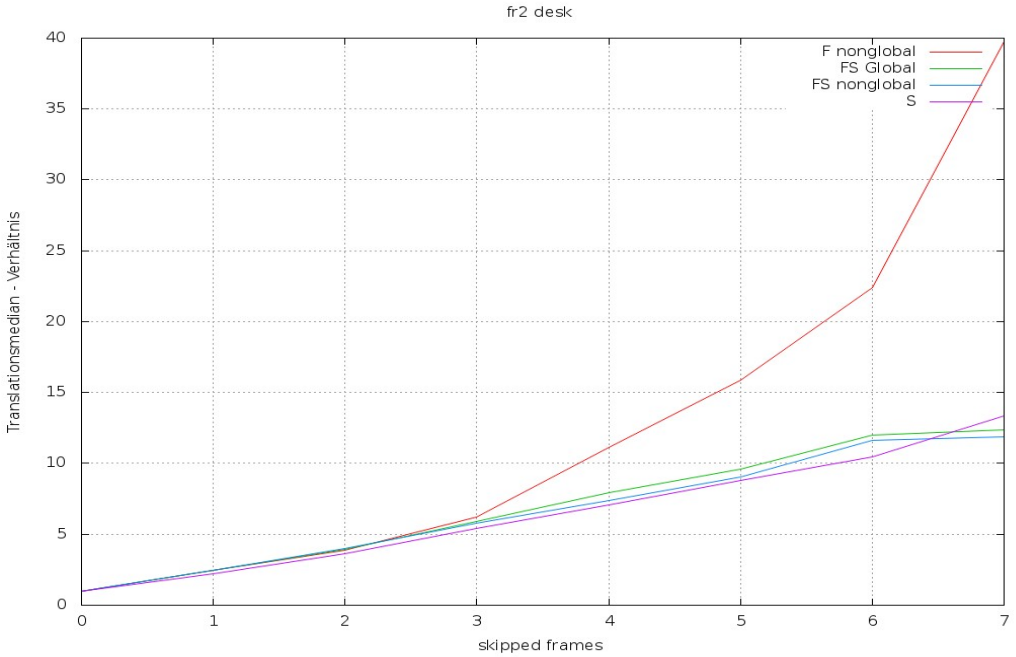

Fig. 6.9: Skipframes Verhältnis Freiburg 2 Desk

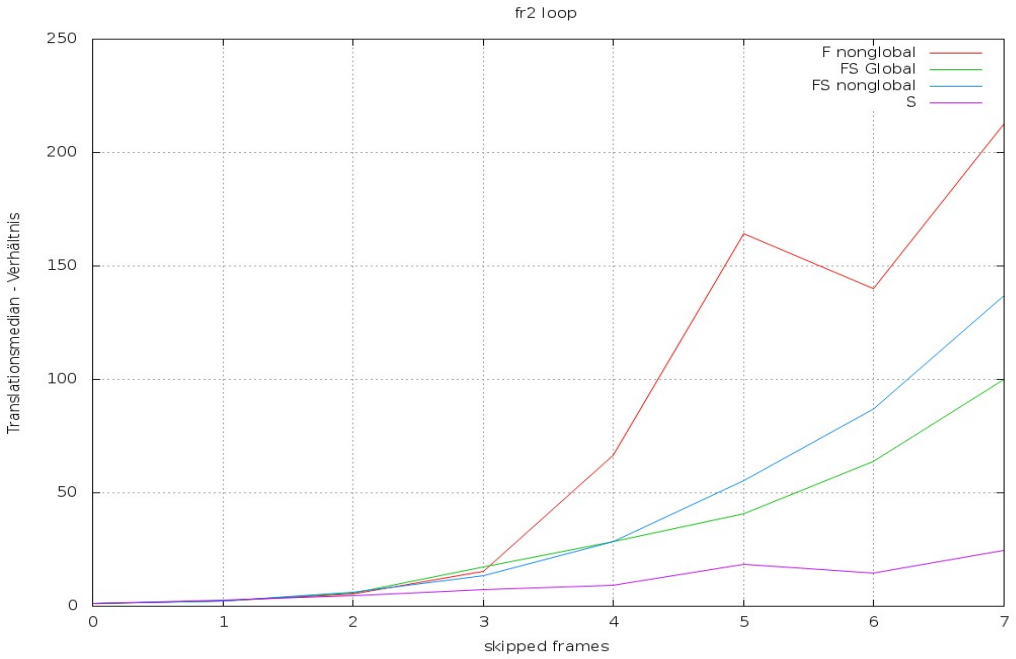

Fig. 6.10: Skipframes Verhältnis Freiburg 2 Large with Loop

Die meisten Fehlerfaktoren verschlechtern sich anfangs ähnlich und annähernd linear und driften dann auseinander. Ab welchem *Skip*-Wert die Faktoren abdriften hängt dabei vom Datensatz ab. Während in Abbildung 6.9 Varianten, die Surfelmatching verwenden, bis *Skip* 6 relativ konstant ansteigen, verliert sich dieses Verhalten bei den anderen Datensätzen durch unruhigere Bewegungen bzw. mehr Tiefe viel früher.
Dass gute Tiefeninformationen eine dennoch wichtige Rolle spielen erkennt man gut an Abbildung 6.10 und 6.7, da hier der Fehler wesentlich schneller ansteigt und schlechter ausfällt.
Ebenfalls auffällig ist, dass sich überall das Feature-Verfahren als erstes signifikant verschlechtert, was zeigt, dass es für größere Transformationen nicht gut geeignet ist.

6.7 Globale Features

In diesem Abschnitt wird das Trackingverhalten mit und ohne globale Features miteinander verglichen. Das Surfelmatching wird hierfür deaktiviert, um stabilisierende Effekte auszuschließen. Eine Grid wird ebenfalls nicht verwendet. Pro Bild werden 1000 ORB-Feature detektiert. Für Pixel- und Tiefenunsicherheit werden die Standardwerte 5 bzw. 4 benutzt. Tabelle 6.1 zeigt die den Median und dahinter in Klammern den maximalen Fehler für Translation in der ersten und Rotation in der zweiten Zeile.
Die Technik trägt leider nicht zur Verbesserung der Trackinggenauigkeit bei. Nahezu alle Ergebnisse verschlechtern sich durch die Weiterverwendung von bereits optimierten Landmarkenpositionen aus vorigen Iterationen.

Datensatz	global	nicht Global
fr1 360	0.007558m (0.634279m) 0.379946° (21.039278°)	0.006939m (0.346633m) 0.365399° (11.858897°)
fr1 desk	0.007946m (0.106191m) 0.430801° (7.065966°)	0.007727m (0.050686m) 0.417614° (4.034406°)
fr1 desk2	0.008121m (0.377135m) 0.436458° (15.729105°)	0.007669m (0.275151m) 0.417066° (15.201401°)
fr1 floor	0.002298m (0.412351m) 0.192390° (16.727707°)	0.002108m (0.412351m) 0.186324° (21.039443°)
fr1 plant	0.004862m (0.038231m) 0.293290° (3.233067°)	0.004444m (0.034032m) 0.287766° (3.209152°)
fr1 room	0.005679m (0.190900m) 0.348068° (11.415824°)	0.005237m (0.037138m) 0.332792° (3.449693°)
fr1 rpy	0.004516m (0.753639m) 0.407847° (55.312507°)	0.004194m (0.150703m) 0.400637° (15.576924°)
fr1 teddy	0.007673m (0.784026m) 0.359791° (14.553936°)	0.007122m (0.784027m) 0.354766° (14.553854°)
fr1 xyz	0.004654m (0.028549m) 0.273410° (1.635657°)	0.004456m (0.030049m) 0.272074° (1.699718°)
fr2 desk	0.003047m (0.022794m) 0.206056° (1.320892°)	0.002656m (0.019981m) 0.199240° (1.354480°)
fr2 loop	0.014455m (0.371625m) 0.215766° (1.606659°)	0.013547m (0.373115m) 0.210318° (1.676148°)
fr2 rpy	0.001867m (0.045035m) 0.130016° (2.041402°)	0.001634m (0.038336m) 0.118852° (1.752327°)
fr2 xyz	0.001813m (0.020318m) 0.160554° (1.236395°)	0.001624m (0.017240m) 0.153624° (1.244329°)

Tab. 6.1: RPE für Translation und Rotation. Syntax: Median (Maximum)

7. ZUSAMMMENFASSUNG

Ziel dieser Bachelorarbeit war es, ein Verfahren zu implementieren und auszuwerten, welches über spärliches Bildpunkt- und dichtes Tiefenmatching paarweise RGB-D-Bilder einer Sequenz aufeinander registriert und inkrementell die Transformation zwischen den Position, von denen die betrachteten Bilder aufgenommen wurden, schätzt und verbessert. Dafür sollten in einem Bundle Adjustment-Verfahren die geschätzen Landmarkenpositionen schrittweise nachjustiert werden.

Als Grundlage für die Implementierung wurde die Software der „Multiresolution surfel maps“ [7] verwendet, welche bereits in der Lage war, RGB-D-Datensätze einzulesen, schrittweise zu verarbeiten und die rekonstruierte Trajektorie auszulagern. Ebenso beinhaltete die Software die Implementierung von Surfelmatching.

Mit Hilfe verschiedenen Quellen wurde über die Minimierung einer Fehlerfunktion ein Bundle Adjustment-Verfahren konstruiert, welches im Anschluss über Techniken, wie sie g^2o verwendet, zu einer praktikablen Lösung optimiert wurde. Der Implementierung wurden weitere optionale Funktionen hinzugefügt, um womöglich bessere Ergebnisse zu erzielen. Ein Raster beliebiger Größe dient dem Zweck, die detektierten Bildmerkmale gleichmäßiger zu verteilen und eine hohe Dichte von Merkmalen in den Bildregionen zu verringern, während gleichzeitig die Gesamtzahl der zu detektierenden Merkmale erhöht wird, um verworfene Merkmale auszugleichen. Zudem wurde die Möglichkeit implementiert, bereits optimierte Landmarkenpositionen in der nachfolgenden Registrierung zweier Bilder einer Sequenz weiterzuverwenden. Des weiteren wurde die Kombination von Feature- und des Surfelverfahrens sowie bisher fixe Pixel- und Tiefenunsicherheiten parametrisiert.

Um das Verfahren zu evaluieren wurde frei zugängliche RGB-D-Datensätze verwendet, die als gemeinsame Grundlage zum Vergleich verschiedener Verfahren dienen. Die Auswertung der Experimente hat ergeben, dass das reine Bundle Adjustment-Verfahren bei unruhigen Bewegungen der Kamera

und größeren Transformationen schlechtere Ergebnisse liefert. Die Unsicherheiten für Pixel- und Tiefeninformationen können jedoch viel kleiner als angenommen angesetzt werden. Vor allem die Tiefenunsicherheit fällt sehr viel geringer aus. Zudem hat sich die Verwendung von Grids als nützlich erwiesen. Zu große Grids und damit starke erzwungene Verteilungen von Bildmerkmalen wirken sich hingegen wieder negativ aus. Der ursprüngliche Gedanke, durch die Kombination von Bildpunkt- und Tiefenmatching bessere Ergebnisse zu erhalten, hat sich als richtig herausgestellt.

7.1 Ausblick

Für zukünftige Grid-Implementierungen wäre eine effizientere Umsetzung interessant. Statt im gesamten Bild sehr viele Merkmale zu suchen, einzuteilen, zu sortieren und überschüssige rauszufiltern wäre es effizienter direkt auf den jeweiligen Bereichen, die den Gridzellen entsprechen, nach Merkmalen zu suchen.

Im 2. Teil des Pixel- und Tiefenunsicherheitsexperiment lag das Minimum nur knapp im Rahmen der beiden Intervalle. Gegebenenfalls lässt sich ein besseres Minimum in direkter Umgebung finden, wenn die unteren Grenzen der Unsicherheitswerte weiter gesenkt werden. Zudem könnte es sich als nützlich erweisen mit dynamischen Unsicherheiten zu arbeiten. Diese könnten sich beispielsweise an der Bewegung der Landmarke relativ zur Kamera orientieren.

Zu guter Letzt könnte an verschiedenen Gewichtungen der Optimierung gearbeitet werden. Durch Gewichtung der Landmarkenupdates und Gewichtung der Feature- bzw. Surfelterme könnte das Verfahren schneller konvergieren und so wertvolle Zeit einsparen.

LITERATURVERZEICHNIS

[1] K. Madsen, H. B. Nielsen, and O. Tingleff. *Methods for Non-Linear Least Squares Problems (2nd ed.)*. Informatics and Mathematical Modelling, Technical University of Denmark, DTU, Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby, 2004.

[2] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard. An evaluation of the RGB-D SLAM system. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, St. Paul, MA, USA, May 2012.

[3] Henri Gavin. The levenberg-marquardt method for nonlinear least squares curve-fitting problems. Sep. 2011.

[4] Peter Henry, Michael Krainin, Evan Herbst, Xiaofeng Ren, and Dieter Fox. Rgbd mapping: Using depth cameras for dense 3d modeling of indoor environments. In *In RGB-D: Advanced Reasoning with Depth Cameras Workshop in conjunction with RSS*, 2010.

[5] Albert S. Huang, Abraham Bachrach, Peter Henry, Michael Krainin, Daniel Maturana, Dieter Fox, and Nicholas Roy. Visual odometry and mapping for autonomous flight using an rgb-d camera. In *Int. Symposium on Robotics Research (ISRR)*, Flagstaff, Arizona, USA, Aug. 2011.

[6] Richard A. Newcombe, Andrew J. Davison, Shahram Izadi, Pushmeet Kohli, Otmar Hilliges, Jamie Shotton, David Molyneaux, Steve Hodges, David Kim, and Andrew Fitzgibbon. KinectFusion: Real-time dense surface mapping and tracking. In *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, pages 127–136. IEEE, October 2011. https://www.youtube.com/watch?v=quGhaggn3cQ.

[7] Jörg Stückler and Sven Behnke. Multi-resolution surfel maps for efficient dense 3D modeling and tracking. *Journal of Visual Communication and Image Representation*, 2013.

[8] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012. http://vision.in.tum.de/data/datasets/rgbd-dataset/download.

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