Handgeführtes 3D Scanning auf mobilen Plattformen

Masterarbeit

im Fach Informatik

vorgelegt von

Dmitri Presnov

Geboren am 24. April 1976 in Tartu

Angefertigt am

Lehrstuhl für Computergraphik und Multimediasysteme
Naturwissenschaftlich-Technische Fakultät
Universität Siegen

Betreuer:

Prof. Dr. A. Kolb, Lehrstuhl Computergraphik und Multimediasysteme, Universität Siegen

Dr. M. Lambers, Lehrstuhl Computergraphik und Multimediasysteme, Universität Siegen

Beginn der Arbeit: 22. August 2016

Abgabe der Arbeit: 05. Januar 2017

Eidesstattliche Erklärung

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

Siegen, den 31. Dezember 2016

Zusammenfassung

Die vorliegende Masterarbeit beschäftigt sich mit dem Thema Kamera-Tracking im Kontext von 3D Rekonstruktion auf mobilen Geräten. Im Rahmen der Arbeit wird ein punktbasiertes Rekonstruktionsframework, das ursprünglich für Desktop-Umgebungen entwickelt wurde, auf ein mobiles System, bestehend aus einem Tango Tablet und einer externen PMD pico flexx Tiefenkamera, portiert. Das Tracking-Konzept des Frameworks, welches eine ICP-basierte Kameraposenschätzung anhand von 3D Daten vorsieht, wird erweitert, indem zusätzliche, für ein Tablet übliche Bewegungserfassungsquellen, wie Farbkamera und IMU, mit einbezogen werden. Beim hier vorgeschlagenen Multi-Sensor-Tracking fließen Daten der letztgenannten Sensoren in ein als Blackbox benutztes VIO-Verfahren ein; anschließend wird die damit berechnete Pose zur Initialisierung von ICP in der Rekonstruktionspipeline eingesetzt.

Der experimentelle Teil der Arbeit schließt Versuche mit kontrollierter Kamerabewegung ein, bei denen eine Szene aus mittelgroßen Objekten eingescannt wird. Die Genauigkeit des Trackings wird anhand der Abweichungen von der zu erwartenden Kameralaufbahn evaluiert. Die Analyse der Ergebnisse fokussiert sich hauptsächlich auf zwei Aspekte. Zum einen soll festgestellt werden, ob das gegebene Framework auch unter begrenzten Ressourcen einer mobilen Plattform einsetzbar ist, und insbesondere wie sich verschiedene Parameter auf das Tracking auswirken. Zum anderen gilt zu verifizieren, ob eine Beschleunigung der ICP-Konvergenz und eine Erhöhung der Posengenauigkeit, die durch das Einführen des Multi-Sensor-Trackings angestrebt werden, tatsächlich erreichbar sind.

Abstract

This master's thesis deals with the camera tracking in the context of 3D reconstruction on mobile devices. In the preliminary step a point-based reconstruction framework will be ported from a desktop environment to a mobile system consisting of a Tango Tablet and an external Pmd pico flexx range camera. The camera tracking in the framework provides a pose estimation with the ICP algorithm by means of 3D data; the thesis extendes the original concept by including additional motion capture sensors, usually available on a tablet, like color camera and IMU. In the proposed multi-sensor tracking a visual-inertial odometry method, used as a black box, processes data from the mentioned sensors and provides a first pose estimation that will be applied for the initialization of ICP in the reconstruction pipeline.

During the experiments a medium scale scene will be scanned with the motion controlled device. The successive evaluation of the tracking precision is based on the deviation from the expected camera trajectory. The result analysis principally focuses on two aspects. On the one hand, the goal is to ascertain either the original framework implementation allows a 3D reconstruction under restricted resources of a mobile system. On the other hand, it should verify either the objectives of the multi-sensor tracking to provoke a speed-up of the ICP convergence and to increase the pose precision have been achieved.

Inhaltsverzeichnis

Verzeichnis der Bilder **ii**

Verzeichnis der Tabellen **iii**

Abkürzungsverzeichnis **iv**

1 Einleitung **1**

2 Theoretische Grundlagen **3**

2.1 3D Rekonstruktion . . . 3

2.1.1 Bildaufnahme/Vorverarbeitung . . . 4

2.1.2 Schätzung der Kamerapose / Bildregistrierung . . . 5

2.1.3 Aktualisierung des Modells . . . 9

2.1.4 Rendering . . . 9

2.2 Bewegungs-Tracking mit weiteren Sensoren . . . 10

2.2.1 Inertiales Tracking . . . 10

2.2.2 Visuell-inertiale Odometrie . . . 12

3 Bekannte Verfahren **13**

3.1 Punktbasiertes Rekonstruktionsframework . . . 13

3.2 Kombination aus (visuell-)inertialem Tracking und ICP . . . 15

4 Konzeption **17**

4.1 Systementwurf . . . 17

4.1.1 Komponenten . . . 17

4.1.2 Einschränkungen . . . 18

4.1.3 Notwendige Änderungen . . . 19

4.2 Evaluierungsziele . . . 21

4.2.1 Genauigkeit des ICP-basierten Kamera-Trackings . . . 22

4.2.2 Vergleich Tracking mit ICP vs. Multi-Sensor-Tracking . . . 22

4.2.3 Performance . . . 23

4.3 Evaluierungsmethoden . . . 23
4.3.1 Tracking-Genauigkeit . . . 23
4.3.1.1 Erstellung einer Referenz . . . 24
4.3.1.2 Fehlermetriken . . . 24
4.3.2 Einfluss der Initialisierung von ICP mit VIO-Daten auf seine Konvergenz . . . 25

5 Versuchsaufbau und -durchführung 26
5.1 Hardware . . . 26
5.2 Software . . . 27
5.3 Versuchsaufbau . . . 28

6 Ergebnisse 30
6.1 Genauigkeit des ICP-basierten Kamera-Trackings . . . 30
6.1.1 Basiskonfiguration . . . 30
6.1.2 Weitere Konfigurationen . . . 33
6.2 Tracking mit ICP vs. Multi-Sensor-Tracking . . . 34
6.2.1 Versuche mit erhöhter Geschwindigkeit . . . 40
6.3 Performance . . . 42
6.4 Weiteres . . . 43
6.4.1 Konfidenz-Schwellenwert für stabile Punkte . . . 43
6.4.2 Versuch mit anderen Objekten . . . 45

7 Zusammenfassung und Ausblick 48

Literaturverzeichnis 50

Verzeichnis der Bilder

2.1 3D Rekonstruktionspipeline 4
2.2 Bewegung-Tracking mit Drehraten- und Beschleunigungssensor 11

5.1 Versuchsaufbau: Szene 28
5.2 Versuchsaufbau: Maße 29

6.1 ICP Basis1-3: Kameralaufbahn 31
6.2 Orientierungsfehler über Zeit 32
6.3 Fehler beim Starten der Kamerabewegung 33
6.4 ICP Zusätzliche Parameter: Kameralaufbahn 35
6.5 ICP vs. VIO: Gefittete Kreislaufbahnen 36
6.6 VIO und VIO+ICP 38
6.7 VIO und VIO+ICP: Ansicht 2 39
6.8 Versetzen der Laufbahn bezüglich VIO 3 39
6.9 Rekonstruktionsergebnisse 41
6.10 Rekonstruktion mit unterschiedlichen Schwellenwerten 44
6.11 Extra-Versuch: Lego-Szene 45
6.12 Initialer Kamera-Drift bei unterschiedlichen Szenen 47

Verzeichnis der Tabellen

4.1 Verzeichnungskoeffizienten 21

6.1 ICP Tracking 34
6.2 Multi-Sensor-Tracking 36
6.3 ICP Konvergenz 37
6.4 Tracking bei höheren Geschwindigkeiten 41
6.5 Frameraten 42
6.6 ICP Tracking: Lego-Figuren 45

Abkürzungsverzeichnis

ICP	der **I**terative **C**losest **P**oint (Algorithmus)
VIO	**V**isuell-**i**nertiale **O**dometrie, *engl.* Visual Inertial Odometry
EKF	**E**rweitertes **K**alman-**F**ilter, *engl.* Extended Kalman Filter
TOF	**T**ime-**O**f-**F**light(-Kamera)

Kapitel 1

Einleitung

Die Erstellung von dreidimensionalen Modellen aus Tiefendaten ist eine rechenintensive Aufgabe, die ursprünglich nur mit Desktop-CPUs und -GPUs zu meistern war. Dank der Erscheinung leistungsfähigerer Smartphones und Tablets sowie mobil einsetzbarer Tiefenkameras hat das Thema *3D Rekonstruktion auf mobilen Geräten* in den letzten Jahren großes Interesse geweckt, da dies natürliche Einschränkungen von stationären Systemen überwinden und flexiblere Anwendungsmöglichkeiten öffnen würde. Dennoch ermöglichen die Gegebenheiten eines mobilen Systems in der Regel keine direkte Portierung der bewährten Rekonstruktionsverfahren, die ursprünglich auf eine Desktop-Umgebung ausgelegt wurden. In der vorliegenden Masterarbeit wird Umsetzbarkeit des punkbasierten 3D Rekonstruktionsansatzes von Keller et al. [KLL+13] / Lefloch et al. [LKS+17] auf einem Tango-Tablet evaluiert. Dabei werden konzeptuelle Erweiterungen vorgeschlagen, die einer effizienten Adaptation an die mobile Plattform dienen sollen.

Der Fokus unserer Arbeit liegt auf der Evaluierung des Kamera-Trackings. Die Genauigkeit der Schätzung der aktuellen Kamerapose spielt eine Schlüsselrolle für die Qualität der Rekonstruktion. Der oben erwähnte punktbasierte Ansatz sieht Posenbestimmung mit Hilfe des Iterative Closest Point Algorithmus anhand von 3D Daten vor. Das ist ein rechenintensiver Vorgang, dessen Ergebnisse nicht zuletzt von der Qualität der eingehenden Tiefenbildern abhängen, so dass bei seiner Ausführung auf einer mobilen Hardware mit relativ eingeschränkten Ressourcen gewisse Einbußen an Präzision und an Anwendungsperformance zu erwarten sind. Andererseits verfügen moderne Smartphones und Tablets über eine Reihe weiterer Sensoren, wie z.B. Inertialsensoren und Farbkameras, die ebenso Bewegungserfassung ermöglichen. Diese Daten können komplementär zur Posenschätzung in der Rekonstruktionspipeline verwendet werden. Im Rahmen unserer Arbeit wird ein solches kombiniertes Tracking als Erweiterung zum ursprünglichen Verfahren implementiert. Anschließend soll eine vergleichende Evaluierung der beiden Verfahren auf der Basis experimenteller Ergebnisse durchgeführt werden; dabei gilt unter anderem festzustellen, inwiefern das Multi-Sensor-Tracking zu einer präziseren und schnelleren Kameraposenschätzung in der 3D Rekonstruktionspipeline beitragen und ggf. Einschränkungen einer mobilen Plattform kompensieren kann.

Der Rest der Arbeit ist wie folgt aufgebaut. Im Kapitel 2 werden theoretische Grundlagen der 3D

Rekonstruktion aus Tiefendaten sowie des inertialen Bewegungstrackings dargelegt. Das Kapitel 3 bietet eine Übersicht der für die vorliegende Arbeit relevanten Verfahren. Das Konzept eines mobilen Systems für 3D Rekonstruktion sowie die Beschreibung der Evaluierungsziele und -methoden befindet sich im Kapitel 4. Im anschließenden Kapitel 5 werden der genaue Aufbau und die Durchführung der Versuche beschrieben, deren Ergebnisse im Kapitel 6 dargestellt und evaluiert werden. Eine kurze Zusammenfassung im Kapitel 7 schließt die Arbeit ab.

Kapitel 2

Theoretische Grundlagen

In diesem Kapitel werden Ansätze und Techniken dargelegt, die eine theoretische Basis für die vorliegende Arbeit bilden. Im wesentlichen sollen hier zwei Themen erörtert werden, deren Relevanz für die Arbeit aus dem im Kapitel 1 formulierten Ziel hervorgeht. Der erste Teil behandelt die Grundlagen der 3D Rekonstruktion aus Tiefendaten. Eine besondere Aufmerksamkeit gilt dabei dem Kamera-Tracking, während die Darstellung anderer Aspekte weniger ausführlich ist und in erster Linie dazu dient, das Problem des Trackings in den Gesamtkontext zu setzen. Im zweiten Teil werden Möglichkeiten der Bewegungserfassung mit weiteren Sensoren betrachtet. Es handelt sich dabei insbesondere um solche Verfahren, die mit der Ausrüstung der modernen Smartphones und Tablets – wie magnetische und Inertialsensoren, Farbkamera – realisierbar sind. Da es ein umfangreiches Forschungsfeld ist, das über das eigentliche Thema dieser Masterarbeit hinausgeht [1], können hier nur allgemeine Prinzipien dargestellt werden.

2.1 3D Rekonstruktion

Der Rekonstruktionsvorgang kann als eine Pipeline mit Einteilung in einzelne Stufen aufgefasst werden. Eine mögliche Pipeline-Struktur wurde von Rusinkiewicz at al. in [RHHL02] vorgeschlagen (eine verallgemeinerte Darstellung von dieser findet man in [KZY15]). Die Autoren unterscheiden folgende Pipeline-Stufen: **Aufnahme von Tiefenbildern und Datenvorverarbeitung**, **Schätzung der Kamerapose / Bildregistrierung**, **Modellaktualisierung** und **Rendering**. Die Abbildung 2.1 zeigt den schematischen Aufbau der Pipeline. Im weiteren Verlauf dieses Abschnitts werden die einzelnen Stufen detaillierter dargestellt.

[1] Wie es im Kapitel 4 erläutert wird, wird ein solches Tracking-Verfahren in dieser Arbeit als Blackbox benutzt.

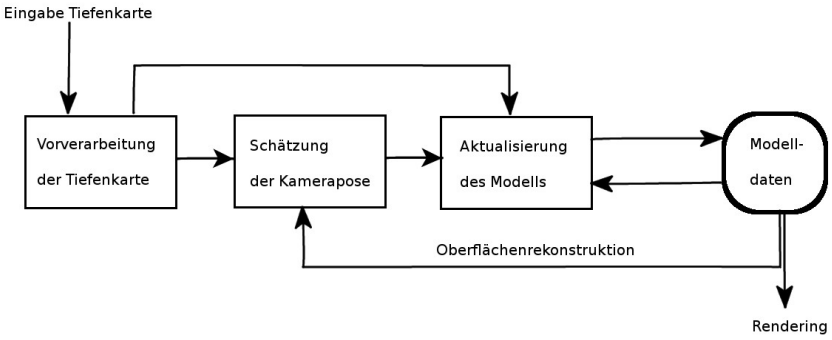


Bild 2.1: 3D Rekonstruktionspipeline

2.1.1 Bildaufnahme/Vorverarbeitung

Den Anfang der Pipeline bildet die Aufnahme der Daten mit einem Tiefensensor. Die dahinter stehenden Technologien können grundsätzlich einer der zwei folgenden Kategorien zugeordnet werden. Bei den Laufzeitmessverfahren (eng. *time-of-flight*, TOF) wird die Szene von der Kamera aktiv beleuchtet, und der Abstand zu einer Oberfläche aus der Zeit zwischen der Ausstrahlung und der Erfassung des reflektierten Lichts ermittelt. Die Abstandsberechnung in den Verfahren der zweiten Kategorie basiert auf Triangulierung. Dabei wird die Tatsache ausgenutzt, dass bei der Aufnahme eines Objekts aus verschiedenen Positionen, auch seine Darstellung im 2D Bild sich ändert. Ist die relative Positionsänderung der Kamera bekannt, kann mit Hilfe dieser Daten Tiefeninformation berechnet werden. Verbreitete Vertreter dieser Kategorie sind z.B. Streifenprojektion-Scanner (eng. *structured light scanner*). Eine Übersicht diverser Tiefensensortypen mit Erläuterung jeweiliger Stärken und Problempunkten findet man z.B. in [KP15].

Die Ausgabe einer 3D Kamera (nach der Verarbeitung der Rohdaten) liegt in Form eines Tiefenkarte $\mathcal{D}$ vor.[2] Beim Scannen wird die 3D Position eines Punktes in der reellen Welt $\mathbf{p} = (x, y, z)$ durch die Projektion in die Bildebene als Pixelkoordinaten $\mathbf{u} = (x', y')$ und Kamera-zu-Punkt-Abstand $\mathcal{D}(\mathbf{u}) = z$ kodiert. Diese Projektion kann mit der folgenden Formel ausgedrückt werden:

$$z\mathbf{u} = K\mathbf{p} \tag{2.1}$$

mit $\mathbf{u}$ in homogenen Koordinaten und intrinsischer Kamera-Matrix $K = \begin{Bmatrix} \frac{f_x}{pix_w} & 0 & x'_c & 0 \\ 0 & \frac{f_y}{pix_h} & y'_c & 0 \\ 0 & 0 & 1 & 0 \end{Bmatrix}$,

[2] Weiter werden Begriffe ‚Tiefenkarte' und ‚Tiefenbild' synonym verwendet.

wobei f_x, f_y Brennweite der Kamera in x- bzw. y-Richtung, pix_w und pix_h Breite bzw. Höhe eines Pixels, und x'_c, y'_c Koordinaten des Zentrums des Kamerasensors sind.[3] f, pix_w, pix_h, x'_c, y'_c stellen intrinsische Parameter einer Kamera dar.

Bewirkt die Kameralinse eine beachtliche Verzeichnung, ist diese bei der Projektionsberechnung zusätzlich zu den oben genannten Parametern zu berücksichtigen. Die komplette Transformation aus 3D in Pixelkoordinaten wird dann durch die folgenden Schritte beschrieben:

$$
\begin{aligned}
& x'' = x/z, \\
& y'' = y/z \\
& x^d = x''(k_1 r^2 + k_2 r^4 + k_3 r^6) + 2p_1 x'' y'' + p_2(r^2 + 2x''^2), \\
& y^d = y''(k_1 r^2 + k_2 r^4 + k_3 r^6) + p_1(r^2 + 2y''^2) + 2p_2 x'' y'' \\
& \text{mit } r^2 = x''^2 + y''^2, \\
& \mathbf{u} = K(x^d, y^d, 1)^T,
\end{aligned}
\tag{2.2}
$$

wobei k_1, k_2, k_3 Koeffizienten der radialen [4] und p_1,p_2 der tangentialen Verzeichnung sind.

Da Tiefenkarten unter anderem auch sensorbedingtes Rauschen enthalten, werden sie vor der weiteren Verwendung in der Regel noch gefiltert, z.B. um Ausreißer zu eliminieren oder Daten zu glätten. Nach dem Entrauschen wird dreidimensionale Information zurückgewonnen, indem aus Bildpixeln mit Hilfe der inversen Matrix K^{-1} 3D Punkte (in Kamera-Koordinaten) erstellt werden. Die Menge aller erzeugten Punkte wird als Punktwolke bezeichnet. Die Punktpositionen werden mit den Positionen ihrer „Ursprungspixel“assoziiert und bilden so eine Vertexkarte $\mathcal{V}$. Neben den Positionen können, je nach konkretem Verfahren, auch weitere punktbezogenen Parameter berechnet werden; üblich ist z.B. Erstellung einer Normalenkarte $\mathcal{N}$.

In den weiteren Pipelinestufen wird auch Rücktransformation einer Punktwolke in Pixelkoordinaten benutzt, z.B. um eine effiziente Suche nach Punkten zu ermöglichen.

2.1.2 Schätzung der Kamerapose / Bildregistrierung

Ein einziges Tiefenbild liefert 3D Information nur über einen begrenzten Ausschnitt der Szene. In den meisten Anwendungsfällen ist jedoch Rekonstruktion eines Objekts aus verschiedenen Blickwinkeln nötig, so dass mehrere Bilder aufgenommen und zusammengeführt werden sollen. Eine Voraussetzung dafür ist ein gemeinsames Koordinatensystem (Modellkoordinaten). Die mit 2.1 erzeugten Punktwolken liegen zuerst in Kamera-Koordinaten vor, welche – bei einer dynamischen Kamera – vom Bild zu Bild unterschiedlich sein können. Um Punkte in die Modellkoordinaten transformieren zu können, muss also die jeweilige Kamerapose bekannt sein. Die Kamerapose des i-ten Frames kann durch die

[3] Aus Übersichtlichkeitsgründen wird im weiteren die komplette Projektion als Multiplikation mit K beschrieben, d.h. Division durch z implizit eingeschlossen.

[4] Koeffizienten höherer Ordnung werden hier nicht betrachtet.

Transformationsmatrix

$$T^i = \begin{Bmatrix} R^i & & & \mathbf{t}^i \\ 0 & 0 & 0 & 1 \end{Bmatrix}, \tag{2.3}$$

beschrieben werden, wo R^i 3×3 Rotationsmatrix und $\mathbf{t}^i$ Translationsvektor sind. [5]

Die zeitlich aufeinander folgenden Bilder (Frames) enthalten in der Regel sich überlappende Regionen. Diese Tatsache wird benutzt, um relative Kameratransformation zwischen den benachbarten Frames zu schätzen, indem sie gegeneinander registriert werden. Mit anderen Worten, es wird in den Frames $i-1$, i nach gemeinsamen Regionen gesucht und anschließend eine Transformation $T^{i\to(i-1)}$ berechnet, die sie wieder (annähernd) zur Deckung brächte. Ist die Anfangspose T^0 [6] bekannt, kann man durch Akkumulation von Frame-zu-Frame-Transformationen die aktuelle Matrix T^i schätzen.

Dient als Registrierungsziel nur der vorherige Frame, spricht man von der Frame-zu-Frame-Registrierung [RHHL02]. Ein Nachteil dieses Ansatzes ist seine Anfälligkeit gegenüber dem Drift. Eine in dieser Hinsicht robustere Variante stellt Frame-zu-Modell-Registrierung dar [NIH+11]. In diesem Fall wird der i-te Frame gegenüber der gesamten geometrischen Information registriert, die im Modell bis einschließlich des $i-1$-ten Frames akkumuliert wurde.

Ein beliebtes Verfahren zur Lösung des Registrierungsproblems ist der **Iterative Closest Point** Algorithmus (**ICP**), der von Besl und McKay in [BM92] vorgeschlagen und nachfolgend von verschiedenen Autoren adaptiert (z.B. in [RL01, NIH+11]) wurde. Der Algorithmus startet mit einer Schätzung der Transformation zwischen der Eingabe- und Zielgeometrie und sucht für die transformierten Eingabepunkte nach den jeweils nächsten Punkten im Registrierungsziel (Korrespondenzen). Anschließend wird die aktuelle Transformation so verfeinert, dass der Fehler nach einer festgelegten Metrik (z.B. Punktabstände) minimiert wird. Dieser Vorgang wird iterativ ausgeführt, bis der Fehler ausreichend klein ist. Nach Rusinkiewicz und Levoy [RL01] lässt sich ein ICP in sechs Schritte gliedern. Der unten stehenden Abschnitt bietet eine kurze Übersicht einzelner Schritte an. Die Beschreibungen der Implementierungsdetails folgen der ICP-Variante aus [NIH+11], welche auch als Basis für das in unserer Arbeit verwendeten Framework [KLL+13, LKS+17] diente; mögliche Alternativen werden nur beispielhaft erwähnt.

Auswahl der Daten

Daten für Registrierung können mit unterschiedlichen Verfahren ausgewählt werden: z.B. durch eine uniforme oder eine Zufalls-Abtastung, durch Extraktion von Punkten mit gewünschten Features. In [NIH+11] werden – sowohl beim zu registrierenden Frame als auch beim Modell – alle verfügbare Punkte genommen.

Registrierung

Die Suche nach den korrespondierenden Punkten im aktuellen Frame und im Registrierungsziel

[5] Damit entspricht R_i der spaltenweise geschriebenen Orientierungsmatrix der Kamera und t_i ihrer Position in Bezugskoordinaten.

[6] Diese wird oft als Identitätsmatrix gesetzt.

kann zu einem rechenintensiven Vorgang werden. In der Originalversion von ICP wird für jeden Punkt $\mathbf{p}_i$ aus dem i-ten Frame die Gesamtmenge der Modellpunkte [7] $\{\mathbf{p}_\mathcal{M}\}$ betrachtet und nach dem nächstliegenden Punkt durchsucht, was die Worst-Case-Komplexität $\mathcal{O}(N_i N_\mathcal{M})$ ergibt (N_i, $N_\mathcal{M}$ ist die Anzahl der Punkte im Eingabeframe bzw. im Modell). Eine wesentliche Suchbeschleunigung stellt die von Blais und Levine [BL95] eingeführte projektionsbasierte Datenassoziierung (*inverse Kamerakalibrierung* in der Terminologie der Autoren) dar, welche in [RHHL02] als eine Optimierung speziell im Kontext von ICP aufgenommen wurde. Hierbei wird jeder Punkt des zu registrierenden Frames in die Modellkoordinaten transformiert und durch eine Projektion in Pixelkoordinaten umgewandelt; der Modellpunkt, der im sich ergebenen Pixel gespeichert ist, wird dann als Korrespondenzkandidat betrachtet. Formal:
Man betrachte den i-ten Eingangsframe mit der Vertexkarte $\mathcal{V}_i$ und einen mit der Pixelposition $\mathbf{u}$ verknüpften Punkt $\mathbf{p}_i = \mathcal{V}_i(\mathbf{u})$. Außerdem sei $\mathcal{V}_{\mathcal{M},i-1}$ die Modell-Vertexkarte in den Kamerakoordinaten des $i-1$-ten Frames und $T^{i\to(i-1)}$ eine Schätzung der Frame-zu-Frame-Transformation. Dann ist

$$\hat{\mathbf{u}} = KT^{i\to(i-1)}\mathcal{V}(\mathbf{u}) \tag{2.4}$$

die geschätzte Pixelposition des zu $\mathbf{p}_i$ korrespondierenden Punkts $\mathbf{p}_{\mathcal{M},i-1} \approx \mathcal{V}_{\mathcal{M},i-1}(\hat{\mathbf{u}})$.

Gewichtung

Die im vorherigen Schritt gefundenen Korrespondenzpaare können unterschiedlich gewichtet werden. Mögliche Kriterien für die Gewichtszuordnung sind z.B. Abstand zwischen den beiden korrespondierenden Punkten oder Sensorungenauigkeit in Abhängigkeit von der Punktposition [RL01].

Ausschluß der Korrespondenzkandidaten

Die assoziierten Punkte können noch zusätzlichen Tests unterzogen werden, anhand deren einige Paare gegebenenfalls aus der Menge der Korrespondenzen auszuschließen sind. In [NIH+11, KLL+13, LKS+17] werden die Distanz zwischen den beiden Punkten und der Winkel zwischen ihren Normalen gegen entsprechende Schwellenwerte ϵ_d bzw. ϵ_θ geprüft, so dass Paare mit

$$\begin{aligned}&\|T^{i\to(i-1)}\mathcal{V}_i(\mathbf{u}) - \mathcal{V}_{\mathcal{M},t-1}(\hat{\mathbf{u}})\| > \epsilon_d\,,\\ &\langle R^{i\to(i-1)}\mathcal{N}_i(\mathbf{u})\,,\,\mathcal{N}_{\mathcal{M},i-1}(\hat{\mathbf{u}})\rangle > \epsilon_\theta\end{aligned}$$

aussortiert werden.

Festlegung einer Fehlermetrik

Der zu minimierende Fehler kann gemäß verschiedener Metriken berechnet werden. Eine naheli-

[7] In [BM92] werden neben den Punkten auch andere geometrische Primitive betrachtet. Da in unserer Arbeit ein punktbasierter Rekonstruktionsansatz verwendet wird (siehe unten), begrenzen wir uns bei den Erörterungen in diesem Kapitel auf 3D Punkte.

gende Fehlerfunktion stellt die Summe der quadratischen Punkt-zu-Punkt-Abstände dar (Punkt-zu-Punkt-Metrik). Eine weitere häufig verwendete Variante ist die Punkt-zu-Ebene-Metrik, bei der Abstände von den Eingabepunkten bis zu den Tangentialebenen in den korrespondierenden Zielpunkten betrachtet werden. Während die erstgenannte Metrik Randbedingungen für alle drei Dimensionen definiert, sind sie bei der Punkt-zu-Ebene-Variante nur für eine Richtung – die Richtung der Ebenennormalen – festgelegt, was in der Regel eine schnellere Konvergenz ermöglicht [CM91]. Die Punkt-zu-Ebene-Fehlerfunktion ist wie folgt definiert:

$$E(T^{i\to(i-1)}) = \sum_{\mathbf{u}\in S} \langle (T^{i\to(i-1)}\mathcal{V}_i(\mathbf{u}) - \mathcal{V}_{\mathcal{M},i-1}(\hat{\mathbf{u}}))\,,\, \mathcal{N}_{\mathcal{M},i-1}(\hat{\mathbf{u}})\rangle^2 \,. \tag{2.5}$$

S ist hier eine Untermenge der Pixelkoordinaten der Eingabe-Vertexkarte, und für jedes $\mathbf{u} \in S$ gilt, dass $\mathcal{V}_i(\mathbf{u})$ einen korrespondierenden Punkt hat.

Fehlerminimierung

Die Frame-zu-Frame-Transformation $T^{i\to(i-1)}$ wird durch Minimierung der Metrik berechnet. In [NIH+11] wird ein Verfahren angewendet, bei dem $T^{i\to(i-1)}$ durch Kleinwinkelnäherung und Linearisierung der Funktion 2.5 iterativ berechnet wird. Die k-te Iteration ändert die Lösung um ein Inkrement

$$\tilde{T}_k^{inc} = \begin{Bmatrix} 1 & \alpha & -\gamma & t_x \\ -\alpha & 1 & \beta & t_y \\ \gamma & -\beta & 1 & t_z \\ 0 & 0 & 0 & 1 \end{Bmatrix} .$$

Durch Bildung eines Parametervektors $\mathbf{x} = (\beta, \gamma, \alpha, t_x, t_y, t_z)^T$ und Linearisierung von 2.5 um die Lösung aus der letzten Iteration, kann das Minimierungsproblem als ein 6×6 lineares Gleichungssystem aufgefasst werden.

Nach der Lösung des Gleichungssystems wird die Gesamttransformation nach der k-ten Iteration als $\tilde{T}_k^{i\to(i-1)} = \tilde{T}_k^{inc}\tilde{T}_{k-1}^{i\to(i-1)}$ berechnet. Die Menge der Korrespondenzen wird vor jeder weiteren Iteration aktualisiert, indem die zuletzt berechnete Frame-zu-Frame-Transformation in 2.4 eingesetzt wird.

Als Abbruchbedingungen für ICP können ein Fehlerschwellenwert τ und eine maximale Anzahl der Iterationen k_{max} definiert werden. Die k-te Iteration wird ausgeführt, wenn

$$k < k_{max} \wedge E(T_{k-1}^{i\to(i-1)}) \geq \tau \,.$$

Der Algorithmus wird üblicherweise mit $T_0^{i\to(i-1)} = \mathbb{I}_4$ initialisiert. Weicht diese von der tatsächlich gesuchten Transformation stark ab, kann es auf die initiale Korrespondenzbildung kritisch auswirken. Eine diesbezügliche Optimierung stellt Einführung einer hierarchischen Struktur dar. Hierbei wird ei-

ne Bildpyramide erzeugt, so dass die Originalvertexkarte $\mathcal{V}$ ihre nullte Ebene bildet und jede weitere Ebene $\mathcal{V}_l$, $l \in \{1, ..., M-1\}$ aus $\mathcal{V}_{l-1}$ durch eine Reduktion der Bildauflösung um Faktor 2 erstellt wird. Diese Vorgehensweise ist auch auf weitere punktbezogenen Daten wie $\mathcal{N}$, falls vorhanden, zu übertragen. Der ICP startet auf der gröbsten Ebene $l = M-1$ und geht zur nächsten über, sobald eine der Abbruchbedingungen erfüllt ist (welche wiederum für jede Ebene separat zu definieren sind). In [NIH+11] wird z.B. die Pyramidenhöhe $M = 3$ verwendet.

2.1.3 Aktualisierung des Modells

Nachdem die aktuelle Kamerapose berechnet wurde, können die Eingabepunkte in die Modellkoordinaten transformiert und mit dem bereits akkumulierten Modell zusammengeführt werden. Es wurden unterschiedliche Ansätze zur Repräsentation und Akkumulation von Modelldaten entwickelt. Eine oft benutzte Lösung stellt der von Curless und Levoy in [CL96] eingeführte volumetrische Ansatz samt seinen Adaptationen (z.B. in [NIH+11]) dar. Gemäß diesem Ansatz wird das Modell als vorzeichenbehaftete Distanzfunktion (*engl.* Signed Distance Function, SDF) in einem diskreten Voxel-Gitter repräsentiert; die Null-Isofläche entspricht dabei der rekonstruierten Objektoberfläche [8]. Die Akkumulation wird als gewichtete Addition der Punkte realisiert. Die Umwandlung zwischen der expliziten Repräsentation als 3D Punkte und impliziten Distanzfunktion kann z.B. mit Raycasting durchgeführt werden.

Beim punktbasierten Ansatz wird ein Modell unmittelbar durch 3D Punkte repräsentiert, so dass rechenintensive Umwandlung zwischen der expliziten und impliziten Darstellung entfällt [KLL+13]. Durch direkte Speicherung von Punktwolken wird außerdem das Problem der ineffizienten Speicherverwaltung, das mit dünnbesetzten regulären Voxel-Gittern einhergeht, automatisch gelöst. Beim volumetrischen Ansatz hingegen müssen spezielle Techniken entwickelt werden (z.B. räumliches Hashing in [KDSX15]), um einen zu hohen Speicherverbrauch zu vermeiden.

Da in unserem Projekt ein punktbasiertes Rekonstruktionsframework aus [KLL+13, LKS+17] verwendet wird, konzentrieren wir uns im Weiteren auf diesen Ansatz. Einzelheiten zum Fusionieren von Modell- und Eingabepunkten findet man in der Beschreibung des Frameworks im Abschnitt 3.1.

2.1.4 Rendering

Da für eine Punktwolke keine explizite Konnektivitätsinformation vorhanden ist, ist ihre Triangulierung für die klassische Dreiecksnetz-Repräsentation eine aufwendige Operation. Eine effiziente Rendering-Alternative für punktbasierte Modelle stellt das sogenannte Splatting dar. Jeder Punkt wird dabei durch einen Splat repräsentiert; der letztere ist als eine Scheibe mit einem definierten Radius vorzustellen, welche durch Projektion in die Bildebene eine elliptische Form bekommt. Die sich überlappenden Splats werden durch Blending vermischt. Mehr Details zu Splatting-Techniken siehe z.B. in [ZPvBG01, BHZK05].

[8] Oft wird die Truncated SDF (TSDF) genannte Variante benutzt, bei der nur die Werte in einem schmalen Intervall um die Null-Isofläche gespeichert werden, während der Rest „abgeschnitten“ wird.

2.2 Bewegungs-Tracking mit weiteren Sensoren

Im vorigen Abschnitt wurden die Grundlagen des Trackings anhand der Tiefendaten einer 3D Kamera bereits dargelegt. An dieser Stelle sind weitere Datenquellen und die darauf aufbauenden Tracking-Verfahren zu betrachten, welche auf modernen mobilen Geräten verfügbar sind.

2.2.1 Inertiales Tracking

Inertialsensoren gehören zur Standardausstattung aktueller Smartphones und Tablets. In der Regel werden diese – kombiniert mit einem Magnetometer – zu einer inertialen Messeinheit (engl. *Inertial Measurement Unit*, **IMU**) zusammengefasst. Genauer gesagt besteht eine IMU aus einem triaxialen Drehratensensor, einem triaxialen Beschleunigungssensor und einem triaxialen Magnetometer. Im folgenden werden die einzelnen Sensoren und die damit messbaren bzw. schätzbaren Daten kurz beschrieben (siehe dazu z.B. [Sab11, YB06, Woo07]).

Drehratensensor liefert aktuelle Drehrate ω um je eine der drei Achsen. Ist die Anfangsorientierung bezüglich eines festgelegten Koordinatensystems bekannt, kann die aktuelle Orientierung durch Integration von Drehraten berechnet werden. Sei $\mathbf{q}_{t_k}$ die Orientierung zum Zeitpunkt t_k in Form von (Einheits-)Quaternion und $\boldsymbol{\omega}_{t_{k+1}}$ der Drehratenvektor aus der nächsten Abtastung, dann ist die nächste Orientierung

$$\mathbf{q}_{t_{k+1}} = \mathbf{q}_{t_k} + \dot{\mathbf{q}}_{t_{k+1}} \Delta t \,,$$

wo $\Delta t = t_{k+1} - t_k$ das Zeitintervall für Euler-Integration und $\dot{\mathbf{q}}_{t_{k+1}} = \frac{1}{2}\mathbf{q}_{t_k} \otimes (0 \; \boldsymbol{\omega}_{t_{k+1}}^T)^T$ die Ableitung der Orientierung nach Zeit sind. [9]

Drehratensensoren ermöglichen eine präzise Schätzung der Orientierung bei sehr dynamischen Bewegungen. Als ein Nachteil dieser Bauteile ist ein Bias zu sehen. Dieses wird durch Integration über längere Zeit akkumuliert, was wiederum einen niederfrequenten Drift erzeugt.

Beschleunigungssensor misst aktuelle Beschleunigung; im Falle eines triaxialen Geräts liegt das Ergebnis als 3D Vektor vor. Bei einem ruhenden Objekt entspricht die gemessene Beschleunigung $\mathbf{a}_m$ der Erdbeschleunigung $\mathbf{g}$ in Sensorkoordinaten. Diese Tatsache kann zur Bestimmung der Orientierung (Neigung) bezüglich der globalen Koordinaten benutzt werden. Eine Rotation um die (globale) vertikale Achse kann jedoch mit einem Beschleunigungssensor nicht erfasst werden, da diese den Vektor $\mathbf{a}_m$ nicht ändern würde.

Befindet sich das Objekt in Bewegung, fließt seine eigene Beschleunigung $\mathbf{a}_o$ ins Ergebnis ein: $\mathbf{a}_m = \mathbf{a}_o + \mathbf{g}$. Bei bekannter Orientierung (z.B. aus Schätzung anhand von Drehraten) ist es möglich, den gemessenen Beschleunigungsvektor in die globalen Koordinaten zu transformieren und, nach Abzug der konstanten Erdbeschleunigung, die aktuelle Position durch doppelte

[9] Der Operator $\otimes$ bezeichnet hier Quaternionen-Multiplikation.

Integration zu schätzen:

$$\begin{aligned} \mathbf{v}^g_{t_{k+1}} &= \mathbf{v}^g_{t_k} + (\mathbf{a}^g_{m,t_{k+1}} - \mathbf{g})\Delta t\,, \\ \mathbf{p}^g_{t_{k+1}} &= \mathbf{p}^g_{t_k} + \mathbf{v}^g_{t_{k+1}}\Delta t\,, \end{aligned}$$

wo $\mathbf{v}$ Geschwindigkeit, $\mathbf{p}$ Position sind und hochgestelltes g auf globale Koordinaten verweist.

Die komplette Bestimmung der aktuellen Pose mit Hilfe von den zwei oben beschriebenen Sensoren lässt sich mit dem Schema aus der Abbildung beschreiben. Eine solche Schätzung führt jedoch schon relativ schnell zu einem Drift der Position, da der durch Rauschen bedingte Fehler beim zweifachen Integrieren quadratisch wächst.

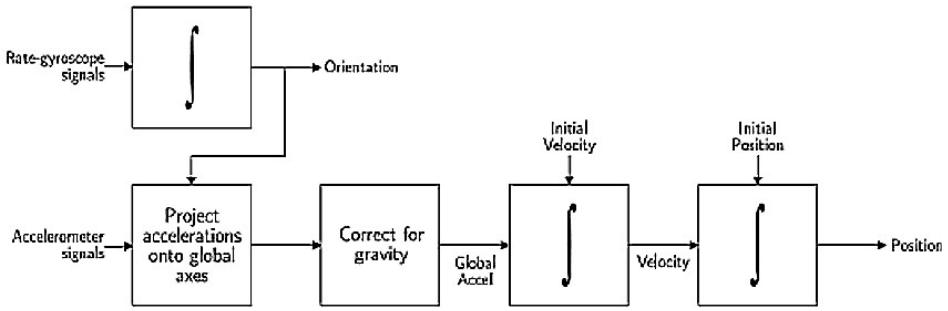


Bild 2.2: Bewegung-Tracking mit Drehraten- und Beschleunigungssensor, Quelle: [NZW+13].

Magnetometer gibt die Horizontalkomponente des Erdmagnetfeldes in Sensorkoordinaten zurück. Setzt man die Richtung des magnetischen Nordens als globale x-Achse fest, ist es dadurch möglich, die Orientierung des Sensors in der Horizontalebene zu berechnen. Damit kann man Ergebnisse aus dem Beschleunigungssensor ergänzen, um die Orientierung für alle drei Freiheitsgrade zu bestimmen.

Sensor-Fusion Mit dem Ziel, Nachteile einzelner Sensortypen zu minimieren, werden Drehratensensor, Beschleunigungssensor und (oft auch) Magnetometer komplementär benutzt, indem Ergebnisse aus allen verfügbaren Quellen mit Hilfe eines Filters fusioniert werden. Dabei wird die Information von dem Beschleunigungssensor und dem Magnetometer verwendet, um das niedrigfrequente Rauschen der Drehratensensors zu reduzieren; die Daten des Drehratenssensors helfen ihrerseits gegen das hochfrequente Rauschen der beiden ersteren Sensoren.

In der Literatur findet man verschiedene Verfahren zum Fusionieren von Sensordaten. Zwei oft benutzte Beispiele davon sind Komplementärfilter und erweitertes Kalman-Filter (engl. *Extended Kalman Filter*, **EKF**), die wiederum als Oberbegriffe für eine Reihe von unterschiedlichen Ausprägungen stehen. Ein derartiger Filterungsalgorithmus im Verbund mit IMU reduziert vor allem Fehler bei der Orientierungsschätzung; für die Schätzung der Position kann hingegen nur der Beschleunigungssensor verwendet werden, so dass keine komplementäre Datenquelle

vorhanden ist.

2.2.2 Visuell-inertiale Odometrie

Ein weiteres Verfahren für Bestimmung der Position und der Orientierung stellt video-basiertes Tracking dar (siehe z.B. [NNB04]). Hierbei handelt es sich um eine Schätzung der relativen Bewegungen anhand der aufeinander folgenden Bildern (Video-Frames). Der Vorgang beinhaltet Extraktion der Merkmalspunkte (engl. *Feature Points*) aus dem Bild, Registrierung der extrahierten Punkte gegen die Merkmalspunkte des vorherigen Bilds und Schätzung der relativen Transformation. [10]

Verfügt das zu trackende System sowohl über eine IMU als auch über eine Kamera, kann ein solches video-basiertes Verfahren als zusätzliche Quelle für Sensor-Fusion verwendet werden, um eine robustere Posenschätzung zu erhalten. Hierbei kann der Positionsvektor der extrahierten Features z.B. als ein weiteres Parameter in das EKF einfließen [DGKR13]. Solche Art von Sensor-Fusion wird als visuell-inertiale Odometrie (engl. *Visual Inertial Odometry*, **VIO**) bezeichnet.

[10] Vgl. Schätzung der Kamerapose auf einem 3D Datensatz in 2.1.2.

Kapitel 3

Bekannte Verfahren

In diesem Kapitel werden Verfahren dargestellt, die entweder in der vorliegenden Arbeit direkt eingesetzt werden, oder mit ihr thematisch eng verbunden sind und damit als Referenz dienen können. Der erste Abschnitt beschreibt das Framework, das die Grundlage für 3D Rekonstruktion in unserem Projekt bildet. Der zweite Abschnitt gibt eine kurze Überblick über die Arbeiten, in denen eine Kombination aus inertialem / visuell-inertialem Tracking und ICP bereits implementiert wurde.

3.1 Punktbasiertes Rekonstruktionsframework

Zur Rekonstruktion von 3D Oberflächen aus Tiefendaten wird in dieser Arbeit das punktbasierte Framework verwendet, das von Keller et al. in [KLL+13] eingeführt wurde. Lefloch et al. erweiterten in [LKS+17] den ursprünglichen Ansatz, indem zusätzlich Krümmungsinformation einbezogen wurde. Die Rekonstruktionspipeline des Frameworks lässt sich im Wesentlichen auf die im Abschnitt 2.1.2 beschriebene allgemeine Struktur zurückführen; hier sind nur noch die framework-spezifischen Details zu betrachten.

Vorverarbeitung

Die Vorverarbeitung der Tiefenkarten schließt Entfernung der Ausreißer und Glättung mit einem Bilateralfilter ein. Bei der anschließenden Erzeugung der Punktwolke werden, neben Positionen (Vertexkarte), noch die im Folgenden beschriebenen Per-Punkt-Parameter bestimmt. Punktnormalen werden als zentrale Differenzen auf der Vertexkarte berechnet. Räumliche Ausdehnung eines Punktes wird durch seinen Radius definiert; letzteren bestimmt man anhand der Nachbarschaftsgröße des jeweiligen Punktes. Alle diese Daten werden in Form von Karten $\mathcal{V}_i, \mathcal{N}_i$ bzw. $\mathcal{R}_i$ abgelegt. In [LKS+17] wird außerdem Krümmunsinformation aus der Normalenkarte abgeleitet; die erste Hauptrichtung $\hat{\mathbf{e}}_\mathbf{1}$ und die beiden Krümmungswerte k_1, k_2 sind dann ebenso als eine Karte $\mathcal{K}_i = \{\hat{\mathbf{e}_\mathbf{1}}, k_1, k_2\}$ zu speichern.

Schätzung der Kamerapose

Das Kamera-Tracking wird mit Hilfe eines hierarchischen ICP-Algorithmus mit Frame-Zu-Modell-

Registrierung und Punkt-zu-Ebene-Metrik, ähnlich wie in [NIH+11], realisiert. Die Höhe der Bildpyramide beträgt 3 Ebenen. Die maximale Anzahl der Iterationen – von der gröbsten zur nullten Ebene – wird auf 4, 5 bzw. 10 gesetzt. Die als Registrierungsziel fungierende Modell-Vertexkarte $\mathcal{V}_{\mathcal{M},i-1}$ wird durch lokale Oberflächenrekonstruktion in der Umgebung der Modellpunkte erstellt.

In [LKS+17] werden Korrespondenzpaare bei der Fehlerminimierung unterschiedlich Gewichtet, wobei das Gewicht

$$w(\hat{\mathbf{u}}) = 1 + \frac{\delta_c}{[\hat{\mathbf{p}}_M]_z^2} \exp\left(-\frac{1}{2}\left[\frac{100}{\hat{k}_M^{max}}\right]^2\right)$$

vom absoluten Maximum der Hauptkrümmung $\hat{k}_M^{max}$ des jeweiligen Modellpunktes abhängt (δ_c ist der Kontrollparameter). Die Fehlerfunktion 2.5 nimmt dann folgende Form an:

$$E(T^{i\to(i-1)}) = \sum_{\mathbf{u}\in S} w(\hat{\mathbf{u}}) \langle (T^{i\to(i-1)}\mathcal{V}_i(\mathbf{u}) - \mathcal{V}_{\mathcal{M},i-1}(\hat{\mathbf{u}})) \, , \, \mathcal{N}_{\mathcal{M},i-1}(\hat{\mathbf{u}}) \rangle^2 \, . \tag{3.1}$$

Aktualisierung des Modells

Zum Fusionieren der Modell- und Eingabepunkte ist eine Datenassoziierung erforderlich, die mit der Korrespondenzsuche in ICP zu vergleichen ist. Das punktbasierte Modell liegt jedoch als „lose“ Punktwolke vor, welche in einem flachen Array $M[\,]$ gespeichert ist. Mit dem Ziel, eine effiziente Suche nach Punkten zu ermöglichen, wird daraus eine Indexkarte $\mathcal{I}_i$ erstellt, indem man die Modellpunkte in die aktuellen Kamerakoordinaten transformiert, in die Bildebene projiziert und in den sich ergebenden Pixeln die Punktindizes speichert. Je nach Kamerapose können dabei mehrere Punkte auf dasselbe Pixel projiziert werden. Um Informationsverlust durch solche Punktkollisionen möglichst zu minimieren, wird die Auflösung der Indexkarte im Vergleich zum Eingabebild künstlich erhöht (in [KLL+13] wird 4x4 und in [LKS+17] 2x2 Supersampling verwendet).

Bei der Suche wird für jede Supersampling-Pixelkoordinate $\mathbf{u}$ ihre Entsprechung bei der normalen Auflösung $\mathbf{u}'$ berechnet. Man betrachtet für jedes $\mathbf{u} \in \mathcal{I}_i$ und den damit assoziierten Punkt $\mathbf{p}_\mathcal{M} = M[\mathcal{I}_i(\mathbf{u})]$ die Nachbarschaft von $\mathcal{V}_i(\mathbf{u}')$ als mögliche Korrespondenzkandidaten. Die Eingabepunkte, deren Tiefe mehr als um einen Schwellenwert $\pm\delta_{depth}$ von der Tiefe des jeweiligen Modellpunkts abweicht beziehungsweise deren Normalen sich mehr als um einen Winkel δ_{norm} von der Modellnormale unterscheiden, werden aussortiert.

Hat man ein Korrespondenzpaar gefunden, wird die Position des Modellpunkts durch eine gewichtete Mittelung mit dem Eingabepunkt aktualisiert:

$$\mathbf{p}_\mathcal{M} \leftarrow \frac{w_i\mathbf{p}_i + w_\mathcal{M}\mathbf{p}_\mathcal{M}}{w_i + w_\mathcal{M}} \, , \, w_\mathcal{M} \leftarrow w_i + w_\mathcal{M} \, . \tag{3.2}$$

w_i ist hier das Konfidenzgewicht des Eingabepunkts, das mit dem (normalisierten) radialen Ab-

stand vom Sensorzentrum γ zusammenhängt: mit dem steigenden Abstand sinkt die Messgenauigkeit und damit auch die Konfidenz. Es wird als gaußsches Rauschen berechnet:

$$w_i = e^{-\gamma^2/2\sigma^2} , \tag{3.3}$$

wo $\sigma = 0.6$ ist. $w_{\mathcal{M}}$ steht für den Konfidenzzähler des Modellpunkts.
In [KLL+13] werden die Normalen beim Fusionieren analog zu den Positionen behandelt. In [LKS+17] wird das komplette Modell-Darboux-Dreibein $\mathbf{C}_{\mathcal{M}} = (\hat{\mathbf{e}}_{\mathbf{1},\mathcal{M}}, \hat{\mathbf{e}}_{\mathbf{2},\mathcal{M}}, \hat{\mathbf{n}}_{\mathcal{M}},)$ durch eine gewichtete Rotation in Richtung des Eingabe-Darboux-Dreibeins $\mathbf{C}_{\rangle} = (\hat{\mathbf{e}}_{\mathbf{1},\mathbf{i}}, \hat{\mathbf{e}}_{\mathbf{2},\mathbf{i}}, \hat{\mathbf{n}}_{\mathbf{i}})$ mit dem letzteren zusammengeführt:

$$\mathbf{C}_{\mathcal{M}} \leftarrow \mathbf{R}(\hat{\mathbf{a}}, \alpha\phi)\mathbf{C}_{\mathcal{M}} ,$$

mit Rotationsmatrix $\mathbf{R}(\hat{\mathbf{a}}, \alpha\phi) = \mathbf{C}_i\mathbf{C}_{\mathcal{M}}^T$, Rotationsachse $\hat{\mathbf{a}}$, Winkel ϕ und Gewicht $\alpha = \frac{w_i}{w_i + w_{\mathcal{M}}}$.
Schließlich wird die Krümmungsamplitude nach dem gleichen Schema wie die Positionen fusioniert.
Der Konfidenzzähler $w_{\mathcal{M}}$ bestimmt, ob sich ein Modellpunkt im stabilen oder instabilen Zustand befindet. Wird für einen Eingabepunkt $\mathbf{p}_i$ beim oben beschriebenen Vorgang keine modellseitige Korrespondenz gefunden, wird er als neuer, instabiler Punkt mit $w_{\mathcal{M}} = w_i$ zum Modell hinzugefügt. Jedes Fusionieren mit einem Eingabepunkt erhöht den Konfidenzzähler. Erreicht er den Schwellenwert $w_{\mathcal{M}} \geq w_{stable}$ geht der Punkt in den stabilen Zustand über. Bleibt der Punkt hingegen über eine längere Zeit im instabilen Zustand, wird er aus dem Modell entfernt.

3.2 Kombination aus (visuell-)inertialem Tracking und ICP

Der Ansatz, ICP-basierte Kameraposenschätzung mit (visuell-)inertialem Tracking zu kombinieren, ist im Kontext einer 3D Rekonstruktion nicht neu. Nießner et al. verwenden in [NDF14] ein System aus einer Kinect-RGB-D-Kamera und einem Smartphone, wobei das letztere als IMU fungierte. Die über das Android-API gelesenen Daten der Drehraten- und Beschleunigungssensors wurden direkt – ohne Anwendung eines Sensor-Fusion-Filters – zur Schätzung der aktuellen Orientierung bzw. Position eingesetzt. Die Positionsschätzung wies größere Ungenauigkeiten aufgrund der doppelten Integration ohne komplementäre Fehlerkompensation auf (siehe dazu 2.2).

Aus der mit den IMU-Daten geschätzten Kamerapose wird dann die jeweilige Frame-zu-Frame-Transformation Δ_{IMU} berechnet, die zur Initialisierung von ICP verwendet wird. Die Autoren führen eine Metrik für Evaluierung von ICP-Fehlern ein. Im Falle, dass der Fehler einen empirisch festgestellten Schwellenwert übersteigt, wird das ICP-Ergebnis abgelehnt und stattdessen die Posenschätzung mit den inertialen Sensoren als aktuelle Kamerapose verwendet.

Die Autoren konnten eine Beschleunigung der ICP-Konvergenz durch Initialisierung mit inertialem Tracking beobachten. Außerdem wurde die bekannte ICP-Schwäche bei den Szenen mit mangelnden geometrischen Details teilweise kompensiert, indem man solche Szenenabschnitte mit reinem

IMU-basierten Tracking überbrücken konnte. Zugleich weisen Nießner et al. auf eine hohe Anzahl der Fehlerfälle bei ihrem System auf, die sie auf die bereits erwähnte mangelhafte Positionsschätzung durch Beschleunigungssensor und auf den globalen Sensordrift zurückführen.

In [KDSX15] beschreiben Klingensmith et al. ein System für 3D Rekonstruktionen auf mobilen Geräten, das insbesondere für das Scannen von großflächigen Szenen konzipiert ist. Das CHISEL genannte System verwendet eine Kombination aus VIO und ICP. Als Hardware fungieren zwei Google-Tango-Geräte: ein „Peanut"-Smartphone und ein „Yellowstone"-Tablet (das letztere Modell wurde auch in der vorliegenden Arbeit eingesetzt). Die Tiefenerfassung geschieht hier mit Hilfe der geräteeigenen Tiefensensoren, deren Aktualisierungsrate (3-6 Hz) für ein stabiles Kamera-Tracking mit ICP zu langsam ist. Die Autoren überwinden dieses Problem, indem sie die aktuelle Kamerapose aus dem visuell-inertialen Tracking-System von Tango abrufen und diese anschließend mit ICP korrigieren. Die verwendete ICP-Variante minimiert quadratische TSDF.

Mit dem oben beschriebenen System konnten großflächige Szenen mit einer Auflösung von 2 cm rekonstruiert werden. Die Autoren erwähnen, dass ein Drift über längere Zeit nicht vermieden werden konnte.

Abschließend ist noch zu bemerken, dass in den beiden zitierten Arbeiten – im Unterschied zum vorliegenden Projekt – eine volumetrische Rekonstruktion verwendet wird. Außerdem legen die Autoren den Schwerpunkt auf die 3D Raumrekonstruktion, während wir uns auf die Rekonstruktion von mittelgroßen Objekten konzentrieren.

Kapitel 4

Konzeption

Wie in der Einleitung in allgemeinen Zügen bereits formuliert, besteht das Ziel dieser Masterarbeit in der Evaluierung des Kamera-Trackings innerhalb der punktbasierten 3D Rekonstruktionspipeline auf einer mobilen Plattform. Insbesondere gilt es, ein Multi-Sensor-Tracking zu implementieren und seine Effektivität experimentell zu verifizieren. In diesem Kapitel sollen die zu lösenden Probleme genau definiert und vor allem entsprechende Lösungskonzepte vorgeschlagen werden.

4.1 Systementwurf

4.1.1 Komponenten

Auch wenn hier einige technische Details bereits vorweggenommen werden müssen, versteht sich der folgende Abschnitt in erster Linie als eine Beschreibung der logischen Bestandsteile des Systems. Die eingesetzten Soft- und Hardwarekomponenten werden im Kapitel 5 ausführlicher beschrieben.

Das Framework

Das Kernstück der Rekonstruktionssoftware bildet das punktbasierte Framework (3.1). Eine von Lefloch et al. gefertigte Implementierung wurde uns zur Verfügung gestellt.

Tablet

Die Rekonstruktionssoftware wird auf einem Tango Tablet [1] ausgeführt. Auch wenn heutzutage beinahe jedes Tablet-Modell mit einer IMU ausgerüstet ist, ist die direkte Posenberechnung anhand der Sensordaten sehr fehleranfällig (siehe 2.2), so dass ein Sensor-Fusion-Algorithmus vorzuziehen ist, dessen Entwicklung wiederum eine komplexe Aufgabe ist. Auf dem Tango-Gerät ist Bewegungstracking auf der Basis eines VIO-Verfahrens bereits implementiert, was das Hauptargument für die Wahl dieses Tablets ist. Auf diese Weise können wir das Tango-Tracking als Blackbox benutzen und uns auf seine Integration in die Rekonstruktionspipeline und auf die Evaluierung konzentrieren.

[1] `https://developers.google.com/tango/hardware/tablet#technical_specifications`

Tiefenkamera

Als Quelle der Tiefendaten dient im System ein extern angeschlossenes CamBoard pico flexx [2] der Firma PMD. Obwohl das Tango Tablet über eine integrierte Tiefenkamera verfügt, ist es vor allem bei schnelleren Kamerabewegungen zu erwarten, dass ihre Aktualisierungsrate von 5 Hz für eine zuverlässige Posenschätzung mit ICP nicht ausreichend ist (ausführlicher dazu in 4.1.2). Da ein Teil der Evaluierungsziele auch die originale, rein ICP-basierte Tracking-Version aus dem Framework einschließt (siehe 4.2), entschieden wir uns für die oben genannte PMD-Kamera, bei der auch höhere Aktualisierungsraten möglich sind.

4.1.2 Einschränkungen

Im Folgenden werden Einschränkungen analysiert, mit denen bei der Portierung des Rekonstruktionsframeworks auf das oben skizzierte System zu rechnen ist. Die vorliegende Framework-Implementierung wurde ursprünglich für die Verwendung auf Desktop-Rechnern in Verbindung mit einer Kinect-Kamera ausgelegt. Daraus resultieren hauptsächlich zwei Problemaspekte, die das zuverlässige Funktionieren des Trackings auf dem mobilen System beeinträchtigen können.

Begrenzte Systemressourcen

Angesichts einer geringeren CPU- und GPU-Leistung ist anzunehmen, dass die mittlere Pipeline-Durchlaufzeit auf einem mobilen Gerät höher als bei einer Desktop-Version ausfallen wird. Zugleich setzen hypothetische Anwendungsszenarien des angestrebten Systems a priori eine frei bewegliche Kamera voraus. Schnellere Bewegungen bei einer relativ niedrigen Bearbeitungsgeschwindigkeit können dazu führen, dass Kameratransformationen zwischen benachbarten Frames für eine zuverlässige Schätzung mit ICP zu groß sind. Beispielsweise ist es bei der Datenassoziierung möglich, dass die Pixelposition, die sich aus der Projektion eines Eingabepunkts ergibt, nicht mit der Position des (in der Realität) korrespondierenden Modellpunkts übereinstimmt. Auf diese Weise können Korrespondenzpaare nicht mehr korrekt gebildet werden.

Niedrige Kameraauflösung

Es bestehen erhebliche Unterschiede in der Auflösung zwischen CamBoard pico flexx und den in [LKS+17] eingesetzten Kameras (224x171 gegen 320x240 bei Kinect 1 bzw. 512x424 bei Kinect 2). Diese Tatsache kann ebenso Auswirkungen auf die Tracking-Genauigkeit haben. Vor allem die Schätzung der Normalen dürfte bei einer niedrig aufgelösten Objektoberfläche an Präzision einbüßen, da diese auf der Finite-Differenzen-Methode basiert und der Diskretisierungsschritt in diesem Falle größer wird. Die Punktnormalen dienen ihrerseits als Grundlage für die Krümmungsberechnung (siehe 3.1). Diese Argumente legen nahe, dass die in [LKS+17] eingeführten krümmungsbasierten Features bei der gegebenen Kameraauflösung nicht mehr zu den gewünschten Ergebnissen führen werden. Diese Annahme ist jedoch experimentell zu verifizieren.

[2] http://pmdtec.com/picoflexx/

4.1.3 Notwendige Änderungen

Zum Umsetzen unseres Konzepts sind einige Änderungen am Rekonstruktionsframework erforderlich. Zum einen sind sie durch Integration der angestrebten, kombinierten Tracking-Methode bedingt, zum anderen stehen sie im Zusammenhang mit der Anpassung an die neue Hardwareumgebung.

Multi-Sensor-Tracking

Wie oben bereits angedeutet, fungiert das VIO-basierte Verfahren aus der Tango-Bibliothek als zusätzliche Quelle für Kameraposenschätzung in unserem System. Das Tango-Tracking wird mit 30 Hz aktualisiert, so dass auch bei dynamischen Bewegungen – und unabhängig von der Bearbeitungszeit im Rekonstruktionspipeline – immer eine aktuelle Pose verfügbar ist. Andererseits ist zu bedenken, dass das visuell-inertiale Tracking – besonders bei einer Anwendung über längere Zeit – einen nicht zu vernachlässigenden Drift akkumulieren kann. Es ist anzunehmen, dass der ICP-Algorithmus, durch Verwendung von Frame-zu-Modell-Registrierung, sich in dieser Hinsicht robuster verhalten wird. Diesen Argumenten folgend, entschieden wir uns für eine Implementierung, bei der das visuell-inertiale Tracking, ähnlich wie in [KDSX15] und [NDF14], für Initialisierung von ICP eingesetzt wird, während der letztere Algorithmus die endgültige Pose berechnet. Angestrebt ist dabei, dass ICP näher zu der gesuchten Zieltransformation startet und dadurch auch schneller konvergiert, gleichzeitig aber auch die Genauigkeit gegenüber der initialen (VIO)-Schätzung und dem Verfahren mit herkömmlicher Initialisierung erhöht.

Da die über das Tango-API abrufbare Pose auf das Tablet bzw. seine Komponenten bezogen ist und in unserem System eine externe Tiefenkamera benutzt wird, setzt die Verwendung des Tango-Trackings eine Kalibrierung voraus. Für diesen Prozess kann als Gegenstück zum CamBoard pico flexx alternativ einer der Tango-Sensoren – IMU, Tiefensensor, RGB-Kamera – gewählt werden. Hier wurde die Farbkamera im Verbund mit 2D Infrarot-Bildern von CamBoard pico flexx verwendet. Das Kalibrieren zwischen zwei 2D Kameras wurde mit der Methode für ebene Muster durchgeführt (siehe dazu z.B. [Zha00]). Für die Wahl sprach die relativ einfache Realisierbarkeit vor Ort: ein solches Kalibrierungsmuster ist leicht herzustellen und mitzutragen, die Kameras können dabei statisch (im Unterschied zu IMU-Kamera-Kalibrierung, siehe z.B. [FMR+11]) verwendet werden.

Bei dieser Methode wird ein zweidimensionales Muster bekannter Größe (z.B. ein Schachbrettmuster) aus verschiedenen Positionen aufgenommen. Die Feature-Punkte (Mustereckpunkte) in den Bildern der beiden Kameras werden extrahiert und dienen als korrespondierende Punkte. Anhand der Korrespondenzen können dann intrinsische und extrinsische Kameraparameter geschätzt werden. Wir benutzen für Feature-Extraktion die `findChessboardCorners` und für Kalibrierung die `stereoCalibrate` Funktion aus OpenCV. Die intrinsischen Parameter und die Verzerrung der Kameras werden über das jeweilige API ermittelt und für die letztgenannte Funktion festgesetzt, so dass nur noch extrinsische Matrizen zu schätzen sind. Das Endergebnis ist eine Matrix $C^{D \to RGB}$, die die Transformation aus der Pose der externen Tiefenkamera in die Pose der Tango-RGB-Kamera durchführt.

Die Kalibrierung wird einmalig vor dem Starten des Scan-Vorgangs durchgeführt; die Matrix kann gespeichert und wieder verwendet werden, solange die Tiefenkamera in der gleichen Position montiert bleibt.
Das komplette Zusammenspiel zwischen den beiden Tracking-Arten lässt sich mit dem folgenden Algorithmus beschreiben:

1. Nach der Eingabe des i-ten Frames in die Pipeline rufe die aktuellste Pose der Tango-Farbkamera T_i^{RGB} ab.
2. Transformiere sie in die Pose der Tiefenkamera: $T_i^{VIO} = T_i^{RGB}\, C^{D \to RGB}$.
3. Berechne Frame-zu-Frame-Transformation und setze sie in die 0-te ICP-Iteration ein:

$$T_{0,z}^{i \to (i-1)} = (T_{i-1}^{VIO})^{-1}\, T_i^{VIO}\ ,$$

wo z die gröbste Ebene der Bildpyramide ist.

Entzerrung der Tiefenkarten

In der vorliegenden Framework-Implementierung wird die Verzeichnung der eingehenden Tiefenbilder nicht speziell behandelt. Im Vergleich zu den Kinect-Kameras, weist jedoch die im pico flexx verbaute Linse höhere Verzeichnungskoeffizienten auf (siehe 4.1.3), so dass diese in unserem System nicht zu vernachlässigen sind. Aus diesem Grunde kann bei der Erstellung einer Punktwolke die jeweilige Pixelposition $\mathbf{u}$ nicht direkt in 3D zurückprojiziert werden, sondern zuerst ist die entzerrte Position zu berechnen (siehe 2.2). Wir benutzen für die Entzerrung die OpenCV-Funktion `undistortPoints`. Diese erzeugt einen verzerrten Punkt $(x^d, y^d, 1) = K^{-1}\mathbf{u}$ und approximiert anschließend mit einem iterativen Algorithmus anhand der bekannten Verzeichnungskoeffizienten die entzerrten Koordinaten x^u, y^u. Die dafür benötigten intrinsischen Parameter werden aus dem PMD-API abgelesen und an die Funktion übergeben. Da x und y nur von der intrinsischen Matrix K^{-1} und den Koordinaten $\mathbf{u}$ abhängen und folglich für die jeweilige Pixelposition invariant bleiben, können sie beim Programmstart berechnet und in einer Entzerrungskarte $\mathcal{U}$ gespeichert werden. Während der Ausführung der Pipeline wird dann aus dem Pixel $\mathbf{u}$ im i-ten Frame der Punkt

$$\mathbf{p}_i = \mathcal{D}(\mathbf{u})(\mathcal{U}()\mathbf{u})^T, 1)^T$$

erzeugt.

Konfidenz-Berechnung

Im letzten Kapitel wurde unter anderem beschrieben, dass im Framework eine Gewichtung der Eingabe-Punkte in Abhängigkeit von ihrem Abstand zum Sensorzentrum verwendet wird (3.3). Dabei beschreibt die verwendete Formel ein generisches Sensormodell. Zugleich ist zu berück-

[3] k1, k2, k3 sind radiale und p1, p2 tangentiale Verzeichnungskoeffizienten. Die Daten für PMD wurden über das entsprechende API abgelesen, für Kinect [FMR+15] entnommen.

Tabelle 4.1: Verzeichnungskoeffizienten [3]

Koeffizienten	pico flexx	Kinect V2
k1	0.067256	0,09357
k2	-2.55499	-0,27394
k3	4.80748	0,09288
p1	0.001609	n.a.
p2	-0.002745	n.a.

sichtigen, dass Daten von der pico flexx Kamera, neben den eigentlichen Tiefenwerten, auch Per-Pixel-Konfidenzen enthalten. Es ist zu erwarten, dass eine auf ein konkretes Kameramodell abgestimmte Konfidenzbestimmung zu mehr Genauigkeit beitragen kann. Infolgedessen wird in unserem System die ursprüngliche Gewichtsberechnung durch Kamerawerte ersetzt. Da die zusammen mit der Tiefe gespeicherte PMD-Konfidenz $c_i \in \{0, ..., 7\}$ ein 3-bit-Integer darstellt, ist noch eine Umwandlung $c_i \mapsto w_i \in [0, 1]$ notwendig. w_i wird dann in 3.2 eingesetzt.

Reduzierung der Bildpyramide

Die hierarchischen Pyramidenebenen innerhalb des ICP-Algorithmus bewirken eine sukzessive Verkleinerung der Kartenauflösung um Faktor 2 (siehe Abschnitt 2.1.2). Setzt man jedoch die in [LKS+17] verwendeten Kameraauflösungen als Referenz für unser System, dann ist zu beachten, dass eine 1,5- bis 2,5-fache Vergröberung der Eingabebilder bereits hardware-bedingt gegeben ist (entsprechende Auflösungen siehe im Abschnitt 4.1.2). Damit entspricht die gröbste, zweite Ebene in [LKS+17] ungefähr der Ebene 1 im mobilen System. Ausgehend von dieser Tatsache kann angenommen werden, dass eine Reduzierung der Bildpyramide auf zwei Ebenen in unserem Projekt die Tracking-Genauigkeit nicht beeinträchtigen wird. Gleichzeitig wird dadurch die Erzeugung von zusätzlichen Karten gespart.

4.2 Evaluierungsziele

Die Evaluierung soll im Wesentlichen zwei Gruppen von Fragen beantworten. Zum einen ist zu verifizieren, ob das vorliegende Rekonstruktion-Framework auch auf einer mobilen Plattform mit aktuell verfügbaren Ressourcen akzeptable Ergebnisse liefern kann und insbesondere ob das Kamera-Tracking mit ICP unter diesen Bedingungen präzise genug funktioniert. Zum anderen soll experimentell festgestellt werden, ob das kombinierte Multi-Sensor-Tracking einen Mehrwert gegenüber der ursprünglichen Tracking-Variante darstellt. Im Weiteren werden Evaluierungsziele ausführlicher formuliert.

4.2.1 Genauigkeit des ICP-basierten Kamera-Trackings

In den Abschnitten 4.1.2 und 4.1 wurden einige Annahmen über Einfluss einzelner Parameter auf das Funktionieren des Trackings im konzipierten System geäußert. Sie lassen sich in Form von Hypothesen zusammenfassen:

- Unter der gegebenen Bildauflösung kann die Tracking-Genauigkeit durch das Einbeziehen der **Krümmungs**information **nicht positiv** beeinflusst werden.
- Unter der gegebenen Bildauflösung kann die Höhe der **Bildpyramide** auf **zwei Ebenen** reduziert werden.
- Verwendung der **sensorspezifischen Konfidenzgewichte** führt zu einer **höheren Präzision** im Vergleich zur generischen Gewichtung.

Zur experimentellen Verifizierung der oben stehenden Hypothesen soll das Framework mit unterschiedlichen Einstellungen ausgeführt werden. Wir bezeichnen die Konfiguration mit den deaktivierten Krümmungs-Features, drei Pyramidenebenen und der generischen Gewichtung als Basis-Konfiguration. Die Genauigkeit des Trackings ist dann bei den folgenden Konfigurationen vergleichend zu evaluieren:

1. **Basis**,
2. Basis mit **Krümmungs**-Features,
3. Basis mit **2 Pyramidenebenen**,
4. Basis mit **PMD-Konfidenz**.

4.2.2 Vergleich Tracking mit ICP vs. Multi-Sensor-Tracking

Aus den bisher aufgeführten Überlegungen bezüglich der kombinierten Tracking-Variante lassen sich ebenso einige Hypothesen ableiten:

- Mit dem kombinierten Verfahren wird eine präzisere Schätzung der Kamerapose sowohl im Vergleich zum rein ICP-basierten Tracking als auch zur alleinigen VIO erreicht.
- Im kombinierten Verfahren konvergiert ICP schneller als im rein ICP-basierten Verfahren.

Obwohl die mit VIO geschätzten Posen als solche nicht direkt für Rekonstruktion verwendet werden, sollen sie bei der Evaluierung der Tracking-Präzision als eine zusätzliche Referenz mit betrachtet werden, um festzustellen, wie sich die anschließende Korrektur mit ICP auf die Posengenauigkeit auswirkt. Insgesamt werden folgende Aspekte und Konfigurationen evaluiert:

1. **Genauigkeit des Trackings**

(a) **Basis**,

(b) Basis mit **Initialisierung durch VIO**,

(c) **VIO**.

2. **Schnelligkeit der ICP-Konvergenz**

(a) **Basis**,

(b) Basis mit **Initialisierung durch VIO**.

4.2.3 Performance

Frameraten stellen ein wichtiges Evaluierungskriterium für jede Echtzeitanwendung dar. Angesichts der begrenzten Hardware-Ressourcen ist es nicht vom Anfang an vorauszusagen, ob beim beschriebenen System Frame-Bearbeitungszeiten erreicht werden können, bei denen eine zuverlässige 3D Rekonstruktion möglich ist. Andererseits können unterschiedliche Framework-Konfigurationen unter anderem auch die Bearbeitungszeit beeinflussen (z.B. negativ durch Berechnung zusätzlicher krümmungsbezogener Daten oder positiv durch weniger ICP-Iterationen). Aus diesem Grund, wird für jede Konfiguration zusätzlich die mittlere Framerate berechnet, auch wenn die Evaluierung der Performance nicht zu den Hauptzielen dieser Arbeit gehört.

Die Tests sollen außerdem die Frage klären, inwieweit die erreichbare Performance für eine Kameraverfolgung auch bei schnelleren Bewegungen ausreichend ist. Zu diesem Zweck sind zusätzliche Versuche mit einer erhöhten Geschwindigkeit durchzuführen, wobei die Basis- und Multi-Sensor-Konfiguration vergleichend evaluiert werden.

4.3 Evaluierungsmethoden

Im vorherigen Abschnitt wurden Evaluierungsziele formuliert, die mehrere Kriterien einschließen: Genauigkeit des Kamera-Trackings in Abhängigkeit von verschiedenen Parametern, Einfluss der Initialisierung mit der Pose aus VIO auf die Konvergenz von ICP, Pipeline-Performance bei jeder zu testenden Konfiguration. Im Folgenden sollen Evaluierungsmethoden vorgestellt werden, die für die genannten Kriterien anzuwenden sind.

4.3.1 Tracking-Genauigkeit

Für die nachträgliche Evaluierung der Tracking-Präzision wird bei der Programmausführung die vom ICP berechnete Kamerapose für jeden Frame persistent gespeichert. Beim kombinierten Verfahren speichert man zusätzlich die Posen aus dem Tango-Tracking. Die unten beschriebene Evaluierungsmethode setzt voraus, dass die Kamera während des Versuchs eine kreisförmige Bewegung ausführt. Der genaue Versuchsaufbau, der diese Bedingung erfüllt, wird im Kapitel 5 beschrieben.

Diese Kenntnis über die reelle Laufbahn der Kamera wird ausgenutzt, um die Genauigkeit des Trackings anhand der Abweichungen von der zu erwartenden Bewegungsform zu evaluieren. Für die Evaluierung wird die Gesamtmenge der Kameraposen $\{T_i\}$ mit $i = 0, ..., N-1$ einbezogen, wo N die Anzahl der bearbeiteten Frames ist. Der Rotationsanteil R_i und der Translationsanteil $\mathbf{t}_i$ (siehe 2.3) der jeweiligen Posenmatrix werden separat analysiert.

4.3.1.1 Erstellung einer Referenz

Im wesentlichen wird in dieser Arbeit das Verfahren aus [LKS+17] angewendet, bei dem ein Referenzkreis mittels Fitting erstellt wird. Die Vorgehensweise dafür kann wie folgt beschrieben werden:

1. Schätzung der Drehebene E^{rot}:
   (a) Erstelle Positionsmatrix $P = \begin{Bmatrix} \mathbf{p}_0^T & 1 \\ ... & \\ \mathbf{p}_{N-1}^T & 1 \end{Bmatrix}$ mit $\mathbf{p}_i = \mathbf{t}_i$,
   (b) Wende Singulärwertzerlegung $P = USV^T$ an, mit $V = \begin{Bmatrix} \mathbf{v}_0 & ... & \mathbf{v}_3 \end{Bmatrix}$,
   (c) Setze Ebenennormale $n = \mathbf{v}_j$, so dass $s_{j,j} = \arg\min(\text{diag}(S))$.
2. Ausrichtung der Drehebene mit der XY-Ebene in Weltkoordinaten:
   (a) Berechne Rotationsmatrix $R^{\hat{\mathbf{n}}->\mathbf{Z}}$ mit $\mathbf{Z} = (0, 0, 1)^T$,
   (b) Rotiere $\mathbf{p'}_i = R^{\hat{\mathbf{n}}->\mathbf{Z}}\,\mathbf{p}_i$, $i = 0, ..., N-1$.
3. Fitting eines Kreises in der XY-Ebene anhand von $\{\mathbf{p'}_i\}$
4. Ausrichtung des Kreises mit der ursprünglichen Drehebene: $\mathbf{m}^{rot} = (R^{\hat{\mathbf{n}}->\mathbf{Z}})^T\,\mathbf{m}$, wo $\mathbf{m}$ der Mittelpunkt ist.

Das Fitting wird mit der Funktion `CircleFitByTaubin` von Nikolai Chernov durchgeführt, welche eine MATLAB Implementierung der Fitting-Methode von Taubin [Tau91] darstellt.

4.3.1.2 Fehlermetriken

Zur quantitativen Beschreibung von Translationsfehlern wird der gefittete Kreis
$C = \{\mathbf{c} : \|\mathbf{c} - \mathbf{m}^{rot}\| = r\}$ mit dem Radius r und $\mathbf{m}^{rot}, \mathbf{c} \in E^{rot}$ betrachtet. Dank einer konstanten Framerate und einer festgesetzten Drehgeschwindigkeit konnten Lefloch et al. in der oben zitierten Arbeit für jeden Frame die genaue Referenzposition auf dem Kreis ermitteln. Die euklidische Distanz zwischen dem jeweiligen tatsächlichen Positionsvektor und seiner Referenz diente als Fehlermaß. In unserer Arbeit spielt jedoch die Frage, ob die auf einem mobilen Gerät verfügbaren Hardware-Ressourcen für eine Echtzeitrekonstruktion mit dem vorliegenden Framework ausreichend sind, eine zentrale Rolle. Aus diesem Grund ist es wichtig, Tests mit reellen Frameraten durchzuführen. Da unter

diesen Umständen die exakte Referenzposition nicht bekannt ist, wird hier der kürzeste Abstand zum Kreis berechnet. Dabei werden die folgenden Schritte ausgeführt:

1. Projiziere die Kamerapositionen in die Drehebene: $\mathbf{p}_i^{proj} = \mathbf{p}_i - \langle \mathbf{p}_i \, , \, \hat{\mathbf{n}} \rangle \hat{\mathbf{n}}$,
2. Finde $\mathbf{c}_i^{near}$ mit $\|\mathbf{p}_i^{proj} - \mathbf{c}_i^{near}\| = \min_{\mathbf{c} \in C} \{\|\mathbf{p}_i^{proj} - \mathbf{c}\|\}$,
3. Berechne Distanz $\mathrm{d}(\mathbf{p}_i, C) = \|\mathbf{p}_i - \mathbf{c}^{near}\|$.

Die Beschreibung der Orientierungsfehler erfolgt auf der Grundlage der relativen Rotationen zwischen der ersten und der aktuellen Pose $R_i^{\Delta} = (R_0)^T R_i$ (normalerweise ist $R_0 = Id$). In [LKS+17] werden an der Stelle Differenzen zwischen dem tatsächlichen Rotationswinkel und dem Bezugswert, der sich aus der Framenummer ergibt, analysiert. Aus dem selben Grund wie bei der Translationsevaluierung, fehlt in unserem Fall auch hier eine genaue Referenz. Stattdessen betrachten wir den Winkel zwischen der Normale der Drehebene und der jeweiligen Rotationsachse der Kamera, was ermöglicht, die Form der Rotation zu evaluiert. Die Fehlerberechnung für den Frame i läuft nach dem unten stehenden Algorithmus ab:

1. Transformiere R_i^{Δ} in die Achsen-Winkel-Darstellung mit der Rotationsachse $\hat{\mathbf{r}}_i$ und Winkel α_i,
2. Berechne $\phi = \arccos(\langle \hat{\mathbf{n}} \, , \, \hat{\mathbf{r}}_i \rangle)$.

Nach der Berechnung der Per-Frame-Fehler werden für die beiden Fehlertypen Erwartungswert $\hat{\mu}^{error}$ und Standardabweichung $\hat{\sigma}^{error}$ geschätzt. Diese statistischen Parameter werden anschließend zum Vergleich unterschiedlicher Testkonfigurationen einbezogen.

Zusätzlich zu den oben erläuterten Statistiken werden bei der Bewertung der Ergebnisse bestimmte Eigenschaften des Versuchsaufbaus zur Hilfe genommen. Insbesondere sind das der Abstand zwischen der Kamera und dem Zentrum des Drehtellers und der Winkel zwischen der Frontalebene der Kamera / des Tablets und der Drehebene (siehe 5.3). Diese werden mit dem Kreisradius r bzw. dem Winkel $90° - \arccos(\langle \hat{\mathbf{n}} \, , \, (0,1,0)^T \rangle)$ verglichen. Da es sich um keinen kalibrierten Aufbau handelt, dient dieser Vergleich keiner quantitativen Evaluierung, trägt jedoch zu einer besseren Interpretierbarkeit der Daten bei und ermöglicht, gröbere Fehler direkt zu erkennen.

4.3.2 Einfluss der Initialisierung von ICP mit VIO-Daten auf seine Konvergenz

Zur Laufzeit des Programms wird ein Protokoll geführt, in dem für den jeweiligen Frame die Anzahl der ICP-Iterationen auf jeder Pyramidenebene und die Angabe, ob nach dem Erreichen einer Abbruchbedingung eine zuverlässige Transformation vorliegt, gespeichert werden. Zur Bewertung der Konvergenzgeschwindigkeit der kombinierten Tracking-Variante werden diese Protokolle bearbeitet, indem der Erwartungswert der Iterationenanzahl und die Standardabweichung je Pyramidenebene geschätzt werden. Zusätzlich ist die Anzahl der Frames zu ermitteln, bei denen die Posenberechnung gescheitert ist. Diese Werte sollen dann den entsprechenden Ergebnissen bei der ursprünglichen ICP-Variante gegenübergestellt werden.

Kapitel 5

Versuchsaufbau und -durchführung

In diesem Kapitel werden die Hard- und Softwarekomponenten des Testsystems sowie der genaue Aufbau und Ablauf der Versuche beschrieben.

5.1 Hardware

Wie es im Kapitel 4 bereits erwähnt wurde, besteht die Hardware unseres Testsystems aus einem Project Tango Tablet Development Kit sowie einem CamBoard pico flexx.

Tablet

Das Tango Tablet verfügt über einen NVIDIA Tegra K1 Prozessor, was unter anderem das Programmieren mit CUDA (siehe 5.2) ermöglicht. Zu seinen weiteren besonderen Merkmalen gehören eine *Structured Light* RGB-IR-Kamera und eine Fischauge-Farbkamera. Die Tango-Tiefenkamera arbeitet im Bereich von 0,5 bis 4 m und liefert Daten mit einer Frequenz von 5 Hz. Die Fischauge-Kamera wird zusammen mit der IMU für Bewegungs-Tracking eingesetzt; die Aktualisierung der Posenschätzung erfolgt mit 30 Hz. Der Tablet-Bildschirm ist 7,02 Zoll groß. Als Betriebssystem ist Android 4.4.2 installiert.

Tiefenkamera

Das CamBoard pico flexx stellt eine TOF-Tiefenkamera dar. Die Auflösung des Sensors beträgt 224x171 Pixel. Die Kamera kann in sechs verschiedenen Modi (*Use Case* in der Terminologie des Herstellers) betrieben werden. Je nach der eingestellten Betriebsart reicht ihr Arbeitsbereich von 0.3 m bis 4 m. Die Aktualisierungsrate ist ebenso vom Modus abhängig und variiert zwischen 5 fps bei der größten Reichweite und 45 fps beim Scannen mit kürzesten Abständen. Als Default-Einstellung verwenden wir den Use Case für 3D Objektrekonstruktion [1], der einen Arbeitsbereich von 0,5-1,5 m und eine Framerate von 15 fps vorsieht.

Das Tango Tablet wird in den Versuchen in seiner Dockingstation aufgestellt. Das CamBoard pico flexx wird über einen USB-Anschluss an die Dockingstation des Tablets angeschlossen und am oberen

[1] Hier Objektrekonstruktion als Rekonstruktion von kleineren Oberflächen im Gegensatz zu Raumrekonstruktion.

Bildschirmrand mittels einer Halterung so befestigt, das es annähernd über der Tango-RGB-Kamera liegt.

5.2 Software

3D Rekonstruktionsframework

Das Rekonstruktionsframework ist in C++ implementiert. Für die Erstellung der grafischen Oberfläche sowie für einige weiteren Hilfsfunktionen wird das Qt Framework verwendet. Die eigentliche Rekonstruktionspipeline ist in CUDA implementiert, was parallele Ausführung vieler Berechnungsschritte auf der GPU, insbesondere aller Per-Pixel-Operationen, ermöglicht. Das Rendering ist mit Hilfe von OpenGL/GLSL realisiert. Für lineare Algebra und numerische Operationen verwendet man die Bibliothek Eigen.

Bei der Portierung konnte die CUDA-Implementierung dank des Vorhandenseins eines Nvidia-Graphikprozessors übernommen werden; der Gebrauch von OpenGL wurde an die ES-Spezifikation angepasst.

Tango-API

Neben der Hardware bilden auch Bibliotheken für 3D Scannen und Bewegungstracking ein Teil der Tango-Plattform. Für Zugriff auf die entsprechenden Funktionalitäten aus einer Anwendung steht ein API zur Verfügung. Das Abrufen der Daten aus Tango-Tracking kann entweder über eine Callback-Funktion oder on demand als Pose zum vorgegebenen Zeitstempel realisiert werden, wobei in unserem System die letztere Variante verwendet wird. Eine solche Pose stellt Orientierung und Position des Tablets relativ zu seinen Koordinaten beim Start des Dienstes dar. Außerdem besitzen einzelne Sensoren wie IMU, Tiefen- und Farbkamera ihre eigene Koordinatensysteme, und interne Transformationen zwischen diesen und den Gerätekoordinaten sind ebenso abrufbar.

Royale-API

Dieses Programmierinterface ermöglicht die Ansteuerung des CamBoard pico flexx und Abfrage der relevanten Daten. Der Zugriff auf Bilddaten erfolgt über callback-basierte Schnittstellen. Im Kontext der vorliegenden Arbeit wurden Tiefenbilder als Eingabe für die Rekonstruktionspipeline sowie Infrarotbilder zu Kalibrierungszwecken benutzt. Mit Hilfe des APIs ist zudem möglich, den aktuellen Betriebsmodus zu setzen.

OpenCV

Zur Kalibrierung der externen Tiefenkamera gegen das Tablet wurden Funktionen aus der OpenCV-Bibliothek verwendet. Mehr dazu siehe im Kapitel 4.1.3.

5.3 Versuchsaufbau

Zum Durchführen der Versuche wird das Tablet samt der montierten Tiefenkamera auf einem Drehteller aufgestellt. Die Position liegt vom Zentrum des Drehtellers zum Rand versetzt, so dass der Abstand zwischen der Kamera und der Rotationsachse ca. 20,5 cm beträgt. Die x-Achse des Tablets ist senkrecht zum Drehradius ausgerichtet. Durch die Form der Dockingstation entsteht zwischen der Frontalebene des Tablets / der Kamera und dem Drehteller ein Winkel von ca. 69° (siehe Abbildung 5.2).

Vor dem Drehteller wird eine einfache Szene aus Keramikfiguren und Kunststoffwürfeln mittlerer Größe sowie einem größeren Holzelement aufgestellt (siehe Abbildung 5.1). Die Objekte stehen in einem Bereich von ca. 60 bis 100 cm von der Kamera entfernt. Um das Tangos merkmalspunktbasierte Tracking zu erleichtern, werden an der Wand im Hintergrund Poster mit Textur aufgehängt.

Der Antrieb des Drehtellers erfolgt mittels eines Schrittmotors in Verbindung mit einer Nanotec SMCI33 Steuerung. Letztere wird über USB an einen PC angeschlossen: auf diese Weise kann man über Nano Pro Software die relevanten Parameter wie Drehwinkel und -geschwindigkeit einstellen und Bewegung starten. Während der Versuche rotiert sich der Drehteller um 45 Grad. Beim Radius $r \approx 20,5$ cm läuft damit die Kamera einen Weg $s = 2\,r\,\pi\,45°/360° \approx 16,1$ cm. Für die meisten Versuche setzen wir den Sollwert der Motorgeschwindigkeit auf 50 Hz, was zusammen mit der Auflösung von 0,028125 Grad/Schritt eine Laufzeit $t = 32$ s und Geschwindigkeit $v \approx 5 \times 10^{-3}$ m/s ergibt. Dieser Wert wurde experimentell so ermittelt, dass die beim Vorbeifahren der Kamera sichtbaren Oberflächen nahezu vollständig rekonstruiert werden, damit man die Qualität der Ergebnisse auch visuell beurteilen kann. Bei den Tests mit dynamischer Bewegung (siehe Abschnitt 4.2.3) verwenden wir die 4-fache und die 8-fache Geschwindigkeit $v \approx 0,02$ m/s bzw. $v \approx 0,04$ m/s.

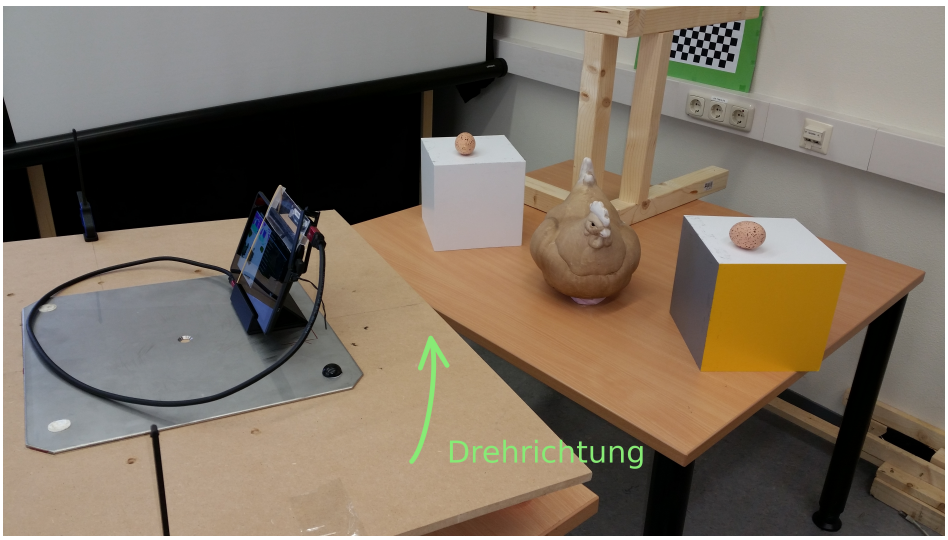


Bild 5.1: Versuchsaufbau: das Tablet mit der montierten Kamera auf dem Drehteller vor der Szene.

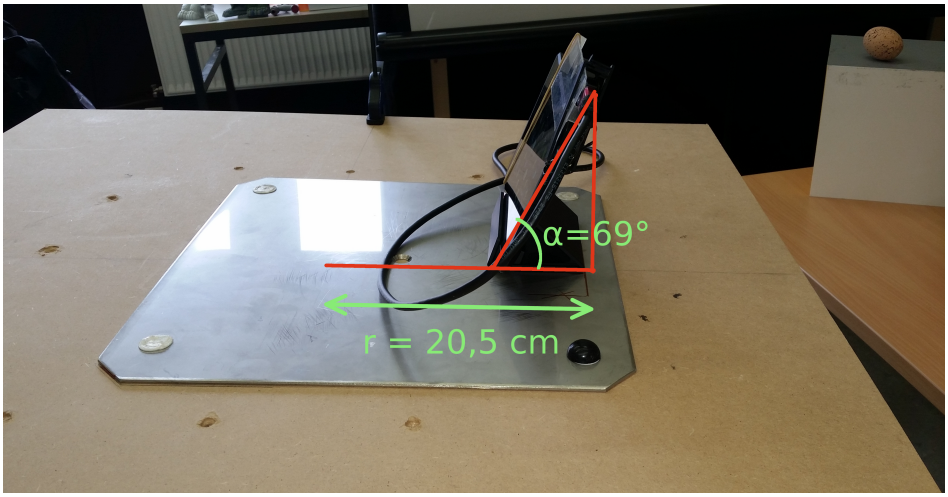


Bild 5.2: Versuchsaufbau: Drehradius und Winkel zur Drehebene.

Kapitel 6

Ergebnisse

6.1 Genauigkeit des ICP-basierten Kamera-Trackings

6.1.1 Basiskonfiguration

Als erstes betrachten wir die Testergebnisse, die mit der Basiskonfiguration erhalten wurden. Aus den im nächsten Abschnitt erklärten Gründen wurde diese Konfiguration in drei Versuchen mit derselben Szene und Drehgeschwindigkeit verwendet. In der unten stehenden Beschreibung werden die statistischen Werte genannt, die über die gesamten Daten aller drei Versuche erhobenen wurden; die Ergebnisse einzelner Versuche findet man in der Tabelle 6.1.2, die Kamerapositionen zusammen mit den gefitteten Kreislaufbahnen sind in der Abbildung 6.1 visualisiert. Hier und im Weiteren sind Kamerapositionen blau, Kreise schwarz und Richtungen des kürzesten Abstands zum Kreis rot dargestellt. Die Kreise und die Abstandsvektoren können je nach Vergrößerungsgrad unsichtbar sein.

Die geschätzte Laufbahn der Kamera weist eine kreisförmige Form auf. Die Positionsfehler mit dem mittleren Abstand zum gefitteten Kreis (siehe Abschnitt 4.3) $\bar{d} \approx 0,5559$ mm und der Standardabweichung $\hat{\sigma^d} \approx 0,6011$ mm lassen sich als ausreichend klein bezeichnen. Der mittlere Kreisradius $\bar{r} \approx 0,1842$ m liegt etwa 2 cm unter dem zu erwartenden Wert von 20,5 cm (siehe Abschnitt 5.3). Dabei ist zu berücksichtigen, dass das Fitting anhand eines Kreisbogens erfolgt und bei einer vollständigen Kreisbewegung der Kamera eine präzisere Radiusschätzung zu erwarten ist.

Im Kontrast zu den relativ geringfügigen Positionenfehlern, sind bei den Kameraorientierungen erhebliche Abweichungen von der festgesetzten Referenz – der Normalen der Drehebene – zu beobachten; genauer gesagt, beträgt der Mittelwert des Winkels zwischen der Kamerarotationsachse und der Normale $\bar{\phi} \approx 15,6176°$ mit der Standardabweichung $\hat{\sigma^\phi} \approx 4,7476$. Es ist jedoch zu beachten, dass das Fitting der Drehebene auf der Grundlage der Positionen durchgeführt wird: infolgedessen kann die Genauigkeit der Positionsschätzung die Referenz für die Evaluierung der Orientierungen und damit auch Ergebnisse dieser Evaluierung beeinflussen. Bei den weiteren Überlegungen wird der erste Versuch mit der Basiskonfiguration (Basis1) beispielhaft analysiert. Die Folgerungen sind auch auf die restlichen Versuche übertragbar, da die hier behandelten Größen dort sehr ähnliche Werte

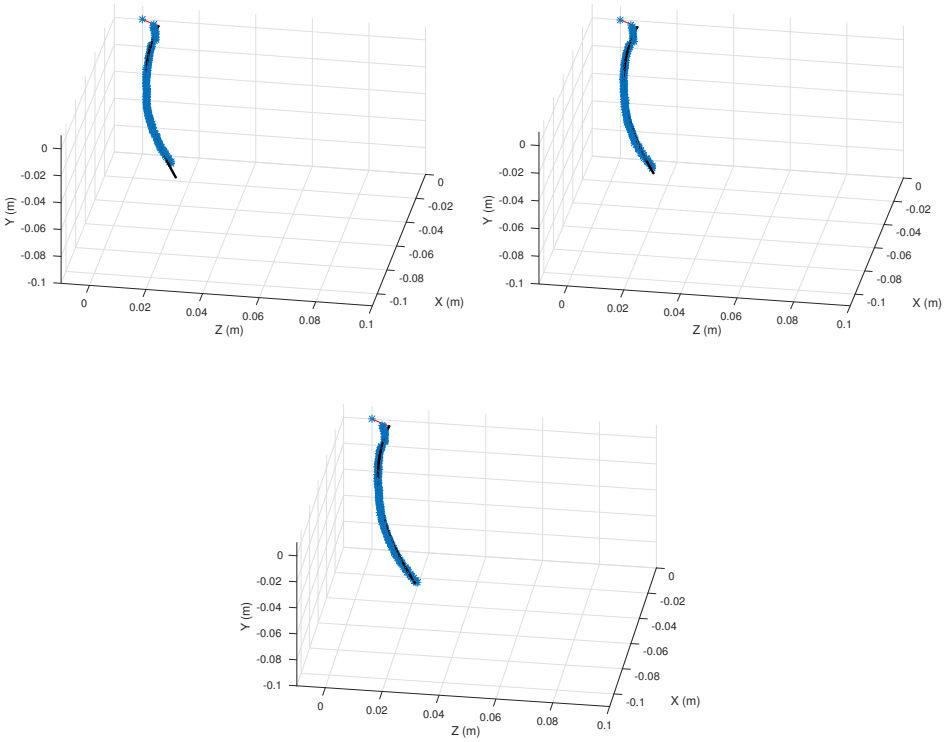


Bild 6.1: ICP Basis1-3: Kameralaufbahn

annehmen.

Man betrachte die Werte $\bar{\phi}$, $\hat{\sigma^{\phi}}$. Die relativ zu dem Winkelfehler kleine Standardabweichung lässt darauf schließen, dass $\bar{\phi}$ eine Art Offset zu dem Sollwert darstellt [1]. Auch die Winkel-Zeit-Kurve in der Abbildung 6.2 zeigt, dass der Fehler, nach einer kurzen Einpendelungsphase, nahezu konstant bleibt. Im Abschnitt 5.3 wurde beschrieben, dass das Tablet in einem Winkel $\alpha \approx 69°$ zu der Drehebene steht. Anhand der geschätzten Ebenennormale ergibt sich jedoch für die Tracking-Daten ein Winkel $\beta \approx 56°$ (siehe Tabelle 6.1.2). Vergleicht man die Differenz mit dem Orientierungsfehler, so erhält man $\phi \cong \alpha - \beta$. Damit ist anzunehmen, dass der Betrag von ϕ hauptsächlich aus einer inkorrekten Neigung der gefitteten Drehebene und nicht bzw. weniger aus einer falschen Orientierungsschätzung hervorgeht.

[1] Der Wert von $\hat{\sigma^{\phi}}$ wird im Wesentlichen von einem Ausreißer im ersten Frame beeinflusst. Entfernt man diesen, sinkt die Standardabweichung auf $\hat{\sigma^{\phi}} \approx 2,3566$.

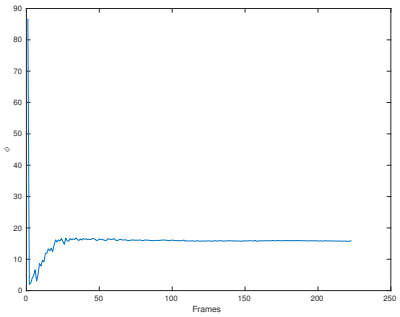


Bild 6.2: Zeitlicher Verlauf des Orientierungsfehlers ϕ in Basis1.

Zur Prüfung der oben formulierten Annahme berechneten wir den Orientierungsfehler bezüglich einer simulierten Normalen $\hat{\mathbf{n}}^s$. Zur Ermittlung der letzteren wird zuerst die y-Achse um $90° - \alpha = 18°$ um die x-Achse rotiert. Um von α, der für das Tablet gemessen wird, auf den entsprechenden Wert für die Tiefenkamera schließen zu können, wird zusätzlich noch die Kalibrierungsmatrix $C^{D \to RGB}$ (siehe 4.1.3) berücksichtigt. Damit ist die Normale $\hat{\mathbf{n}}^s = (C^{D \to RGB})^{-1} \, R_{x,21°} \, (0, 1, 0, 0)^T$ (woraus sich der korrigierte Kamera-Ebene-Winkel $\alpha' \approx 74°$ ergibt). Nach dem Einsetzen von $\hat{\mathbf{n}}^s$ in die Fehlerfunktion erhält man akzeptable Werte $\bar{\phi}^s \approx 3,7752°$, $\hat{\sigma^{\phi}}^s \approx 7,3127$, die für die obige Annahme sprechen.

Beim Betrachten der Form der Laufbahnen ist eine Art Knick am Anfang der Aufnahme in allen drei Versuchen zu sehen. Die Abbildung 6.3 stellt den initialen Teil der Laufbahn aus dem ersten Versuch noch einmal vergrößert dar. Während der ersten Frames gleitet die Kamera vom zu erwartenden Pfad nach unten (negative y-Richtung) und nach hinten (positive z-Richtung) ab, wodurch auch gröbere Positionsfehler entstehen. Anschließend stabilisiert sich die Posenschätzung, so dass die Kamerapositionen fast ideal auf einer Kreislinie liegen. Diese Linie verläuft jetzt jedoch versetzt bezüglich der tatsächlichen Laufbahn. Es lässt sich annehmen, dass der initiale Knick die erwähnte falsche Neigung der Drehebene und die Verkürzung des Radius zur Folge hat. Dies steht auch in Übereinstimmung mit dem Verlauf des Orientierungfehlers, welcher, nach einer Steigung in der Knick-Phase, weiter nahezu konstant bleibt.

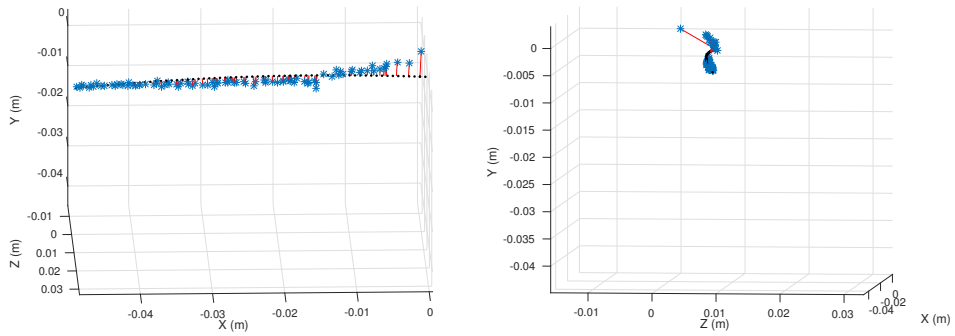


Bild 6.3: Fehler beim Starten der Kamerabewegung: die Laufbahn gleitet während der ersten Frames entlang der y- und z-Achse ab. Links: XY-Ansicht, rechts: YZ-Ansicht.

Zusammenfassend kann gesagt werden, dass die Schätzung der Kameraposition mit ICP die reelle kreisförmige Bewegung insgesamt gut und nur mit minimalen Fehlern reproduzieren konnte. Die approximierte Kreisebene der Kamera lag dabei um einen Winkel von ca. 15 Grad versetzt relativ zu der tatsächlichen Drehebene. Da dieses Phänomen in allen Basis-Versuchen und, wie aus der Abbildung 6.4 und Tabelle 6.1.2 zu sehen ist, bei den meisten anderen Parameterkonfigurationen mit nahezu konstanten Fehlergrößen zu beobachten ist, handelt es sich offensichtlich um einen systematischen Fehler. Es ist als Ursache in Erwägung zu ziehen, dass die in der Szene vorhandenen geometrischen Details dem ICP-Algorithmus nicht ausreichend Information für eine präzisere Posenschätzung liefern. Gegen das als alleinige Erklärung spricht jedoch der beinahe symmetrische Aufbau der Szene, während die Fehler speziell am Anfang der Rekonstruktion auftreten. Mehr zu diesem Thema siehe im Abschnitt 6.4.2.

6.1.2 Weitere Konfigurationen

Die unten stehende Tabelle 6.1.2 enthält eine Übersicht über die Fehlerstatistiken aller getesteten Konfigurationen des ICP-basierten Kamera-Tracking. In der Abbildung 6.4 sind noch zusätzlich die geschätzten Kamerapositionen samt der gefitteten Kreislaufbahn für den jeweiligen Versuch dargestellt. Wie es diesen Quellen zu entnehmen ist, weisen zwei weitere Konfigurationen – *Gewichtung mit Konfidenzwerten aus dem PMD-API* und *ICP-Bildpyramide mit 2 Ebenen* – keine deutliche Unterschiede in der Genauigkeit des Trackings relativ zu den Basiseinstellungen auf. Die Differenz zwischen dem mittleren Positions- bzw. Orientierungsfehler in der Basiskonfiguration und den entsprechenden Werten in den beiden erwähnten Konfigurationen ist erheblich kleiner als die Standardabweichung $\hat{\sigma^d}$ bzw. $\hat{\sigma^\phi}$ und ist damit eher der Zufallsverteilung der Ergebnisse als dem Einfluss des jeweiligen Parameters zuzuschreiben. Im Bezug auf die im Abschnitt 4.2.1 aufgestellten Hypothesen lässt sich Folgendes sagen:

- Bei einer Verkürzung der **Bildpyramide** auf **2 Ebenen** ist kein – und insbesondere **kein negativer** – **Einfluss** auf die **Präzision** des Trackings festzustellen;

- Die **Gewichtung** der Eingabepunkte mit den **Konfidenzen** aus dem **Kamera-API** zeigt **keine Erhöhung** der **Tracking-Präzision** relativ zu dem generischen Gewichtungsschema.

Tabelle 6.1: ICP Tracking

	Basis1	Basis2	Basis3	Krümmung	Pmd-Konfidenz	2 Ebenen
Positionsfehler, mm	0,5701 ± 0,5893	0,5204 ± 0,5977	0.577 ± 0,6159	0,7428 ± 0,5646	0,544 ± 0,5697	0,5531 ± 0,6
Orientierungsfehler, °	15,7029 ± 5,3286	15,7895 ± 4,3107	15,3733 ± 4,5854	8,4927 ± 1,2631	15,2854 ± 1.5706	15,5454 ± 2,9712
Drehradius, m	0,1912	0,1859	0,1756	0,1763	0,17036	0,1826
Winkel zur Drehebene, °	56	56	56	64	56	56

Die ICP-Variante mit den aktivierten Krümmungs-Features hebt sich bei den Fehlerstatistiken von den anderen Testkonfigurationen etwas ab. Der mittlere Positionsfehler steigt, befindet sich jedoch immer noch in einem akzeptablen Bereich unter 1 mm. Als Ausgleich dazu liegt die gefittete Drehebene in einem Winkel zu der vertikalen Kameraachse, der dem zu erwartenden Richtwert deutlich näher ist; infolgedessen ist auch der Orientierungsfehler deutlich kleiner als in den anderen Versuchen. Dies kann mit dem geringeren initialen Knick der Laufbahn in Verbindung gebracht werden (siehe Abbildung 6.4). Geht man vom Ziel aus, die **beiden** Fehlerarten möglichst minimal zu halten, erweist sich diese Konfiguration als optimaler Kompromiss. Damit ließ sich die Hypothese, dass die Krümmungsinformation unter der gegebenen Auflösung zu keiner Verbesserung der Tracking-Ergebnisse führen wird, nicht bestätigen. An der Stelle soll aber darauf hingewiesen werden, dass die Ursache des systematischen Fehlers, der eine zusätzliche Neigung der Drehebene verursacht, im Moment unklar ist. Damit kann eine endgültige Aussage über den Einfluss der Krümmungsinformation erst nach der Erklärung (und ggf. Behebung) des Fehlers getroffen werden.

6.2 Tracking mit ICP vs. Multi-Sensor-Tracking

Während der Vorbereitungsphase stellten wir fest, dass Ergebnisse des internen Tracking-Verfahrens des Tango Tablets bei einer gleich bleibenden Szene / Bewegung stark variieren können. Um diese Tatsache bei der Evaluierung zu berücksichtigen und eine quantitative Aussage über die Stabilität des Tango-Trackings zu erhalten, entschieden wir, für das kombinierte Tracking drei Testversuche unter denselben Bedingungen durchzuführen. Zusätzlich sollte auch der Versuch mit der Basiskonfiguration des ICP-basierten Trackings (siehe Abschnitt 6.1.1) dreimal wiederholt werden, um seine Ergebnisse zum Vergleich heranzuziehen.

Die Abbildung 6.5 stellt die durch das Fitting geschätzte Kreislaufbahnen der Kamera in diesen sechs Versuchen dar. Während die drei Ergebnisse von ICP dicht bei einander liegen ($\bar{r} =$

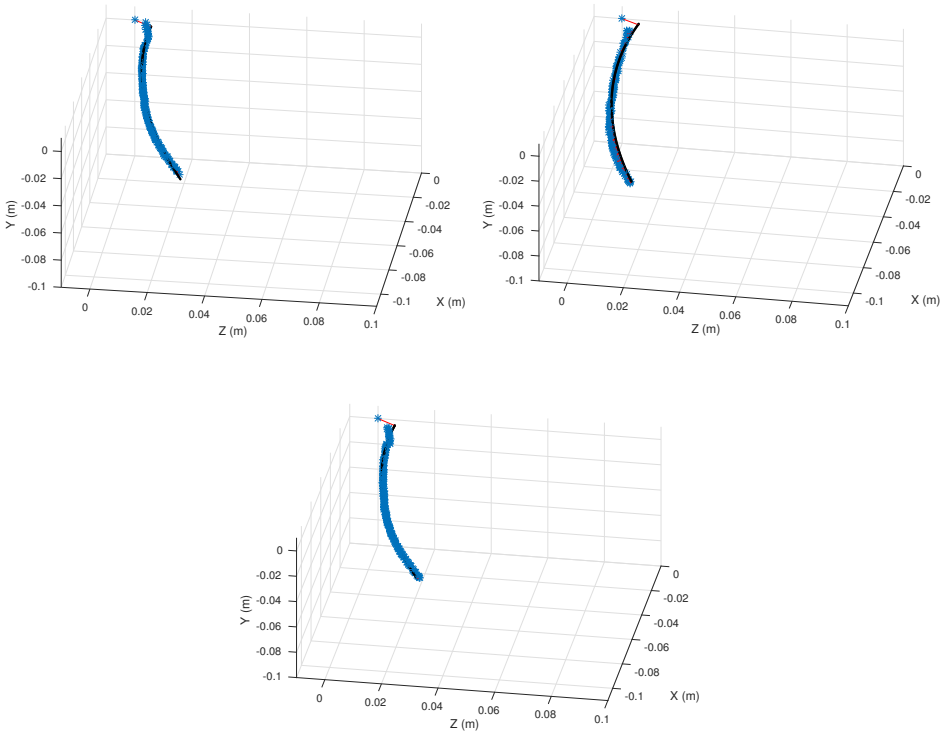


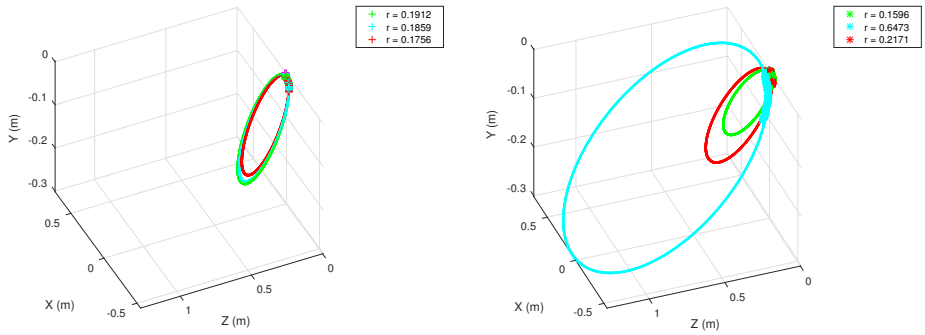
Bild 6.4: ICP mit zusätzlichen Parametern: Kameralaufbahn. Oben links: 2-Ebenen-ICP, oben rechts: Krümmung, unten: Pmd-Konfidenzen.

$0,1842m$, $\hat{\sigma^r} = 0,0079$), ist bei den Kreisen von Tango-VIO eine beträchtliche Varianz zu sehen ($\bar{r} = 0,3413m$, $\hat{\sigma^r} = 0,2665$). Dabei liegt nur eine Tango-Laufbahn mit dem Radius $r \approx 0,2171$ m dem gemessenen Richtwert von 20,5 cm ausreichend nah. Es ist zu betonen, das die durchgeführten Versuche und die vorliegende Ergebnisinterpretation sich nicht als eine Evaluierung des Tango-Trackings verstehen. Dies würde weitere Tests unter verschiedenen Bedingungen erfordern und den Rahmen unserer Arbeit sprengen. Es ist folglich vorstellbar, das dieses Verfahren unter anderen Konstellationen (Szeneaufbau, Bewegungscharakteristika) bessere Resultate liefern kann. Das Problem scheint jedoch nicht für unseren Fall und das uns verfügbaren Gerät spezifisch, sondern auch anderen Tablet-Nutzern bekannt zu sein.[2]

Im Weiteren werden die Fehlerstatistiken aus den Versuchen mit dem kombinierten Tracking ausfürlicher dargelegt. Zuerst betrachten wir nur Daten des visuell-inertialen Verfahrens. Anschließend wird analysiert, welche Wirkung das Initialisieren mit einer VIO-Kamerapose auf das Endergebnis und die Konvergenz von ICP ausübt.

Bis auf einen Ausreißer im zweiten Versuch liegen die Positionsfehler der VIO-Schätzungen auf dem gleichen Niveau wie beim ICP-basierten Tracking (siehe Tabelle 6.2). Der Orientierungsfehler

[2]Sieh z.B. Forumdiskussion `http://stackoverflow.com/questions/30876433/project-tango-pose-data-producing-drift-while-stationary-and-in-motion`, abgerufen am 22.12.2016.



(a) ICP (b) Tango

Bild 6.5: Gefittete Kamerakreislaufbahnen aus ICP und Tango-VIO

verhält sich jedoch anders als beim letztgenannten Verfahren. Zum einen ist er deutlich niedriger, zum anderen scheinen auch größere Ungenauigkeiten in der Positionsschätzung ihn nicht zu beeinflussen, so dass in allen Versuchen sehr ähnliche Werte ermittelt wurden. Passend zu diesem Bild liegt auch der Winkel zwischen der vertikalen Achse und der Drehebene nah dem gemessenen Wert. Damit wirken sich Positionsfehler beim Tango-Tracking vor allem auf die Radiusschätzung, jedoch nicht bzw. nur geringfügig auf die Neigung des gefitteten Drehebene aus.

Insgesamt lieferte das Tango-Tracking nur im Versuch 3 eine akzeptable Posenschätzung, diese erweist sich aber als das ausgeglichenste Resultat unter allen Tests, wenn man alle Parameter – Positions- und Orientierungsfehler, Drehradius, Winkel zur Ebene – in Betracht zieht. Der Versuch 1 soll aufgrund eines zu kleinen Radius, trotz seiner günstigen Fehlerstatistik, aussortiert werden.

Tabelle 6.2: Multi-Sensor-Tracking

	VIO 1	VIO+ICP 1	VIO 2	VIO+ICP 2	VIO 3	VIO+ICP 3
Positionsfehler, mm	0,5058 ± 0,39	0,5382 ± 0,5680	4,5338 ± 3,484	0,5366 ± 0,5939	0,6603 ± 0,4984	0,5677 ± 0,5573
Orientierungsfehler, °	2,1302 ± 0,3275	15,758 ± 2,3694	3,1857 ± 0,2668	11,7201 ± 3,6093	4,6591 ± 0,253	12,7469 ± 5,6264
Drehradius, m	0,1596	0,1969	0,6473	0,1953	0,2171	0,1890
Winkel zur Drehebene, °	72	56	77	60	70	58

Die Endergebnisse des Multi-Sensor-Trackings weisen weder statistisch (Tabelle 6.2) noch visuell (Abbildung 6.6) nennenswerte Unterschiede zu dem rein ICP-basierten Verfahren auf. Die Initialisierung von ICP mit einer Pose aus VIO scheint damit keinen wesentlichen Einfluss auf die Präzision der Posenschätzung zu haben, was zwar für die Stabilität des Algorithmus spricht, das Kombinieren

der beiden Verfahren in der hier verwendeten Form jedoch als wenig sinnvoll erscheinen lässt. Im Versuch 2 gelingt dem ICP-Algorithmus sogar gröbere Fehler in der Tango-Pose zu kompensieren und auf seine „herkömmliche“ Lösung zu kommen, obwohl die gesuchte Transformation von der initialen relativ weit weg liegt (zum Vergleich der initialen und endgültigen Positionen siehe auch Abbildung 6.7). Einen interessanten Fall stellt der Versuch 3 dar. Wenn in den zwei ersten Versuchen eine „schlechte Initialisierung“ den Algorithmus nicht zum Scheitern brachte, ist hier festzustellen, dass auch umgekehrt, eine ausreichend präzise initiale Pose zu keiner Verbesserung des Endergebnisses führt. Insbesondere konnte der oben beschriebene initiale Knick der Kameralaufbahn durch die Initialisierung mit VIO-Daten nicht verhindert werden. Da die VIO-Posenschätzung aus diesem Versuch die tatsächliche Kamerabewegung am ehesten approximiert, ist ihre Visualisierung zusammen mit den ICP-Daten zum Illustrieren der Laufbahnversetzung in den letzteren besonders aufschlussreich (siehe Abbildung 6.8).

Tabelle 6.3: Mittlere Anzahl der ICP-Iterationen auf der Hierarchieebene l

	Basis1	Basis2	Basis3	Krümmung	Pmd-Konfidenz	2 Ebenen
l = 2	2,7040 ± 0,6313	2,7026 ± 0,6047	2,6971 ± 0,6221	3,6096 ± 0,6235	2,705 ± 0,5827	0 ± 0
l = 1	1,95067 ± 0,3036	1,8922 ± 0,3375	1,8797 ± 0,3623	3,0439 ± 1,023	1,8620 ± 0,4430	2,387 ± 0,5186
l = 0	1,2556 ± 0,4372	1,3233 ± 0,4687	1,2697 ± 0,4447	2,5526 ± 1,976	1,2682 ± 0,4439	1,2951 ± 0,4569

	VIO+ICP1	VIO+ICP2	VIO+ICP3
l = 2	3,6313 ± 0,5065	3,9189 ± 0,2737	3,645 ± 0,4797
l = 1	1,8994 ± 0,531	2,7892 ± 1,222	1,75 ± 0,4567
l = 0	1,2682 ± 0,4442	1,4703 ± 0,6997	1,415 ± 0,494

Wie man aus den oben dargelegten Zusammenhängen bereits folgern konnte, führt eine Initialisierung mit der aktuellen VIO-Pose unter diesen Umständen zu keiner Beschleunigung der Konvergenz von ICP. Im Gegenteil, der Algorithmus braucht beim kombinierten Tracking im Durchschnitt mehr Iterationen als bei den anderen Konfigurationen. Am größten ist die Konvergenzverzögerung im zweiten Versuch, in dem Posenschätzungen der beiden Verfahren am weitesten auseinander liegen. (Eine höhere Anzahl der ICP-Iterationen bei der Konfiguration mit den Krümmungs-Features ist offensichtlich durch Berücksichtigung zusätzlicher Informationen / Rahmenbedingungen verursacht. Bei allen Versuchen konnte eine vom Algorithmus akzeptierte Transformation in allen Frames gefunden werden, d.h. die Anzahl der Frames mit gescheiterten Posenschätzungen war überall gleich Null.)

Damit ist festzustellen, dass durch das Kombinieren des ICP-basierten mit dem visuell-inertialen

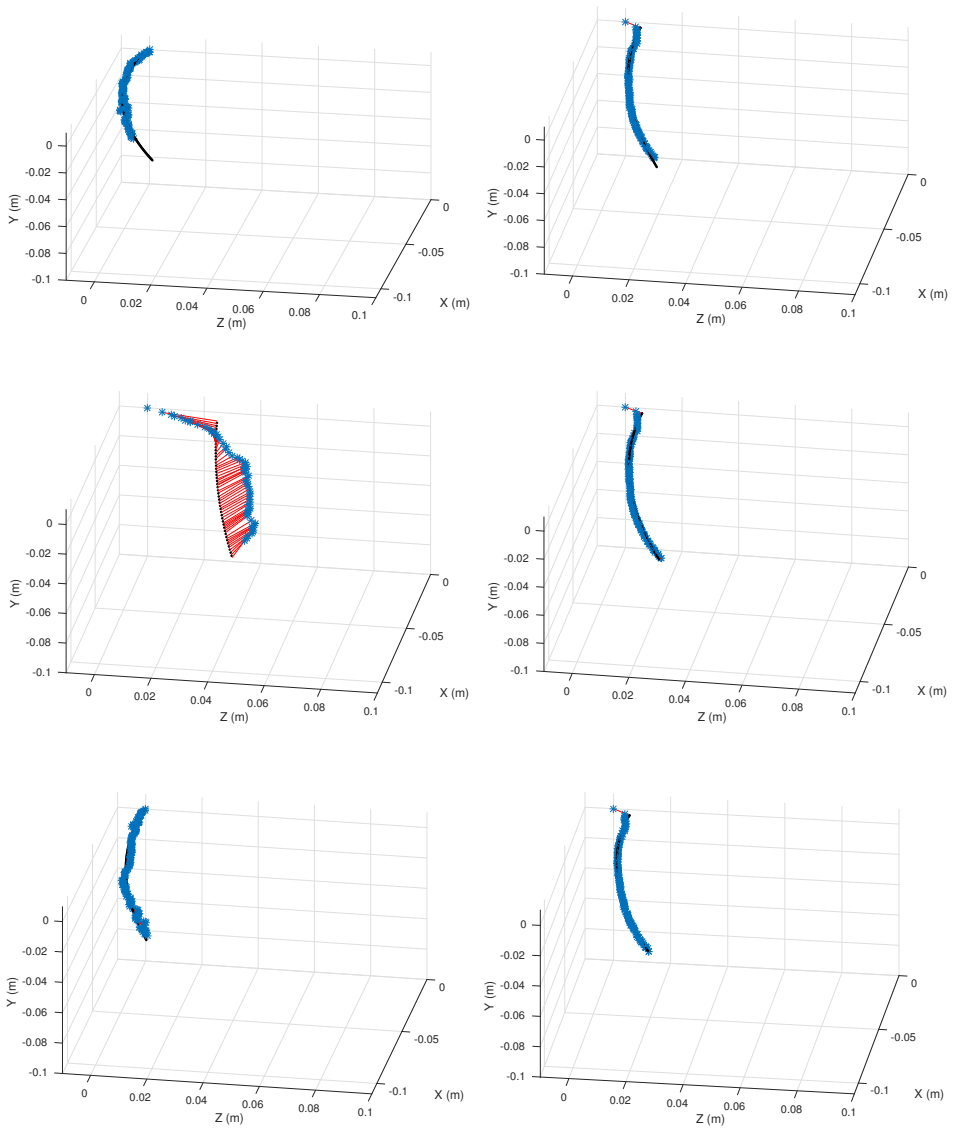


Bild 6.6: VIO und VIO+ICP: Kameralaufbahnen. Links: VIO; rechts: ICP nach der Initialisierung mit der VIO-Pose; von oben nach unten: Versuch 1-3.

Tracking in den durchgeführten Versuchen keine Optimierung des herkömmlichen Verfahrens erzielt werden konnte. Die beiden Hypothesen, die im Abschnitt 4.2.2 formuliert wurden, wurden durch die Testergebnisse widerlegt:

- Das kombinierte Verfahren brachte **keine Erhöhung** der **Präzision** der Posenschätzung und
- führte zu **keiner Beschleunigung** der **Konvergenz** des ICP-Algorithmus.

Zum Teil lag es an einem instabilen Funktionieren von VIO. Gleichzeitig legt aber eine Analyse des Zusammenspiels der beiden Verfahren den Gedanken nahe, dass das Konzept des kombinierten

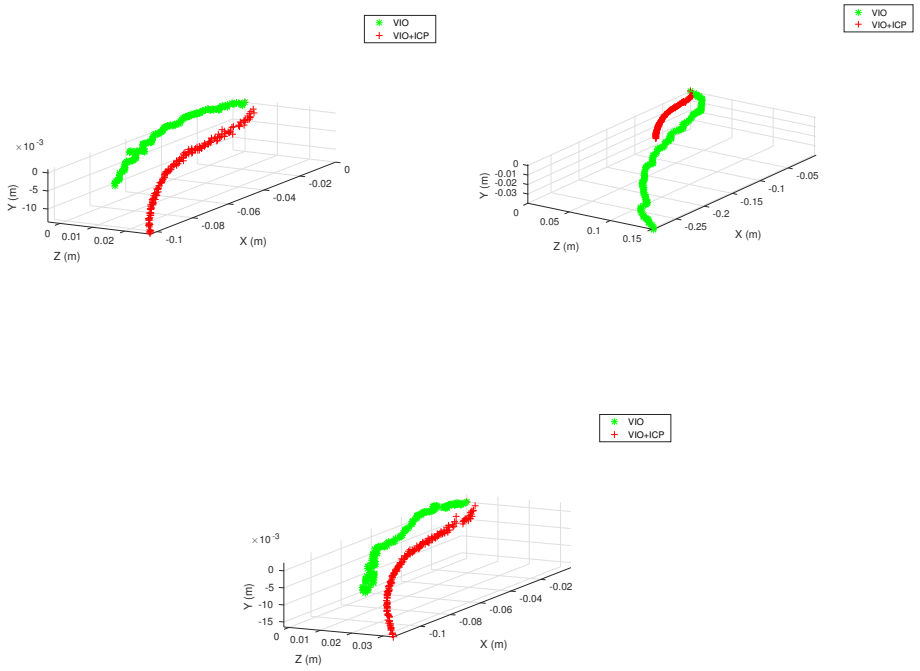


Bild 6.7: VIO und VIO+ICP: Divergenz zwischen der initialen VIO-Pose und der ICP-Lösung. Oben links: Versuch 1; oben rechts: Versuch 2; unten: Versuch 3.

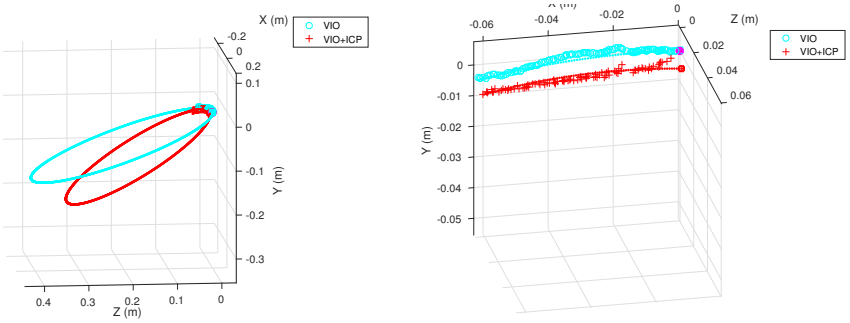


Bild 6.8: Versetzen der ICP-Kameralaufbahn bezüglich der VIO-Ergebnisse im Versuch 3. Links: gefittete Kreise stehen in einem Winkel zu einander; rechts: vergrößerter initialer Teil der Laufbahnen mit dem Abgleiten der ICP-Positionen.

Trackings in unserem System zu revidieren ist. Die angestrebte Beschleunigung der Konvergenz erscheint nur dann erreichbar, wenn sowohl ICP als auch VIO (annähernd) fehlerfreie Posenschätzungen liefern (bzw. ähnliche Fehler machen). Genauer gesagt soll dafür die aktuelle Pose aus VIO näher zur Lösung von ICP (und nicht unbedingt zur tatsächlichen Kamerapose) liegen, als die im letzten Frame berechnete Transformation, was in unseren Tests nicht zutraf.

Da anhand der Testergebnisse kein relevanter Einfluss der Initialisierung auf die Lösung von ICP festgestellt werden konnte, zeigt sich, dass eine Erhöhung der Tracking-Genauigkeit auf diesem Wege kaum möglich ist. (Als Ausnahme kann der Fall betrachtet werden, bei dem die aktuelle Pose – wegen einer dynamischen Bewegung – weit weg von der Pose des letzten Frames liegt, so dass eine korrekte Lösung – und insbesondere eine korrekte Datenassoziierung – mit der Standardinitialisierung nicht mehr möglich ist. Siehe dazu den Abschnitt 6.2.1.) Es lässt sich jedoch darüber spekulieren, dass eine derartige Präzisionssteigerung erfolgen würde, wenn man die Posenschätzungen der beiden Verfahren in einem Filter-Algorithmus fusioniert, statt eine davon (in unserem Fall ICP) als endgültiges Ergebnis zu übernehmen. Das Ziel dabei ist, die Fehler, die in den Test bei den beiden Tracking-Varianten festzustellen waren, durch komplementäre Quellen auszugleichen. Das würde bedeuten, dass ICP auch als Teil eines (ggf. mehrstufigen) Sensor-Fusion-Algorithmus zu betrachten ist. Es ist aber zu bedenken, dass dies mit einem höheren Rechenaufwand verbunden ist, der unter begrenzten Hardware-Ressourcen einen kritischen Faktor darstellt.

6.2.1 Versuche mit erhöhter Geschwindigkeit

Wie es in den Abschnitten 4.2.3 und 5.3 beschrieben wurde, wurden für die Multi-Sensor- und ICP-Basis-Konfiguration noch zusätzlichen Versuche mit vierfacher und achtfacher Drehgeschwindigkeit ausgeführt, um Robustheit des Trackings gegenüber schnelleren Kamerabewegungen zu testen. In der Tabelle 6.2.1 sind Ergebnisse dieser Tests dargestellt. Wie erwartet, wirkt sich ein Anstieg der Geschwindigkeit negativ auf die Genauigkeit der Kameraposenschätzung mit ICP aus. Die Resultate des ersten Versuchs (vierfache Geschwindigkeit) zeigen eine Erhöhung sowohl des Positions- als auch des Orientierungsfehlers. Im zweiten Versuch sinken zwar diese Werte wieder, die Abweichung des geschätzten Drehradius vom Richtwert beträgt jedoch ca. 9 cm und ist damit nicht mehr akzeptabel. Es ist anzunehmen, dass die entstehenden Fehler nicht nur mit den größeren Transformationsschritten zwischen den benachbarten Frames zusammenhängen, sondern auch auf eine mangelhafte Modellinformation zurückzuführen sind, denn bei solchen Geschwindigkeiten konnten deutlich weniger Modellpunkte akkumuliert werden. Tatsächlich bestand das Modell in den Tests mit der Basiskonfiguration und Standardgeschwindigkeit durchschnittlich aus 148656 Punkten, bei der vierfachen Geschwindigkeit waren es 76402 und bei der achtfachen nur noch 59170. Vergleiche visuelle Ergebnisse der Rekonstruktionen in der Abbildung 6.9.

Der Tabelle 6.2.1 ist zu entnehmen, dass eine Initialisierung mit der VIO-Pose hier, im Unterschied zu den Versuchen mit der Default-Geschwindigkeit, die Ergebnisse von ICP deutlich beeinflussen konnte, auch wenn das erzielte nicht dem angestrebten Effekt entsprach. Im ersten Versuch war das visuell-inertiale Tracking unzuverlässig. Trotzdem ist zu bemerken, dass ICP nicht mehr in der Lage war, die groben Initialisierungsfehler zu kompensieren. Im zweiten Versuch hingegen lieferte VIO gute Posenschätzung, das Endergebnis des kombinierten Verfahrens kann jedoch nur bedingt als Verbesserung bezüglich des alleinigen ICP bezeichnet werden: zwar wird der Abstand zum Kreis kleiner und die Radiusschätzung deutlich präziser, so dass diese Werte mit denen aus den Standard-

Tabelle 6.4: Tracking bei höheren Kamerageschwindigkeiten

	4-fache Geschwindigkeit $v \approx 0,02$ m/s			8-fache Geschwindigkeit $v \approx 0,04$ m/s		
	ICP Basis	VIO	VIO+ICP	ICP Basis	VIO	VIO+ICP
Positionsfehler, mm	0,8263 ± 0,7489	1,2384 ± 1,2643	1,1913 ± 0,9649	0,7984 ± 0,7779	0,5495 ± 0,5201	0,5969 ± 0,6584
Orientierungsfehler, °	20,0189 ± 1,2666	14,0852 ± 0,6068	26,0492 ± 3,5772	8,4026 ± 1,3037	2,2361 ± 0,2101	18,4857 ± 2,7725
Drehradius, m	0,1883	0,6911	0,1846	0,2963	0,2257	0,1930
Winkel zur Drehebene, °	52	61	46	64	76	53

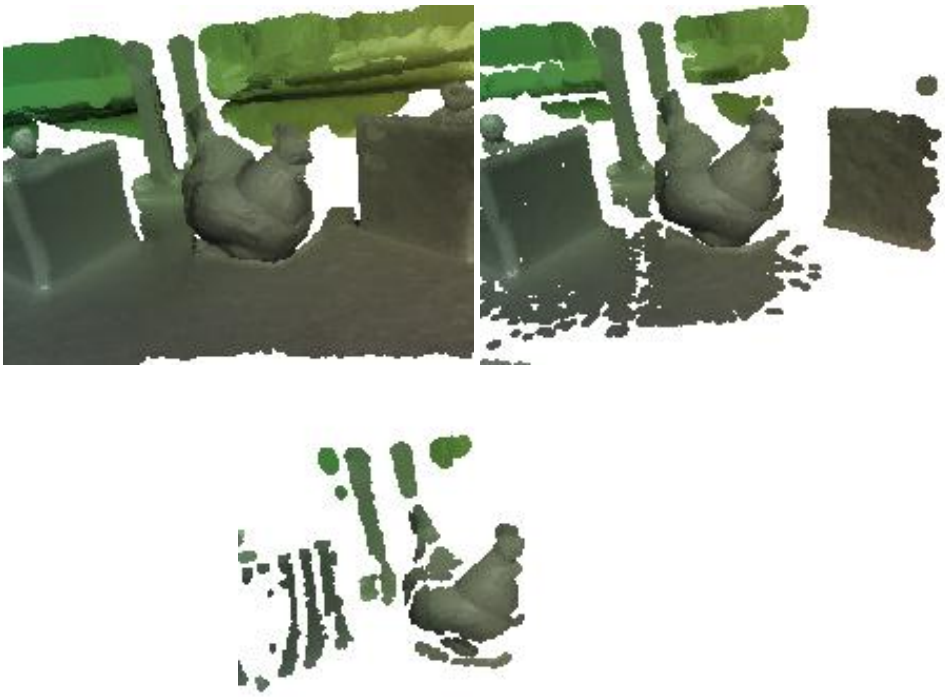


Bild 6.9: Rekonstruktionsergebnisse mit verschiedenen Kamerageschwindigkeiten. Oben links: $v \approx 5 \times 10^{-3}$ m/s; oben rechts: $v \approx 0,02$ m/s; unten: $v \approx 0,04$ m/s.

versuchen vergleichbar sind, aber der Winkel zur Drehebene ist hier zu scharf und liegt vor allem weit weg von dem Initialisierungswert aus VIO. Insgesamt wurden die Ergebnisse aus dem Abschnitt 6.1 / 6.2 auch bei dieser Konfiguration nicht erreicht. Die vermutliche Ursache dafür ist die bereits erwähnte zu niedrige Anzahl der Modellpunkte.

Als eine Schlussfolgerung aus der Evaluierung der Testergebnisse in diesem Abschnitt ist festzustellen, dass, neben größeren Transformationssprüngen zwischen Nachbarposen, auch dünnbesetzte

Modelle einen kritischen Faktor bei schnelleren Kamerabewegungen darstellen. Während die Initialisierung von ICP mit der aktuellen Pose aus einem externen Tracking gegen das erstere Problem hilfreich sein kann, bietet sie für das letztere Problem keine Lösung. Als Abhilfe kann das ICP-basierte Tracking in bestimmten Frames durch die Pose aus einer externen Quelle (z.B. VIO) komplett ersetzt werden (vgl. [NDF14]). Dies setzt einen Entscheidungsalgorithmus voraus, um solche Frames mit unzuverlässigen ICP-Ergebnissen erkennen zu können, was wiederum eine ausgeklügelte Evaluierungsmetrik erfordert. Das entsprechende Entscheidungskriterium in dem vorliegenden Framework ist nicht auf ein kombiniertes Tracking ausgelegt und sieht nur das Ablehnen der zu großen Transformationsschritten vor; es ist jedoch gerade ein der Ziele des kombinierten Verfahrens, solche Schritte zu ermöglichen.

6.3 Performance

Die mittlere Framerate für die jeweilige Testkonfiguration ist in der Tabelle 6.3 zu finden. Die ICP-Basiskonfiguration lief in allen Versuchen mit 8 FPS. Eine niedrigere Framerate im Falle der Krümmungskonfiguration ist auf zusätzliche Rechenoperationen und insbesondere auf eine höhere Anzahl der ICP-Iterationen (siehe Abschnitt 6.2) zurückzuführen. Die Reduzierung der ICP-Hierarchie auf zwei Ebenen und damit eine Verringerung der Gesamtanzahl der Iterationen führte hingegen zu einer Performancesteigerung, welche jedoch nur 1 FPS beträgt. Die ebenso höhere Framerate beim Einsetzen der Konfidenzwerte aus dem Kamera-API ist eventuell dadurch zu erklären, dass die zusätzliche Pro-Punkt-Gewichtsberechnung erspart wurde. Es soll jedoch darauf hingewiesen werden, dass Frameratenunterschiede zwischen einzelnen Konfigurationen sehr gering sind, so dass auch eine Erklärung durch Zufallsverteilung der Werte nicht auszuschließen ist.

Es konnte eine Tendenz festgestellt werden, dass das Initialisieren des Tango-Dienstes mit einem Frameratenabfall um ca. 2 FPS einhergeht. Diese Phänomen scheint nicht anwendungsspezifisch zu sein und konnte auch bei einem einfachen Testprogramm, das nur wenige Systemressourcen beansprucht, beobachtet werden. Damit hängt die niedrige Framerate bei der Multi-Sensor-Konfiguration nicht direkt mit dem kombinierten Verfahren als solchem zusammen, sondern ist – unter anderem – technisch bedingt.

Als Zusammenfassung kann hier gesagt werden, dass die in den Versuchen erreichten Frameraten für eine Echtzeitanwendung kritisch niedrig sind und insbesondere unter der Aktualisierungsfrequenz der Tiefenkamera (15 FPS) liegen. Wie im Abschnitt 6.2.1 gezeigt wurde, führt das zu Einbußen an Tracking-Genauigkeit bei einer Erhöhung der Kamerageschwindigkeit.

Tabelle 6.5: Frameraten

	Basis	Krümmung	Pmd-Konfidenz	2 Ebenen	Multi-Sensor
FPS	8	7	9	9	6

6.4 Weiteres

6.4.1 Konfidenz-Schwellenwert für stabile Punkte

Im Laufe der Versuche sind wir auf ein unerwartetes Verhalten des Systems aufmerksam geworden: wenige Frames nach dem Scan-Start verschwand das Modellbild im Render-Fenster (je nach Szene komplett oder zum größten Teil) und erschien nach und nach, ausgehend vom Bildzentrum, wieder. Die Erklärung dafür war, dass sich alle bzw. die meisten Modellpunkte zu diesem Zeitpunkt in einem instabilen Zustand befanden (nach der Standardeinstellung werden instabile Punkte nicht gerendert). Die nachfolgende Frameworkanalyse ergab, dass dieses Problem durch Verwendung einer anderen Tiefenkamera im mobilen System zustande kam. Der unten stehende Abschnitt enthält die Problembeschreibung und einen Lösungsvorschlag.

Wie es in 3.1 erörtert wurde, wird der Zustand der Modellpunkte durch den Schwellenwert w_{stable} geregelt. Zusätzlich dazu wird ein weiterer Schwellenwert für die Anzahl der Frames $I^{unstable}$ festgelegt, so dass die Untermenge der Modellpunkte $\{\mathbf{p}_{\mathcal{M}} | w_{\mathcal{M}} < w_{stable}\}$ im Frame $i < I^{unstable}$ bei der Erzeugung der Modell-Vertexkarte $\mathcal{V}_{\mathcal{M}}$ mit berücksichtigt wird. Dies ist zur Initialisierung des Modells nötig, wenn alle Punkte noch instabil sind. Beim Fusionieren mit einem Eingabepunkt $\mathbf{p}_i$ wird der Konfidenzzähler $w_{\mathcal{M}}$ gemäß der Formel 3.3 erhöht. In der aktuellen Implementierung wird die Eingabekonfidenz w_i dabei noch umgekehrt proportional zu seinem Radius gewichtet: $w_i \leftarrow \frac{w_i}{r_i}$. Damit tragen feinere Details stärker zur Oberflächenrekonstruktion bei. Der Radius r_i hängt von der Größe der Nachbarschaft des Punktes p_i ab und insbesondere vom Abstand zum Punkt p_i^{diag}, der aus dem Diagonalpixel erzeugt wird. Nach der Formel 2.1 hängen die aus einem Pixel $\mathbf{u}$ erstellte Punktkoordinaten $\mathcal{V}(\mathbf{u}) = (x, y, z)^T$ mit der Brennweite zusammen: $x \propto \frac{1}{f_x}$, $y \propto \frac{1}{f_y}$. Damit haben Nachbarpunkte bei einer kleineren Brennweite (und demselben Tiefenwert) weitere Abstände und folglich auch größere Radien.

Die ursprüngliche Einstellungen des Frameworks ($w_{stable} = 5, I^{unstable} = 20$) sind auf die Kinect-Kameras ausgelegt, die eine größere Brennweite als das in unserem System verwendete CamBoard pico flexx haben. So beträgt sie z.B. beim Modell Kinect 2 366 Pixel [FMR+15], während beim pico flexx das 213,117 Pixel sind (die Werte für x- und y-Dimension sind in beiden Fällen gleich). Daraus ergibt sich für einen Punkt der Pmd-Kamera eine Vergrößerung der x und y Koordinate um einen Faktor $k \approx 1,7$, was auch im Allgemeinen zu einem größeren Radius führt (die Abhängigkeit des Radius von der Brennweite ist etwas komplexer, da sein Wert auch von den jeweiligen Tiefe und Punktnormale mit bestimmt wird.) Als Folge bekommen die Eingabepunkte in unserem System tendenziell kleinere Konfidenzwerte als bei der Verwendung einer Kinect-Kamera. Aus diesem Grund können die meisten Modellpunkte den Zählerwert $w_{\mathcal{M}} \geq w_{stable}$ zum $I^{unstable}$-ten Frame noch nicht akkumulieren und werden bei der Erstellung der Modell-Vertexkarte aussortiert. In den nachfolgenden Frames erreichen immer Punkte den stabilen Zustand, so dass $\mathcal{V}_{\mathcal{M}}$ allmählich wieder gefüllt wird.

Aus der obigen Beschreibung folgt die Notwendigkeit, den Schwellenwert w_{stable} entsprechend den intrinsischen Parametern der Tiefenkamera so anzupassen, dass die Modellpunkte nicht unnötig

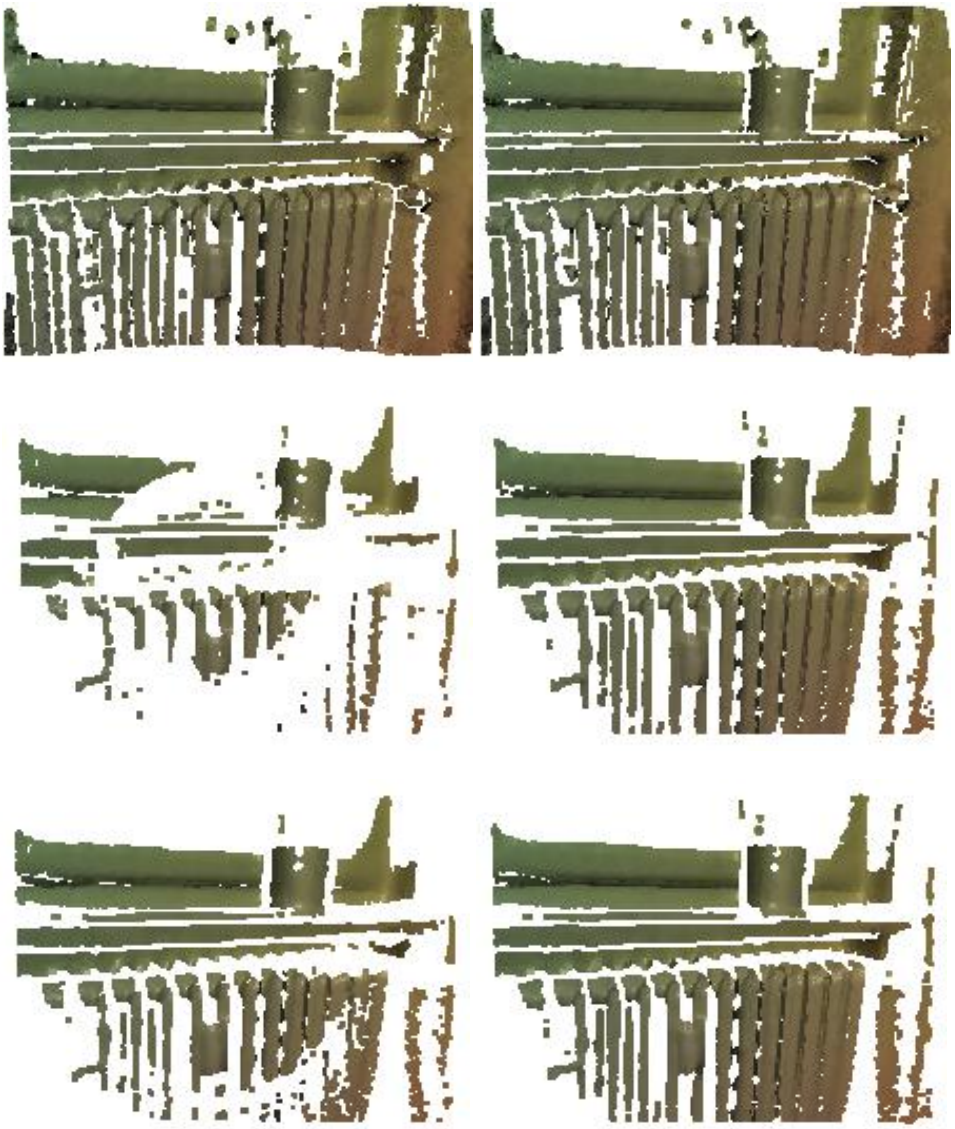


Bild 6.10: Rekonstruktion mit unterschiedlichen Schwellenwerten für stabilen Zustand. Linke Spalte: der Default-Wert $w_{stable} = 5$, rechte Spalte $w_{stable} = 2,94$; von oben nach unten: Frame i = 19, 20, 25.

lange im instabilen Zustand bleiben. Der optimale Wert dafür soll experimentell ermittelt werden – z.B. durch Offline-Rekonstruktion mit unterschiedlichen Einstellungen – was nicht unter die Ziele dieser Masterarbeit fällt. An der Stelle ist nur noch beispielhaft zu demonstrieren, wie eine erste grobe Anpassung des Wertes das beschriebene Systemverhalten ändert. Die Abbildung 6.10 zeigt Rekonstruktion einer Szene mit dem Default-Wert $w_{stable} = 5$ und als Vergleich mit einem angepassten Schwellenwert $w'_{stable} = w_{stable}/1,7 \approx 2,94$. Es sind der Frame $i = I^{unstable} - 1$, der „Verschwinden-Frame "$i = I^{unstable}$ und der Frame $i = I^{unstable} + 5$ mit dem zum Teil wiederhergestellten Bild dargestellt. Es ist zu sehen, dass nach dem Reduzieren des Schwellenwerts ein

wesentlicher Teil des Modells im Frame $i = I^{unstable}$ bereits stabil ist.

6.4.2 Versuch mit anderen Objekten

Mit dem Ziel zu überprüfen, inwieweit die in den Abschnitten 6.1 / 6.2 beschriebenen Fehler am Anfang des ICP-Trackings durch die Szenengeometrie und inwieweit durch systeminterne Ursachen bedingt sind, wurde ein zusätzlicher Versuch mit anderen Objekten durchgeführt. Die neue Szene bildeten strukturreiche Figuren aus Lego-Duplo-Steinen, die mehr geometrische Information für ICP liefern sollten (siehe Abbildung 6.11). Der Versuch erfolgte mit der ICP-Basiskonfiguration. Die Statistik in der Tabelle 6.4.2 zeigt eine geringfügige Genauigkeitssteigerung bezüglich der Rekonstruktion der ursprünglichen Szene (siehe Tabelle 6.1.2). Aufschlussreich ist visuelle Analyse des initialen Abschnitts der rekonstruierten Kameralaufbahnen in den beiden Szenen (siehe Abbildung 6.12, als Repräsentant der ersten Versuchsreihe ist hier Basis1 genommen). Auch im Extra-Versuch ist ein deutlicher Kamera-Drift beim Starten des Scannens zu merken. Die beiden Drift-Abschnitte weisen eine sehr ähnliche Form auf, auch wenn der aus der Lego-Szene etwas kürzer ausfällt. In der Abbildung sind auf den beiden Kamerapfaden Positionen markiert, an denen der (grobe) Drift aufhört und die Posenschätzung sich gewissermaßen stabilisiert.

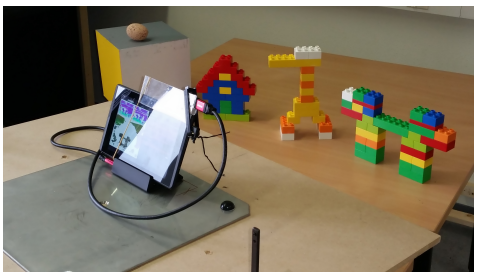


Bild 6.11: Szene aus den Lego-Duplo-Figuren für den Extra-Versuch.

Tabelle 6.6: ICP Tracking: Szene aus Lego-Duplo-Figuren

Positionsfehler, mm	0,4933 $\pm$ 0,5373
Orientierungsfehler, $^\circ$	13,1262 $\pm$ 4,3891
Drehradius, m	0,1805
Winkel zur Drehebene, $^\circ$	62

Die oben stehende Beobachtung legt die Vermutung nahe, dass das ICP-basierte Tracking im vorliegenden System generell an Präzisionsverlust in der Anfangsphase leidet, in der das Modell noch relativ dünnbesetzt ist und damit noch wenig Information akkumuliert hat. Dieses Effekt kann durch

Scannen „informationsreicher“ Objekte oder Verwendung zusätzlicher Informationen wie Krümmung minimiert werden, in den durchgeführten Tests war es jedoch nicht vollständig zu beheben. Zum Ergründen der genauen Ursachen des Problems sind weitere, zielgerichtete Versuche erforderlich.

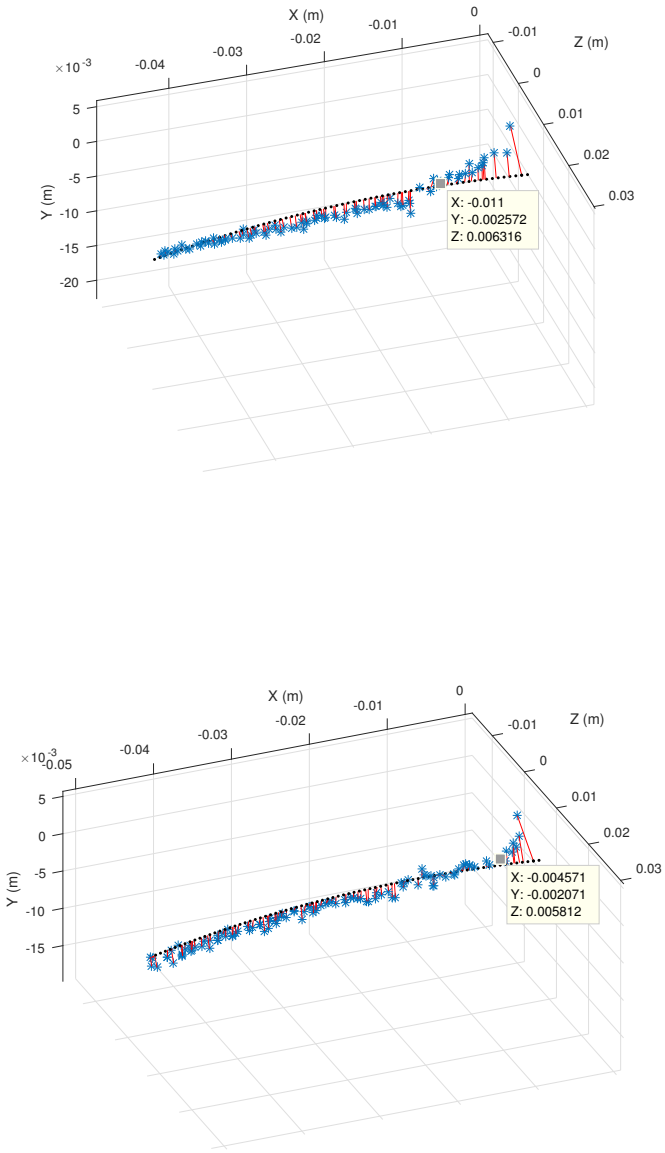


Bild 6.12: Initialer Kamera-Drift bei Rekonstruktion unterschiedlicher Szenen. Links: Basis1-Rekonstruktion der Standardszene aus der ersten Versuchsreihe; rechts: Extraversuch mit der Lego-Szene. Die markierten Stellen entsprechen ungefähr dem Ende des Drifts.

Kapitel 7

Zusammenfassung und Ausblick

Im Rahmen der vorliegenden Arbeit wurde das punktbasierte 3D Rekonstruktionsframework auf ein mobiles System, bestehend aus einem Tango Tablet und einer pico flexx Tiefenkamera, portiert. Mit dem Ziel, Schwächen eines mobilen Systems zu kompensieren und seine Stärke auszunutzen, wurde eine Erweiterung des ursprünglichen 3D Kamera-Trackings um zusätzliche Bewegungserfassungsquellen konzipiert und implementiert. Konkret handelt es sich um die Nutzung der aktuellen Pose, die mit dem Tango-VIO-Verfahren geschätzt wurde, zur Initialisierung des ICP-Algorithmus im Rekonstruktionspipeline. Es wurde angenommen, dass ICP dadurch näher zur gesuchten Lösung startet, was wiederum eine Konvergenzbeschleunigung sowie eine Steigerung der Tracking-Genauigkeit herbeiführen würde.

Zur Überprüfung dieser Annahmen sowie zum Testen des Verhaltens des punktbasierten Frameworks in einer neuen – mobilen – Umgebung wurden Versuche mit kontrollierter Kamerabewegung durchgeführt, bei denen eine Szene aus wenigen Objekten mittlerer Größe eingescannt wurde.

Die Evaluierung der Versuchsergebnisse zeigte, dass mit dem Framework auch auf dem gegebenen System akzeptable Rekonstruktionsergebnisse, mit einigen Vorbehalten, erreicht werden konnten. Die Einschränkungen betreffen vor allem zwei Aspekte. Zum einen wurde ein systematischer Kamera-Drift während wenigen Frames am Anfang des Scan-Vorgangs festgestellt. Zum anderen erlauben die erreichten Frameraten von 7-9 FPS nur langsame Kamerabewegungen.

Die Annahmen über das Multi-Sensor-Tracking konnten durch experimentelle Ergebnisse nicht bestätigt werden. Eine Ursache dafür war ein unzuverlässiges Funktionieren des visuell-inertialen Trackings, das nur in wenigen Versuchen eine ausreichend genaue Posenschätzung lieferte. Andererseits konnte kein relevanter Einfluss der Initialisierung auf das Endergebnis des ICP-Algorithmus, und insbesondere auch auf den oben erwähnten Kamera-Drift, festgestellt werden.

Schlussfolgernd aus den Evaluierungsergebnissen wurde konstatiert, dass das kombinierte Kamera-Tracking in der hier verwendeten Form keine Optimierung gegenüber dem ursprünglichen Verfahren erzielt. In der Perspektive kann jedoch eine andere Gestaltung des Zusammenspiels verschiedener Tracking-Varianten in Erwägung gezogen werden. Anzustreben ist dabei ein Fusion-Algorithmus, der ein komplementäres Ausgleichen der Fehler ermöglichen würde, statt die Lösung einer der Varianten

als endgültiges Resultat zu übernehmen.

Literaturverzeichnis

[BHZK05] Mario Botsch, Alexander Hornung, Mathias Zwicker, and Leif Kobbelt. High-quality surface splatting on today's gpus. In *Eurographics Symposium on Point-Based Graphics*, pages 17–24. IEEE, 2005.

[BL95] Gérard Blais and Martin D. Levine. Registering multiview range data to create 3d computer objects. *IEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):820–824, 1995.

[BM92] Paul J. Besl and Neil D. McKay. A method for registration of 3-d shapes. *IEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992.

[CL96] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques)*, pages 303–312. ACM, 1996.

[CM91] Yuang Chen and Gérard Medioni. Object modeling by registration of multiple range images. In *Proceedings of the Third International Conference on Robotic and Automation*, pages 2724–2729. IEEE, 1991.

[DGKR13] Sean L. Bowman Dimitrios G. Kottas, Joel A. Hesch and Stergios I. Roumeliotis. On the consistency of vision-aided inertial navigation. In *Experimental Robotics*, pages 303–317. Springer, 2013.

[FMR+11] Michael Fleps, Elmar Mair, Oliver Ruepp, Michael Suppa, and Darius Burschka. Optimization based imu camera calibration. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3297–3304. IEEE, 2011.

[FMR+15] Michael Fleps, Elmar Mair, Oliver Ruepp, Michael Suppa, and Darius Burschka. First experiences with kinect v2 sensor for close range 3d modelling. In *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, volume XL-5/W4, pages 93–100. ISPRS, 2015.

[KDSX15] Matthew Klingensmith, Ivan Dryanovski, Siddhartha S. Srinivasa, and Jizhong Xiao. Chisel: Real time large scale 3d reconstruction onboard a mobile device using spatially-hashed signed distance fields. In *Robotics: Science and Systems*, 2015.

[KLL+13] Maik Keller, Damien Lefloch, Martin Lambers, Shahram Izadi, Tim Weyrich, and Andreas Kolb. Real-time 3d reconstruction in dynamic scenes using point-based fusion. In *3D Vision - 3DV 2013, 2013 International Conference on*, pages 1–8. IEEE, 2013.

[KP15] Andreas Kolb and Fabrizio Pece. Range imaging. In *Digital Representations of the Real World*, chapter 4, pages 51–64. CRC Press, 2015.

[KZY15] Andreas Kolb, Jiejie Zhu, and Ruigang Yang. Sensor fusion. In *Digital Representations of the Real World*, chapter 9, pages 133–150. CRC Press, 2015.

[LKS+17] Damien Lefloch, Markus Kluge, Hamed Sarbolandi, Tim Weyrich, and Andreas Kolb. Comprehensive use of curvature for robust and accurate online surface reconstruction. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2017.

[NDF14] Matthias Nießner, Angela Dai, and Matthew Fisher. Combining inertial navigation and icp for real-time 3d surface reconstruction. In *EUROGRAPHICS 2014*. EG, 2014.

[NIH+11] Richard Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew Davison, and Andrew Fitzgibbon Pushmeet Kohli Jamie Shotton, Steve Hodges. Kinectfusion: Real-time dense surface mapping and tracking. In *Proceedings of the 10th IEEE International Symposium on Mixed and Augmented Reality*, pages 127–136. IEEE, 2011.

[NNB04] David Nistér, Oleg Naroditsky, and James Bergen. Visual odometry. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 652–659. IEEE, 2004.

[NZW+13] Nargess Sadaghzadeh Nokhodberiz, Leila Mekacher Zouaghi, Achim Wagner, Eugen Nordheimer, Javad Poshtan, and Essameddin Badreddin. Cascaded kalman filters for inertial navigation system error estimation fusing optical and inertial sensors. In *THE 5th INTERNATIONAL CONFERENCE ON MECHATRONICS*, 2013.

[RHHL02] Szymon Rusinkiewicz, Olaf Hall-Holt, and Marc Levoy. Real-time 3d model acquisition. In *MAC Transactions on Graphics (Proceedings SIGGRAPH)*, volume 21, pages 438–446. ACM, 2002.

[RL01] Szymon Rusinkiewicz and Marc Levoy. Efficient variants of the icp algorithm. In *Proceedings of the Third International Conference on 3-D Digital Imaging and Modeling*, pages 145–152. IEEE, 2001.

[Sab11] Angelo Maria Sabatini. Kalman-filter-based orientation determination using inertial/magnetic sensors: Observability analysis and performance evaluation. *Sensors*, (11):9182–9206, 2011.

[Tau91] Gabriel Taubin. Estimation of planar curves, surfaces, and nonplanar space curves defined by implicit equations with applications to edge and range image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(11):1115 – 1138, Nov 1991.

[Woo07] Oliver J. Woodman. An introduction to inertial navigation. Technical Report 696, University of Cambridge, 2007.

[YB06] Xiaoping Yun and Eric R. Bachmann. Design, implementation, and experimental results of a quaternion-based kalmman filter for human body motion tracking. *IEEE Transactions on Robotics*, 22(6):1216–1227, 2006.

[Zha00] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000.

[ZPvBG01] Mathias Zwicker, Hanspeter Pfister, Jeroen van Baar, and Markus Gross. Surface splatting. In *Computer Graphics (Proceedings SIGGRAPH)*, pages 371–378. ACM, 2001.