Abstract

Im Rahmen dieser Studienarbeit soll die Kinect-Kamera in Bezug auf SLAM (Simultaneous localization and mapping) und andere Computer-Vision-Anwendungen untersucht werden.
Die Kinect ist ein Gerät aus der Unterhaltungselektronik, das Sensorik für den Einsatz im Bildverstehen enthält. Das sind eine herkömmliche Farbkamera und ein Sensormodul, das Information über Distanz von Punkten im Bildbereich liefern kann.

Damit macht das Gerät Funktionalität massenverfügbar, die eine wichtige Rolle im ROS (Robot Operating System) spielen, wie etwa das umfangreiche und verteilte Erfassen von Alltagsgegenständen, aber auch umgekehrt das Erkennen von bereits im ROS erfassten Objekten in der eigenen Umgebung.

Es soll evaluiert werden, inwiefern sich das SLAM-Verfahren dafür eignet, ein Abbild der direkten Umwelt und der darin enthaltenen Gegenstände in diesem Sinne zu erstellen. Davon ausgehend können dann in einem nächsten Schritt Werkzeuge für diese Anwendungen entwickelt werden, die sich bestehenden Strukturen einfügen.

Inhalt

1 Einführung

1.1 Vorstellung des Kinect-Gerätes

Die Kinect ist ein Gerät aus dem Segment der Unterhaltungselektronik, das im November 2010 von der Firma Microsoft im Rahmen seiner Unterhaltungselektroniksparte auf den Markt gebracht wurde.
Es ist als Zubehör zu einer Spielekonsole konzipiert, das die klassische Bedienung der Konsole durch ein manuelles Eingabegerät ersetzen kann.

Die mechanische und elektronische Ausrüstung des Geräts umfasst Komponenten, die es erlauben, Ton- und Bilddaten dreidimensional zu erfassen. Außerdem enthalten ist ein Motor und begleitende Sensorik, die das kontrollierte Kippen des Geräts in kleinem Maße erlauben.
Diese Arbeit konzentriert sich auf die bilderfassende Sensorik des Geräts.

Bald nach seinem Erscheinen hat die Kinect großen Anklang in den Kreisen von Hobbyisten und Wissenschaftlern im Bereich der Bildverarbeitung gefunden. Noch im Monat der Veröffentlichung standen nicht-offizielle Open-Source-Treiber zur Verfügung, die programmatischen Zugriff auf die Video-Komponenten des Geräts unter dem Betriebssystem Linux erlaubten.

Die Bedeutung der Kinect für Anwendungen der Bildverarbeitung besteht zum Teil darin, dass RGBD-Funktionalität – das Erfassen von Farb- und Tiefeninformation in einem Gerät - traditionell nur zu weitaus höheren Preisen verfügbar war.

Alternative RGBD-Geräte, die in Internet-Recherchen prominent auftauchen (konkret der SwissRanger 4000 der Firma Mesa Imaging und der CamCube der Firma PMD Technologies), sind in Preisregionen von etwa USD10000 (Stand August 2011) zu finden. Das steht dem Preis der Kinect von USD150 (Stand August 2011) gegenüber.

Diese Spezialgeräte decken natürlich ein anderes Anwendungsgebiet ab und bieten einen besseren Funktionsumfang.
Es bleibt jedoch die Tatsache bestehen, dass mit der Kinect und durch seine Massenproduktion erstmals ein erschwingliches RGBD-Gerät erhältlich wurde.

1.2 Motivation

Die Art von Aufnahmen, die mit der Kinect möglich sind, macht sie für viele Bereiche des digitalen Bildverstehens interessant.
SLAM (Simultaneous Localization and Mapping) ist ein Dachbegriff für eine Vielzahl an Verfahren, die sich mit der Erfassung der Umwelt durch Roboter und autonome Geräte befassen.
Die 3D-Bildinformationen, die das Gerät zur Verfügung stellt, haben in diesen Verfahren eine große Bedeutung.
Es ist also von Interesse, herauszubekommen, inwiefern sich die Kinect für diese Art Anwendungen eignet.
Zudem hat das Produkt durch seine breite Verfügbarkeit das Potential, diese Art Technologie, die sich traditionell nur in Laboren und in Industrieanwendungen findet, etwas näher in den Alltag zu bringen und so neue Anwendungsfelder zu öffnen.

1.3 Aufbau der Arbeit

Zunächst wird auf die wichtigsten technischen Aspekte des Kinect-Gerätes eingegangen, um seine Funktionsweise darzulegen und Besonderheiten darzulegen.

Mit diesem Wissen wird dann auf das Thema der Kalibrierung eingegangen, speziell im Hinblick auf die Kinect.

Davon ausgehend werden einige Verfahren der Bildregistrierung vorgestellt, die Kern der digitalen Rekonstruktion von Objekten sind, wie sie auch beim SLAM zum Einsatz kommt.

Daraufhin wird auf den Implementierungsteil der Arbeit eingegangen. Die wichtigsten Komponenten werden vorgestellt und einige praktische Erfahrungen, die sich aus dem Umgang mit der Kinect ergeben haben, werden dargelegt.

Abschließend folgt eine Zusammenfassung mit Hinblick auf die Aufgabenstellung.

1.4 Verwandte Arbeiten

- Nutzung von RGB-D-Kameras zum Erfassen von Innenräumen (1)

- RGBDemo – Ein Programm von Nicolas Burrus, das mit unterschiedlichen Anwendungen der Kinect experimentiert. U.a. Objektrekonstruktion, Menschenerkennung, Kalibrierung sind enthalten[1].

- RGBD-6D-SLAM (2)

- RoboEarth-Projekt[2], das Module für Objektrekonstruktion enthält

[1] http://nicolas.burrus.name/index.php/Research/KinectRgbDemoV6

[2] http://www.ros.org/wiki/roboearth

2 Technische Betrachtung der Kinect-Bilderfassung

Im Folgenden wird die Bilderfassung der Kinect vorgestellt und eine kurze qualitative Betrachtung der Bilddaten ausgeführt.

2.1 3D-Bilderfassung

Die Details zur Funktionsweise der Kinect-Kameras sind der technischen Beschreibung durch das ROS-Projekt entnommen (3).

Um die Erfassung von 3D-Bilddaten zu erlauben, enthält das Gerät einen Projektor von strukturiertem Infrarotlicht, eine Infrarotvideokamera und eine Farbvideokamera.
Hierbei erlaubt der Projektor in Kombination mit der Infrarotkamera die Schätzung von Abständen im erfassten Bereich.
Die Farbvideokamera kann diese Tiefendaten nach entsprechender Verarbeitung um Farbinformationen bereichern und außerdem als klassischer 2D-Bildnehmer funktionieren.

Der Infrarot-Projektor und die Infrarot-Kamera arbeiten schemahaft folgendermaßen zusammen:

- Der Projektor projiziert ein bekanntes Muster in den Raum
- Die Infrarot-Kamera nimmt die Szene im Raum mit dem überlagerten Muster auf
- Für eine Referenzentfernung/Referenzebene ist bekannt, wie das Muster im Bild erscheinen muss
- Aus der horizontalen Versetzung des Projektors und der Kamera ergibt sich nun, dass das Muster weiter links im Bild erscheint, falls es hinter der Referenzebene auf eine Oberfläche trifft und weiter rechts im Bild erscheint, falls es vor Referenzebene auf eine Oberfläche trifft
- Da das Muster bekannt ist, kann einem beobachteten Teilmuster die erwartete Position auf der Referenzebene zugewiesen werden
- Aus der erwarteten und der tatsächlichen horizontalen Position lässt sich per Triangulation die Entfernung einer Oberfläche im Raum bestimmen

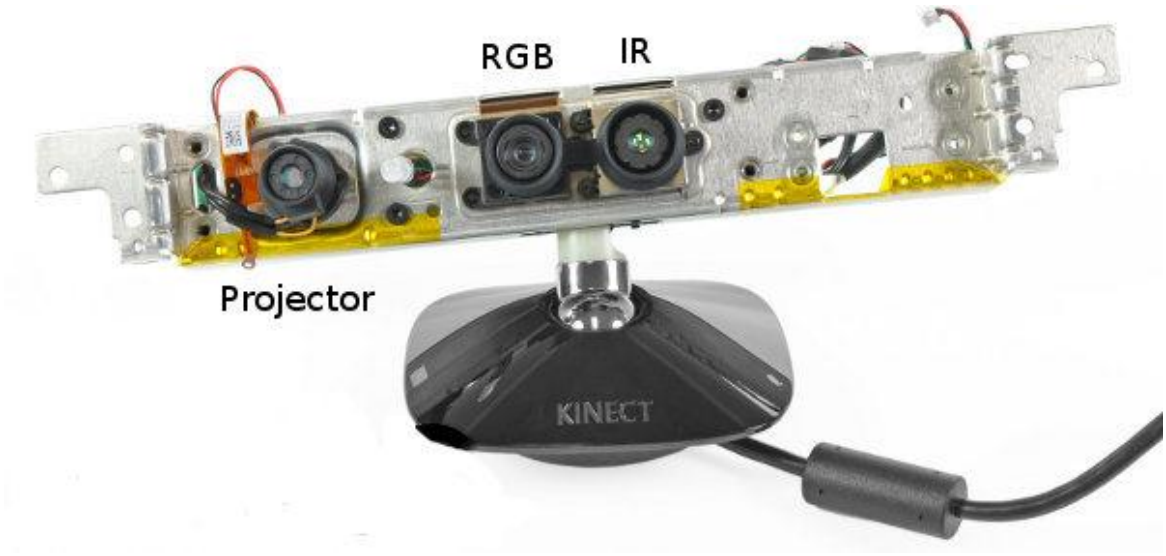

Abbildung 1: Kinect ohne Gehäuse mit Bezeichnung der Bildkomponenten[3]

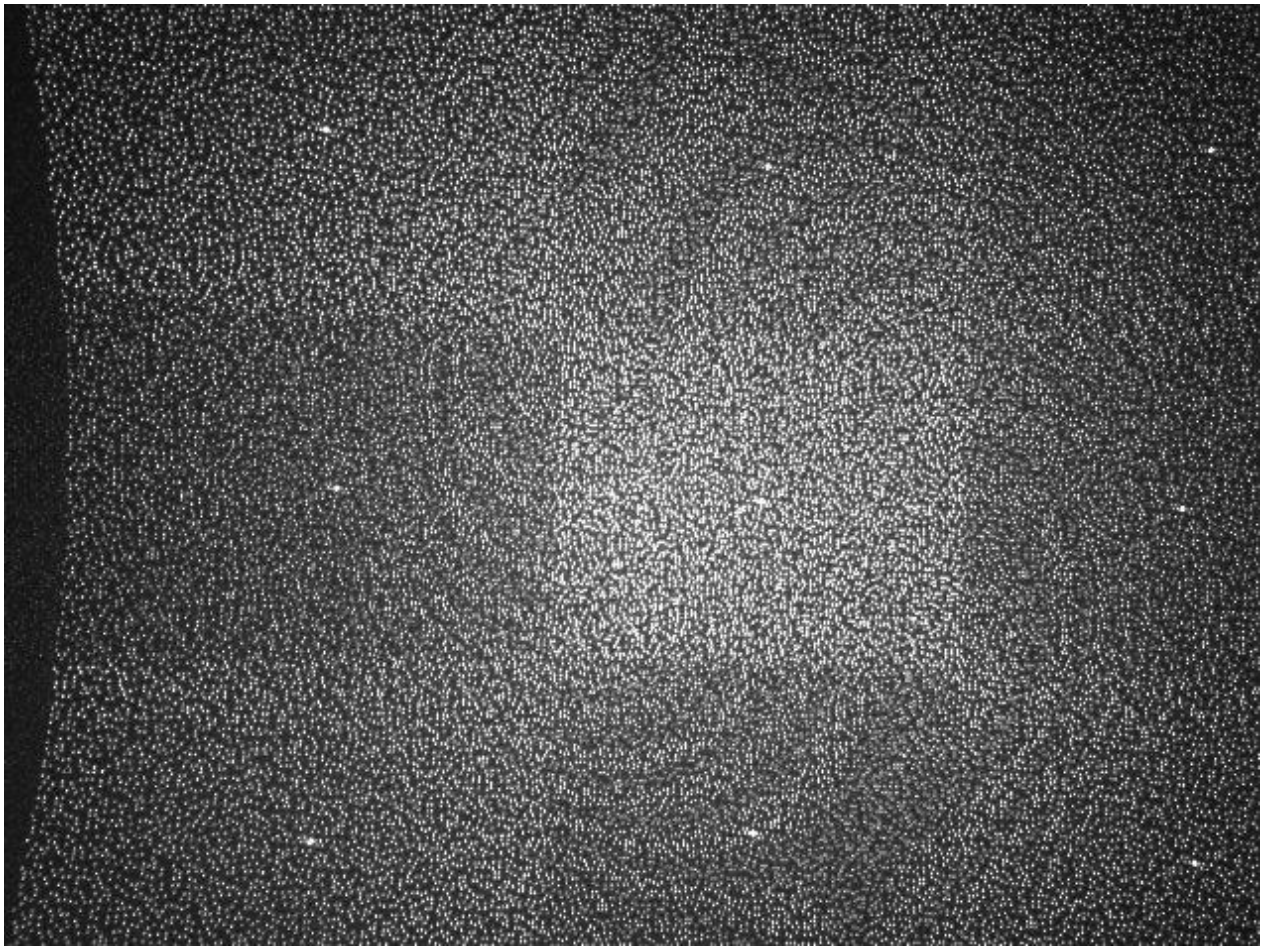

Abbildung 2: IR-Projektion der Kinect

[3] http://www.ifixit.com/Teardown/Microsoft-Kinect-Teardown/4066/3 (Creative Commons)

2.2 Betrachtung der Bilddaten

Die Verwendung des infraroten Lichtbereichs für Tiefenerfassung scheint sich aus eigener Betrachtung daher zu ergeben, dass IR-Quellen unter Innenraumbedingungen in der Regel flächenhaft strahlen und sehr schwach sind. Das projizierte Muster ist jedoch sehr punktuell und vergleichsweise intensiv, sodass hier wenig Beeinflussung stattfindet. Zudem spricht sicherlich die Notwendigkeit, den Benutzer nicht zu blenden und das herkömmliche Farbbild nicht zu stören, für die Verwendung eines nicht-sichtbaren Wellenlängenbereichs.

Aus der Untersuchung des Tiefenbildes lässt sich feststellen, dass die Kinect Entfernungen im Bereich von ca. 0,6m – ca. 6m berechnen kann.
Es fällt auf, dass die Qualität der Tiefendaten für Objekte in Entfernungen von mehr als 1,5m-2m abnimmt. Hier scheint über die Tatsache hinaus, dass für weiter entfernte Objekte weniger Pixelinformationen zur Verfügung stehen, auch eine größere Unsicherheit in der Schätzung der Entfernung hineinzuspielen, da mehr Ausreißer in den Daten zu beobachten sind.
Schwierigkeiten machen erwartungsgemäß auch durchsichtige und reflektierende Oberflächen, sowie starke Lichtquellen (etwa einfallendes Sonnenlicht).

Aufgrund der horizontalen Versetzung des IR-Projektors und der IR-Kamera kommt es zu einer Art Schattenwurf. Das Licht des Projektors erreicht nicht alle Stellen, auf die die Kamera eigentlich freien Blick hat.

2.3 Erstellen einer Punktwolke aus den Daten

Für die Berechnung einer Punktwolke in metrischen Einheiten benötigen wir die intrinsischen Parameter der IR-Kamera.
Diese werden im Kapitel *Kalibrierung* vorgestellt. Hier nur die Berechnungsformeln, um von einem Tiefenwert eines Pixels des libfreenect-Treibers zur metrischen Distanz zu gelangen (Modell der Lochkamera, Parameter sind (4) entnommen):

$$Z = 1.0/(z * -0.0030711016 + 3.3309495161)$$

$$X = \frac{(x - c_x)Z}{f_x}$$
$$Y = \frac{(y - c_y)Z}{f_y}$$

Hierbei sind x und y die Koordinaten eines Pixels im Tiefenbild und z der Tiefenwert.
Das Ergebnis ist der Punkt $P = (X, Y, Z)$ der Szene in metrischen Einheiten.

3 Kalibrierung

Bei der Behandlung von Bilddaten geht man vom Modell der Lochkamera aus.

Dieses Modell liefert uns eine Umrechnung von Weltkoordinaten $Q = (X, Y, Z)$ in die Pixelkoordinaten $p = (x, y)$ des Kamerasensors:

$$x = f_x\left(\frac{X}{Z}\right) + c_x, \qquad y = f_y\left(\frac{Y}{Z}\right) + c_y$$

f_x, f_y ergeben sich aus der physikalischen Brennweite umgerechnet in die Einheit Pixel entsprechend der jeweiligen Abmessung eines Sensorelements ($f_x = Fs_x$, $f_y = Fs_y$).
c_x, c_y beschreiben, um welchen Abstand das Zentrum des Sensorelements von der optischen Achse entfernt ist.

Diese vier Parameter werden als die intrinsischen Parameter der Kamera bezeichnet.

Reale Kameras führen Abweichungen von diesem idealisierten Modell ein, die berücksichtigt werden müssen.
Je nach Montageverfahren kann es passieren, dass Linse und Sensor nicht exakt parallel ausgerichtet sind (etwa durch Befestigung mit Klebstoff).
Der entstehende Fehler nennt sich tangentiale Verzerrung.

Außerdem führt die Verwendung von Linsen zu Effekten optischer Verzerrung. Hier ändert sich die Bildvergrößerung durch die Linse mit dem Abstand der einfallenden Lichtstrahlen zum Linsenzentrum.
Dieser Fehler wird als radiale Verzerrung bezeichnet. Der Fehler ist an den Linsenaußenrändern am stärksten ausgeprägt.

Mit der Kalibrierung einer Kamera lassen sich Modellparameter finden, um die radiale und tangentiale Verzerrung herauszurechnen.

Die radiale Verzerrung kann durch eine Taylor-Reihe modelliert werden (5) (r ist der Abstand vom Linsenmittelpunkt):

$$f_{radial}(r) = a_0 + a_1 r + a_2 r^2 + a_3 r^3 + \cdots$$

Unter den Annahmen, dass wir um den Linsenmittelpunkt ($r = 0$) entwickeln und die Verzerrung symmetrisch zum Linsenmittelpunkt ist, gelangen wir in der Praxis zu

$$f_{radial}(r) = a_2 r^2 + a_4 r^4$$

Im Falle besonders ausgeprägter Verzerrung (z.B. Fischaugeneffekt) wird bis zum Grad 6 der Reihe entwickelt (5).
Dieses Modell führt also die zu bestimmenden Parameter a_2, a_4 $(, a_6)$ ein.

Die tangentiale Verzerrung äußert sich in einer Abweichung

$$f_{x,tangential}(r, x, y) = 2p_1 y + p_2(r^2 + 2x^2)$$
$$f_{y,tangential}(r, x, y) = p_1(r^2 + 2y^2) + 2p_2 x$$

3.1 Stereokalibrierung

Um die Tiefeninformationen und die Farbinformationen miteinander in Beziehung bringen zu können, muss nun noch eine Stereokalibrierung erfolgen.
Denn die Anordnung der beiden Kameras in der Kinect bedingt, dass sie die Bildszene aus einer jeweils leicht anderen Perspektive betrachten. Die Überlagerung kann also nicht Eins-zu-Eins erfolgen.
Es muss eine Beziehung gefunden werden, um die Datenpunkte der beiden Kamerabilder einander zuordnen zu können.

Die Anordnung der Kameras zueinander lässt sich durch eine Rotation und eine Translation beschreiben. Es müssen also die beiden Matrizen gefunden werden, die diese Bewegung beschreiben.

4 Bildregistrierung

Die Bildregistrierung ist ein Optimierungsproblem, deren Ziel es ist, zwei oder mehr Bilder einer Szene in Beziehung zu bringen.
Ein Bild der Menge wird als Referenzbild gewählt und für die verbleibenden Bilder wird jeweils eine Transformation gesucht, die eine Überführung in das Referenzbild beschreibt.
Wenn diese Transformationen paarweise gefunden wurde, können die verschiedenen Bilder zu einem Ganzen ergänzt werden.

4.1 Iterative-Closest-Point-Algorithmus (ICP)

Die Eingabe in den ICP-Algorithmus ist ein Paar von Punktwolken, die überlappen. Die Ausgabe im Erfolgsfall ist eine rigide Transformation (Translation, Rotation), die die zweite Punktwolke in die erste Punktwolke überführt.
In diesem iterativen Verfahren wird versucht, die Summe der Quadrate der Abstände zweier korrespondierender Punkte für den überlappenden Bereich zu minimieren.
Der Algorithmus wurde in (6) erstmals beschrieben und läuft wie folgt ab:

- Rate eine anfängliche Transformation, setze den Fehler zu ∞
- Iteriere
    - Für jeden Punkt in M finde den nächsten Punkt in P
    - Berechne die Transformation für die im vorherigen Schritt berechnete Zusammengehörigkeit
    - Wende die Transformation an
    - Berechne den Fehler
    - Falls der Fehler akzeptabel ist, sind wir fertig

Hierbei ist M die Referenzpunktwolke und P die auszurichtende Punktwolke.

Das Verfahren setzt voraus, dass die Punktwolken zu einem großen Teil überlappen.

Gegen Schwächen des Verfahrens, wie sie in (7) beschrieben werden, wurden verschiedene Erweiterungen und Verbesserungen des grundlegenden ICP-Algorithmus in der Literatur beschrieben.
So beschreibt (8) eine ICP-Implementierung, die sich u.a. auch durch ihre Tauglichkeit für Echtzeitanwendungen auszeichnet.

In (9) wird eine Variante beschrieben, die das Minimierungskriterium (minimale Quadrate der Abstände, bzw. least squares (engl.)) durch ein anderes ersetzt (Least Trimmed Squares, LTS). Durch Verwendung von LTS fallen Ausreißer in den Daten weniger ins Gewicht und somit verbessert sich die Robustheit des Verfahrens.

In (10) werden weitere Varianten und ihr Laufzeitverhalten diskutiert und auch eine Betrachtung der Robustheit der jeweiligen Algorithmen vorgenommen. Es werden auch Alternativen zum Nächsten-Nachbar-Problem beschrieben.

Eine einfache denkbare Erweiterung im Zusammenhang mit der RGBD-Bilderfassung ist es, die Farbinformation an einem Punkt mit zu berücksichtigen. Allerdings führen leichte Perspektivänderungen, die durch Freihandrekonstruktion gegeben sind, und entsprechender Schattenwurf, schnell zu Farbänderungen.

Die PCL-Bibliothek verfügt über eine Implementierung des ICP-Verfahrens, deren Implementierungsdetails in (7) diskutiert werden.

4.2 Merkmalsbasierte Bildregistrierung

Bei merkmalsbasierten Verfahren wird ein Satz von Merkmalen aus dem Referenzbild extrahiert. Jedes dieser Merkmale wird nun in den Alternativansichten der Szenen gesucht.
Wird ein Merkmal in beiden Bildern gefunden, kann mithilfe dieser Korrespondenz eine Transformation berechnet werden, die die Bilder ineinander überführt.

Für die Anwendung in der Bildregistrierung werden Merkmale gesucht, die invariant gegenüber Skalierung und Rotation sind.
Ein solches Verfahren ist SIFT (Scale-invariant feature transform). Bei diesem Verfahren werden für die gefundenen Merkmale Deskriptoren errechnet. Diese Deskriptoren sind Grundlage der Skaleninvarianz dieses Verfahrens.

In (11) wird ein Verfahren beschrieben, um SIFT für Merkmalsfindung in Punktwolken zu benutzen. Da SIFT sich nicht für Merkmalsextraktion in 3D-Punktwolken eignet, beschreiben die Autoren eine Umwandlung der Punktwolke in 2D-Daten. Sie gehen dabei so vor, dass sie die Euklidische Distanz eines 3D-Punktes zum Ursprung nehmen und darauf die Wurzelfunktion anwenden.
Sie verwenden dann RANSAC (12), um die Korrespondenzen zwischen den Punktwolken zu finden.

SURF (13) ist ein moderneres Verfahren, das einige Ursprünge in SIFT findet, jedoch den Autoren zufolge mehrfach schneller arbeitet.
Die Bibliothek OpenCV, die später noch behandelt wird, beinhaltet eine Implementierung des SURF-Algorithmus.

4.2.1 Verfeinerung mit ICP

Die mit den merkmalsbasierten Verfahren gefundenen Transformationen können nun noch weiter verbessert werden, in dem auf die transformierten Punktwolken und die Referenz jeweils das oben beschriebene ICP-Verfahren angewandt wird (14).

5 Implementierung

5.1 Verwendete Software

In folgenden Abschnitten wird an einigen Punkten auf bestimmte Softwarepakete verwiesen. Diese sollen hier kurz vorgestellt werden.
Die Entwicklung und Verwendung fand unter einem Linux-Betriebssystem statt.

5.1.1 FloDiEdi[4] (Flow Diagram Editor)

Das Programm FloDiEdi ist eine Entwicklung der Abteilung Bildverstehen des Instituts für Parallele und Verteile Systeme (IPVS) der Universität Stuttgart.

Kernpunkt ist die graphische Benutzeroberfläche, die die Verschaltung von Blöcken ermöglicht.

Diese Blöcke repräsentieren

- verschiedene Algorithmen des Bildverstehens, die als Teil des Funktionsumfangs implementiert sind

- Datenquellen (wie Kameras, Dateien)

- Datensenken (wie Anzeige, Dateispeicherung)

So können oft wiederkehrende Schritte in Verfahren der Bildverarbeitung auf einfache Weise in neue Entwicklungen integriert und wiederverwendet werden.
Die Architektur ist Plug-In-basiert.

Der Darstellung halber wird auf FloDiEdi-Plugins, die üblicherweise als Klasse mit der Bezeichnung

> *<plugin-name>plugin* (z.B. *KinectFreenectplugin*)

im Quelltext auftreten, als

> `FloDiEdi::<plugin-name>` (z.B. `FloDiEdi::KinectFreenect`)

verwiesen.

[4] http://sourceforge.net/projects/flodiedi/

5.1.2 OpenCV[5] (Open Computer Vision)

OpenCV ist eine freie Bibliothek, die eine große Menge an Algorithmen aus dem Gebiet der digitalen Bildverarbeitung in 2D und 3D umfasst. Sie ist größtenteils auf Echtzeitverarbeitung ausgelegt und entsprechend optimiert.

5.1.3 ROS[6] (Robot Operating System)

Das ROS ist ein Rahmenwerk, das die Entwicklung von Software für den Einsatz in Robotern erleichtern soll. Funktionen sind u.a. Hardwareabstraktion, Software-Paketverwaltung, Gerätetreiber und Methoden zur Kommunikation von ROS-Bestandteilen untereinander (15).

5.1.4 PCL[7] (Point Cloud Library)

Die PCL ist ein Bestandteil des ROS, steht jedoch auch als eigenständige Bibliothek zur Verfügung.
Sie wurde für die Behandlung von n-dimensionalen Punktwolken und 3D-Geometrie-Problemen entworfen.

5.1.5 OpenKinect libfreenect[8]

Eine Open-Source-Software-Bibliothek, die den Zugriff auf die Kinect-Datenströme ermöglicht, und das Gerät steuern kann.
Ein vergleichbares Projekt ist OpenNI[9].
libfreenect wurde gewählt, da ihr Einsatz zum Zeitpunkt der Erstverwendung unkomplizierter war. So erforderte OpenNI zu diesem Zeitpunkt eine weitauf aufwendigere Einrichtung auf dem Linux-basierten Zielsystem als libfreenect.

[5] http://opencv.willowgarage.com/
[6] http://www.ros.org/
[7] http://pointclouds.org/
[8] http://openkinect.org/
[9] http://www.openni.org/

5.2 Kalibrierung

5.2.1 FloDiEdi::KinectFreenect

Um auf die Daten der Kinect-Kamera zugreifen zu können, muss innerhalb von FloDiEdi der libfreenect-Treiber verfügbar sein.
Folgender Block mit 4 Ausgängen wurde implementiert:

- Farbbild der Farbkamera (24 Bits pro Pixel)
- Werte des Tiefensensors als Graustufenbild (8 Bits pro Pixel)
- Original-Werte des Tiefensensors (11 Bits pro Pixel)
- Graustufenbild der IR-Kamera (8 Bits pro Pixel, über Blockparameter zuschaltbar)

Die Ausgänge sind liefern Daten in einer Auflösung 640x480 Pixeln

Das Bild der IR-Kamera wird zusätzlich geringfügig in den Dimensionen verändert, da es in einer anderen Auflösung vom Gerät geliefert wird. Laut (3) ist die Diskrepanz auf ein Implementierungsdetail der internen Rechenhardware zurückzuführen.

Der Block kann auf Kalibrierungsdaten zurückgreifen und diese auf die Ausgänge anwenden (optional zuschaltbar).

5.2.2 FloDiEdi::KinectCalibrate

Mithilfe von `FloDiEdi::KinectFreenect` kann ein Kalibrierungsvorgang ausgeführt werden. Dazu müssen folgende Eingänge am Block anliegen:

- RGB-Bild
- IR-Bild

Bei der Ausführung werden jeweils die Kalibrierungsdaten für die Farb- und IR-Kamera gefunden. Anschließend wird eine Stereokalibrierung ausgeführt.
Das Ergebnis wird in den lokalen Speicher von FloDiEdi geschrieben und kann von `FloDiEdi::KinectFreenect` verwendet werden.

Die Implementierung der Block-Logik greift auf die OpenCV-Bibliothek zurück. Wie im Kapitel *Kalibrierung* beschrieben, muss eine Menge an Parametern errechnet werden, um ein konkretes Modell für die Verzerrungseffekte zu erhalten.

Diese Bibliotheksfunktionen arbeiten auf einer Menge von Kamerabildern, in denen die Kalibrierungsstruktur in Gänze zu erkennen ist.
Die Kalibrierungsstruktur ist ein planes Objekt, auf dem ein Schachbrettmuster aufgebracht ist. Die Eigenschaften der Struktur (Mustergröße, Abmessungen, Anzahl an Feldern) werden als Parameter übergeben.

Man achtet bei der Prozedur darauf, dass die Struktur in möglichst vielen Stellungen und in allen Bildbereichen in der Bildmenge auftritt, um eine gute Kalibrierung zu erhalten. Außerdem ist erwünscht, dass die Struktur möglichst nah an die Kamera herangeführt wird.

Wie eingangs erwähnt, wird mit `FloDiEdi::KinectCalibrate` auch die IR-Kamera kalibriert, obwohl wir in den Anwendungen nicht direkt am IR-Bild interessiert sind. Wie sich später jedoch zeigen wird, benötigen wir die Parameter beider Kameras, um die Stereokalibrierung ausführen zu können. Die Tiefeninformation wird von der Kinect aus dem IR-Bild gewonnen.

In dem Zusammenhang hat sich beim Testen ergeben, dass die Kalibrierung der IR-Kamera am besten funktioniert, wenn der IR-Projektor abgedeckt wird. Das liegt mit hoher Wahrscheinlichkeit am projizierten Muster, welches zum einen dem Schachbrettmuster ein schwächeres Muster überlagert und zum anderen wegen seiner punktartigen Natur zu starken Intensitätsunterschieden im Bild führt.

Das Abdecken des IR-Projektors verdunkelt die Aufnahme der IR-Kamera jedoch sehr stark. Eine Kalibrierung ist dann nur noch in Tageslicht möglich. Versuche unter Innenraumbedingungen führten dann dazu, dass die OpenCV-Funktion das Schachbrettmuster nur sporadisch erkannte.

5.3 Bildregistrierung

Für das Registrierungsverfahren wurde PCL als Grundlage der Implementierung gewählt.
FloDiEdi hatte hierfür bereits einige Blöcke, von denen `FloDiEdi::displayPointCloud` verwendet wird.

Die Blöcke, die zusätzlich implementiert wurden, werden im Folgenden beschrieben.

5.3.1 FloDiEdi::KinectRgbdToPointCloud

Dieser Block kümmert sich um die Umrechnung der Rohdaten aus dem libfreenect-Treiber in das Format der PCL-Bibliothek (`pcl::PointCloud`). Der wichtigste Schritt besteht in der Umrechnung in metrische Einheiten unter Berücksichtigung der Kameraparameter.
Die Details hierzu wurden im Kapitel *Technische Betrachtung der Kinect-Bilddaten* erörtert.

5.3.2 FloDiEdi::SingularObjectExtractor

Um die Registrierungsalgorithmen testen zu können, musste eine Methode gefunden werden, um die Punktwolke eines Objektes zu erhalten.

Um die Aufgabe zu vereinfachen habe ich eine Tischszene gewählt, die genau ein Objekt enthält. Wie später beschrieben, können nun die Punkte, die zum Tisch gehören, aus der Punktwolke entfernt werden.

Wenn nun zusätzlich nur Punkte berücksichtigt werden, die ungefähr in der Tiefe des Objektes liegen (manueller Parameter), erhält man eine Punktwolke, in der kein Hintergrund vorhanden ist.
Die Punktwolke enthält nun nur noch wenige Punktsammlungen, die als Objekt in Frage kommen können.

Im nächsten Schritt wird mit Hilfe der PCL eine Clusteranalyse ausgeführt, um die verbleibende Punktwolke in Punktwolken von eigenständigen Objekten zu spalten.
Die Clusteranalyse verwendet zur Untersuchung der Zugehörigkeit das Euklidische Abstandsmaß.

In der Implementierung wird auf `pcl::EuclideanClusterExtraction` zurückgegriffen. Die Verwendung lässt sich konfigurieren. Den größten Einfluss auf das

erzielte Ergebnis hatte die Einstellung über die Cluster-Toleranz. Diese Einstellung muss je nach Szene variiert werden.

Aus diesen verbleibenden Punktwolken muss nun per Hand die Punktwolke des zu betrachtenden Objekts herausgesucht werden.
In den meisten Fällen wird man auch einfach die Punktwolke mit den meisten Punkten wählen können.

5.3.3 FloDiEdi::PointCloudAggregator

Die verschiedenen Ansichten eines Objektes, die von `FloDiEdi::SingularObjectExtractor` erfasst werden, müssen nun in die Bildregistrierung gegeben werden.

Nachdem die erste einkommende Punktwolke als Referenz genommen wurde, wird nun sukzessive versucht, eine Transformation zu finden, die die jeweils einkommende Punktwolke in die Referenzpunktwolke überführt. Falls erfolgreich, wird das Ergebnis dieser Ausrichtung als Referenz übernommen und in der nächsten Iteration verwendet.

Das Plugin basiert auf dem ICP-Algorithmus, der im Kapitel *Bildregistrierung* beschrieben wurde.

6 Fazit

In meinen Untersuchungen hat sich gezeigt, dass die Kinect-Kamera prinzipiell gut für die genaue Erfassung von Objekten geeignet ist, obwohl ihr eigentliches Einsatzgebiet diese Anwendung nicht vorsieht.
So sind die Qualität der Tiefenaufnahme und die Auflösung in dem Abstand gut, der sich für die Aufzeichnung mittelgroßer Objekte anbietet.

Eine Software-Lösung, die tatsächlich von einem interessierten Normalverbraucher benutzt werden kann – hierauf wurde ja in der Einführung hingedeutet – muss jedoch noch viele Probleme vereinfachen. So sind derzeit noch sehr viele manuelle Schritte nötig. Nicht nur das Einrichten der verfügbaren Anwendungen an sich gestaltet sich schwierig. Auch bei der Auswahl von Ansichten (bzw. Punktwolken) eines Objekts ist oft noch Handarbeit nötig. Das Ergebnis muss derzeit immer überprüft werden.

7 Literaturverzeichnis

1. **Henry, Peter, et al., et al.** *RGB-D Mapping: Using Depth Cameras for Dense 3D Modeling of indoor Environments.* 2010.
2. *Real-time 3D visual SLAM with a hand-held RGB-D camera.* **Engelhard, Nikolas, et al., et al.** Vasteras, Sweden : s.n., 2011. Proc. of the RGB-D Workshop on 3D Perception in Robotics at the European Robotics Forum.
3. **Konolige, Kurt und Mihelich, Patrick.** Technical description of Kinect calibration. [Online] [Zitat vom: 30. August 2011.] http://www.ros.org/wiki/kinect_calibration/technical.
4. **Burrus, Nicolas.** Kinect Calibration. [Online] [Zitat vom: 30. August 2011.] http://nicolas.burrus.name/index.php/Research/KinectCalibration.
5. **Bradski, Gary und Kaehler, Adrian.** *Learning OpenCV: Computer Vision with the OpenCV.* s.l. : O'Reilly Media, 2008. 0596516134.
6. **Zhang, Zhengyou.** Iterative point matching for registration of free-form curves and surfaces. *International Journal of Computer Vision.* 1994, Bd. 13, 2.
7. **Rusu, Radu Bogdan.** *Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments.*
8. **Fioaraio, Nicola und Konolige, Kurt.** *Realtime Visual and Point Cloud SLAM.*
9. **Chetverikov, D., et al., et al.** *The Trimmed Iterative Closest Point Algorithm.*
10. **Rusinkiewicz, Szymon und Levoy, Marc.** *Efficient Variants of the ICP Algorithm.*
11. *Real-Time Scale Invariant 3D Range Point Cloud.* **Sehgal, Anuj, Cerna, Daniel und Makaveeva, Milena.** Povoa de Varzim, Portugal : s.n., 2010. International Conference on Image Analysis and Recognition.
12. **Fischler, Martin A., Bolles und C., Robert.** Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM .* 6, 1981, Bd. 24.
13. **Bay, Herbert, Tuytelaars, Tinne und Van Gool, Luc.** *Speeded up robust features (SURF).* 2006.
14. **Thomas, Stephen J.** *Real-time Stereo Visual SLAM.*
15. ROS (Robot Operating System). [Online] [Zitat vom: 30. August 2011.] http://en.wikipedia.org/wiki/ROS_(Robot_Operating_System).
16. **Besl, Paul J. und McKay, Neil D.** A Method for Registration of 3-D Shapes. *IEEE Transactions On Pattern Analysis and Machine Intelligence.* Februar 1992, Bd. 14, 2.