EBERHARD-KARLS-UNIVERSITÄT TÜBINGEN
Wilhelm-Schickard-Institut für Informatik
Lehrstuhl Kognitive Systeme

Bachelorarbeit

Fusion von 3D-Laserscanner-Daten mit 2D-Bilddaten

Richard Hanten

Gutachter:	Prof. Dr. rer. nat. Andreas Zell Wilhelm-Schickard-Institut für Informatik
Betreuer:	Dipl.-Inform. Stefan Laible Wilhelm-Schickard-Institut für Informatik
Begonnen am:	01. Mai 2010
Beendet am:	17. August 2011

Erklärung

Hiermit versichere ich, diese Arbeit selbstständig verfasst und nur die angegebenen Quellen benutzt zu haben.

Tübingen am 17. August 2011

_____

Richard Hanten

Kurzfassung. 3D-Sensoren gewinnen heutzutage immer mehr an Interesse und Bedeutung für den Einsatz auf mobilen, autonomen Robotern. Sie sind hervorragend für Aufgaben, wie Objekt- und Hinderniserkennung oder Selbstlokalisierung und Mapping (SLAM), geeignet. Nicht alle Sensortypen sind in beliebigen Umgebungen verwendbar. 3D-Laserscanner arbeiten allerdings unter den meisten Bedingungen zuverlässig und robust. Das Problem bei diesen Sensoren ist jedoch, dass sie in der Regel nur gering aufgelöste Daten liefern. Um die Auflösung dieser Daten zu verbessern, werden in dieser Arbeit verschiedene Interpolationsverfahren unter Benutzung von Bildern einer Farbkamera vorgestellt und implementiert.

Inhaltsverzeichnis

1 Einführung

Bei autonomen, mobilen Robotern ist heutzutage der Einsatz von 2D-Sensoren, beispielsweise von 2D-Laserscannern, beinahe Standard. Ohne diese Sensoren wäre der Roboter nicht im Stande dazu seine Umgebung wahr zu nehmen und könnte seinen autonomen Aufgaben nicht gerecht werden. Sie ermöglichen ihm frühzeitig Hindernissen auszuweichen und seine eigene Position in der Umwelt zu bestimmen.

Zu den elementarsten Aufgaben eines autonomen, mobilen Roboters zählt zum Beispiel die Erstellung einer Karte von verschiedensten Orten. Es existieren allerdings auch erweiterte Aufgaben wie die Klassifikation von Objekten jeglicher Art.

Komplexere Aufgaben sind mit zweidimensionalen Sensoren meist schwieriger zu realisieren, weswegen 3D-Sensoren immer mehr an Interesse gewinnen. Wie es die Bezeichnung schon aussagt, kann ein 2D-Sensor nur eine Ebene der Umgebung zu einem Zeitpunkt erfassen. Da einem 3D-Sensor eine Dimension zusätzlich zur Verfügung steht, kann er über die Umgebung weitere wichtige Daten ermitteln. Um dies mit einem zweidimensionalen Sensor realisieren zu können, müsste dieser bewegt werden, was nicht immer von Vorteil ist.

3D-Sensoren unterliegen bei ihrer Verwendung genauso wie 2D-Sensoren bestimmten Beschränkungen bezüglich den umweltbedingten Gegebenheiten. Bei beiden Kategorien von Sensoren liegen häufig die gleichen Verfahren zur Ermittlung der Messdaten zugrunde. Soll ein Roboter in jeder Umgebung gut operieren können, ist es sinnvoll einen Sensor zu wählen, der unter allen Bedingungen zuverlässig funktioniert.

1.1 Verfügbare Sensoren und Techniken

Für die Messung dreidimensionaler Umgebungsdaten stehen einige Techniken und damit auch eine Anzahl verschiedener Sensoren zur Verfügung. Diese stützen sich dabei zum Beispiel auf optische Verfahren oder Pulslaufzeitmessungen.

Visuelle Verfahren

Ein Beispiel für ein optisches Verfahren zur Berechnung von dreidimensionalen Umgebungsdaten ist die sogenannte Stereo-Vision. Das Verfahren besitzt Ähnlichkeit zur menschlichen Tiefenwahrnehmung. Häufig werden hierfür zwei identische Kameras benutzt, die sich in fester Position zu einander befinden.

Der Abstand und die Winkel von einem zum anderen Aufnahmegerät müssen bestimmt werden und sind für die Berechnung der 3D-Daten grundlegend. Ausserdem muss jede Kamera kalibriert werden, damit Abweichungen in der Aufnahme festgestellt und korrigiert werden können (Abschnitt 3.2).

Sind diese Vorbereitungen getroffen, können über Abweichungen in den Bildern und mithilfe von Triangulierung die 3D-Daten berechnet werden. Die Distanz eines Objektes ist dabei invers proportional zur Abweichung in den Aufnahmen [BK08a].

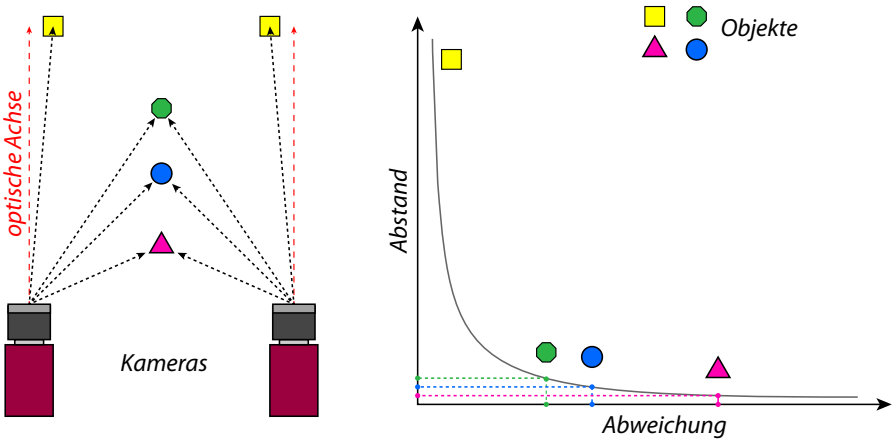

Abbildung 1.1: Stereo-Vision

Pulslaufzeitmessungen

Bei dieser Art Verfahren wird ein Lichtimpuls ausgesendet und wieder aufgenommen. Der Impuls wird von den in der Umgebung vorkommenden Objekten reflektiert. Um die Distanz zu berechnen wird schließlich die Laufzeit ermittelt. Dieses Verfahren wird von 2D- und 3D-Laserscannern sowie von sogenannten *PMD-Kameras* genutzt. PMD steht hierbei für Photo-Misch-Detektor.

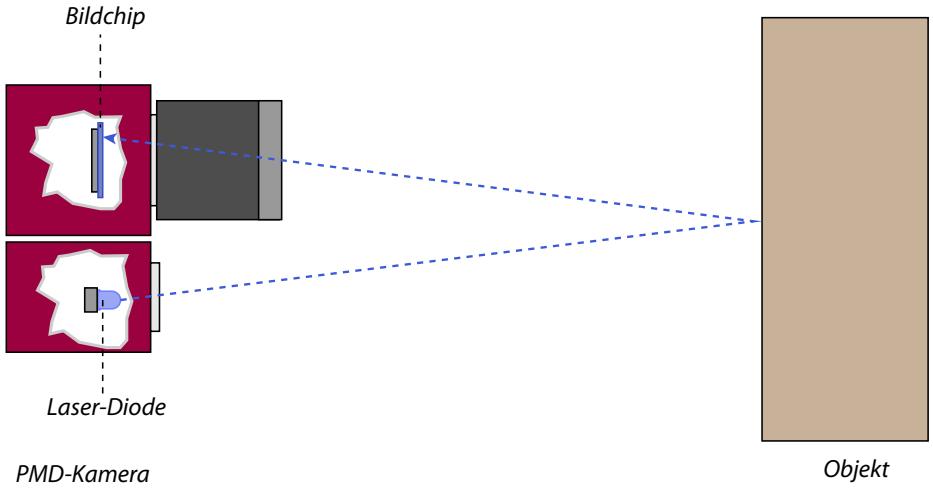

Abbildung 1.2: Tiefenmessung mit einer TOF-Kamera

Das für den Puls verwendete Licht hat dabei eine bestimmte Wellenlänge, beispielsweise im Bereich von Infrarot. Dies hat den Sinn, dass der Impuls aus dem restlichen Umgebungslicht

isoliert werden kann, um Störungen vorzubeugen. Ausgesandt wird der Impuls durch LEDs oder Laserdioden.

Ähnlich einer normalen Kamera besitzt eine PMD-Kamera einen Bildchip. Dieser ist allerdings speziell auf die für den Puls verwendete Wellenlänge angepasst. Die Kamera berechnet den Tiefenwert für jeden Bildpunkt des Sensors einzeln. Aufgrund des komplexeren Aufbaus des Bildchips für dieses Verfahren ist die Auflösung von PMD-Kameras im Moment relativ gering [Wikb].

Laserscanner können sich von der PMD-Kamera wesentlich unterscheiden. Laserscanner besitzen häufig nur eine einzelne Photodiode, die den ausgesandten Puls wieder aufnimmt. Der Bildchip einer PMD-Kamera wird in mehrere Bildpunkte unterteilt, die jeder für sich eine Art Photodiode bilden. Bei Laserscannern existiert häufig eine Spiegeloptik. Häufig werden die Laserdiode und die Photodiode rotiert, um die Abtastung der Umgebung zu ermöglichen.

Mit der Kinect von Microsoft kommt ein weiterer Sensor hinzu. Bei dieser wird statt einem PMD-Sensor oder Photodioden eine normale Schwarz-Weiß-Kamera verwendet. Hinzu kommt eine Infrarot-Projektor, der auf die Umgebung ein Muster überträgt. Nachdem die Schwarz-Weiß-Kamera das abgebildete Muster aufgenommen hat, kann sie die Verzerrungen und Verschiebungen in diesem berechnen und damit die Tiefeninformationen ermitteln.

Vergleich der Sensoren

Die Verfahren haben verschiedene Vor- und Nachteile. Zunächst spielen Größe und Gewicht eines Sensors eine enorme wichtige Rolle für den Einsatz auf einem bestimmten autonomen mobilen Roboter. Die Verwendung von großen Messinstrumenten auf kleinen mobilen Robotern ist eher als unwahrscheinlich anzusehen.

Neben der Größe sind weitere Punkte zu betrachten. Dazu gehören unter anderem die Rechenleistung für die Auswertung und die Störempfindlichkeit.

Betrachtet man beispielsweise ein Stereo-Kamera-System, dann ist es recht einfach zu verwirklichen. Werden zudem kleine, leistungsfähige Kameras verwendet, ist die Gesamtgröße entsprechend klein. Allerdings ist die Ermittlung der Messdaten relativ aufwendig. Es müssen schließlich zwei eventuell hochaufgelöste Bilder verglichen und auf Unterschiede überprüft werden. Anschließend müssen die Tiefenwerte über eben diese berechnet werden. Normalweise hält sich die Leistung zugunsten der Laufzeit in Grenzen.

Ein weiterer Faktor, der bei der Auswertung eine wichtige Rolle spielt, ist die Qualität der Komponenten. So können durch schlechte Optik und mäßige Verarbeitung wesentliche Messfehler produziert werden (siehe Abschnitt 3.2). Die Ausrichtung der Kameras zu einander ist elementar und darf deswegen auch nicht vernachlässigt werden.

Im Bezug auf die Störempfindlichkeit hat ein Stereo-Kamera-System einen Vorteil. Die Ermittlung der Tiefendaten stützt sich nicht, wie bei einer PMD-Kamera oder einem Laserscanner, auf Licht einer bestimmten Wellenlänge. Für die Berechnung können direkt Farbbilder verwendet werden, also werden alle Wellenlängen verwendet und nicht eine Wellenlänge, die gestört werden kann. Die einzige Schwachstelle ist, wenn es zu dunkel werden sollte. Die Kamera kann in dieser Situation keine sinnvollen Bilddaten mehr liefern. Wenn es sehr hell wird, hat das System normalerweise kein Problem, da die meisten Kameras eine Blende besitzen und sich die Helligkeit anpassen lässt.

Bei PMD-Kameras und 3D-Laserscannern muss die Tiefe nicht erst über den Vergleich verschiedener Bilder berechnet werden. Die Messdaten müssen in der Regel weniger komplex berechnet werden wie beim Stereo-Kamera-System. Da der Kamerachip, der PMD-Kameras einen sehr komplexen Aufbau hat, ist ihre Auflösung aktuell noch beschränkt. Laserscanner können hingegen ganz unterschiedliche Auflösungen erreichen. Höhere Auflösungen stehen häufig in Zusammenhang mit zunehmender Größe der Geräte und sind meist mit höheren Kosten verbunden. Daher muss abgewägt werden, welches Gerät in Größe, Auflösung und Kostenpunkt optimal für die Anwendung ist.

Eigentlich sind die Einsatzbedingungen beider Geräte kaum beschränkt, da beide mit einer eigenen Lichtquelle arbeiten und nicht auf natürliches Licht oder andere Beleuchtungen angewiesen sind. Bei PMD-Kameras besteht allerdings das Problem, dass trotz ausreichender Filterung das Licht überlagert werden kann und damit unbrauchbare Messwerte ermittelt werden. Laserscanner benutzen häufig nur eine oder wenige Photodioden und sind diesem Problem gegenüber wesentlich robuster.

Mit dem Kinect Sensor von Microsoft ist ein sehr kostengünstiges Gerät erhältlich geworden, dass eine relativ hohe Auflösung zur Verfügung stellt. Wie bereits erwähnt, werden hier ein Infrarotprojektor und ein Schwarz-Weiß-Kamera-Chip in Verwendung gebracht. Dieser liefert dabei normal VGA-Auflösung, also 640 mal 480 Bildpunkte. Der Aufwand, der bei der Kinect zur Ermittlung der Umgebungsdaten betrieben werden muss, ist nicht höher als bei Laserscannern und PMD-Kameras. Es ist des Weiteren ohne Probleme möglich, den relativ kleinen, leichten Sensor auf ein mobiles Robotersystem zu montieren, wie in Abb. 1.3. Ein großer und leider kaum zu korrigierender Nachteil der Kinect ist jedoch, dass sie beim Einsatz im Aussenbereich durch einfallendes Sonnenlicht gestört wird und keine sinnvollen Werte mehr zu liefern vermag.

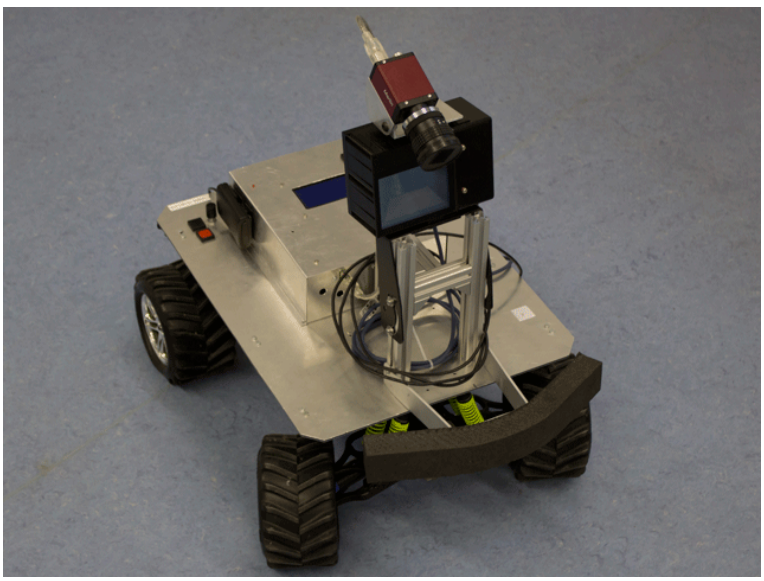

Abbildung 1.3: Outdoor-Roboter des Lehrstuhls Kognitive Systeme - Universität Tübingen

1.2 Ziel dieser Arbeit

Wie sich im Vergleich der verschiedenen Verfahrensweisen und Gerätetypen zur Ermittlung von dreidimensionalen Messdaten gezeigt hat, sind 3D-Laserscanner sehr zuverlässige Messinstrumente. Sie arbeiten, unter nahezu allen umweltbedingten Umständen, verlässlich und robust.

Nachteil ist allerdings, dass die meisten, für mobile Roboter verwendbaren Geräte eine zu gering aufgelöste Datenmenge zur Verfügung stellen. Kann ein 3D-Laserscanner die Umgebung höher auflösen, ist er in der Regel zu groß für das Robotersystem oder unbezahlbar.

Das Ziel dieser Arbeit ist es einen geeigneten 3D-Laserscanner mit einer VGA-Farbkamera so zu fusionieren, dass der Nachteil der geringen Auflösung ausgeglichen werden kann.

Die Fusion soll dabei über verschiedene bildgestützte Interpolationsverfahren realisiert werden. Anschließend sollen die Verfahren auf ihre Genauigkeit und Echtzeitfähigkeit überprüft werden. Die Implementierung soll für Benutzung mit ROS (*Robot Operating System*), einer gängigen Framework für Roboter, durchgeführt werden.

1.3 Gliederung der Arbeit

Die für diese Arbeit relevanten Themen werden in acht Kapiteln abgedeckt. In Kapitel 2 werden zunächst einige, allgemeine Informationen zu verwendeten Bibliotheken und der Rahmensoftware ROS geliefert. Kapitel 3 beschäftigt sich mit der Fusion von Kamera- und 3D-Laserscannerdaten. Folgend, werden in Kapitel 4 Grundlagen für die Interpolation von Tiefenwerten besprochen. In Kapitel 5 werden schließlich die Interpolationsverfahren vorgestellt. Kapitel 6 beschäftigt sich mit der Implementierung der für die Interpolationen entwickelten Software. In Kampitel 7 wird die Genauigkeit und die Laufzeit der Algorithmen ausgewertet. Abschließend wird in Kapitel 8 zusammengefasst und ein Ausblick gegeben.

2 Grundbausteine der Implementierung

Die Grundbausteine der Implementierung dieser Arbeit bilden verschiedene Softwarepakete. Unter diesen befindet sich das Robot Operating System, kurz ROS [ROSa]. Beinahe jedes auf einem Standard-Computer basierende Robotersystem lässt sich mit dieser Framework betreiben.

Neben ROS finden weitere, unterschiedliche Bibliotheken Verwendung. *OpenCV* ist eine von diesen Bibliotheken uns stellt eine Vielzahl von Algorithmen zur Bildverarbeitung zur Verfügung.

Eine weitere Bibliothek ist die *Point Cloud Library (PCL)*. Mit dieser Sammlung von Algorithmen und Datenstrukturen ist es möglich Mengen von Punkten unterschiedlicher Dimensionen zu bearbeiten, beispielsweise Messpunkte von 3D-Sensoren.

Das letzte große Softwarepaket bildet *CGAL*, die *Computational Geometry Algorithms Library*. In dieser Bibliothek befinden sich einige Algorithmen und Strukturen zur Verarbeitung geometrischer Probleme.

Sicher gibt es bei den Bibliotheken einige Bereiche, in denen sich ihre Fähigkeiten überschneiden, dennoch sind sie je nach Anwendung besser oder schlechter geeignet.

2.1 ROS - Robot Operating System

Das Robot Operating System bildet die Schnittstelle zwischen Software und Roboter. Bei ROS handelt es sich nicht um ein Betriebsystem für den Roboter. ROS ist eine Softwareumgebung, in der Treiber für die Ansteuerung für Sensoren und Aktuatoren eines Roboters enthalten sind. Zu dem gibt es bereits Programme für viele verschiedene Zwecke.

Das Besondere an ROS ist, dass die interne Software-Architektur wie ein großer Graph aufgebaut ist. Jeder Treiber und jedes Programm wird als Knoten (*Node*) in diesen Graphen aufgenommen.

Damit die Architektur lauffähig wird benötigt man den sogenannten *ROS Master*. Alle Knoten müssen sich bei diesem Prozess anmelden. Ohne diesen Prozess wäre keine interne Kommunikation möglich. Im ROS Master integriert ist ein *Parameter Server* über den die einzelnen Knoten Einstellungen und Werte beziehen können. Das für die Kommunikation verwendete Protokoll nennt sich *TCPROS* und verwendet ganz normale TCP/IP Sockel.

Die Kommunikation unter den Knoten wird über sogenannte *Topics* ermöglicht. Ein Topic ist dabei eine Art Nachrichtenkanal über den Datennachrichten gesendet werden. Nachrichten kapseln dabei einen bestimmten Datentyp und diverse Informationen. Welche Topics zur Verfügung stehen, ist dem ROS Master bekannt.

Soll ein Knoten beziehungsweise ein Teilprogramm bestimmte Daten aufnehmen, muss er hierfür ein Topic abonnieren. Wenn verarbeitete oder aufgenommene Daten den restlichen Programmen zur Verfügung gestellt werden, dann muss ein Knoten diese Daten als Nachricht auf einem bestimmten Topic, also Kanal, publizieren. Diese Daten können dann, wie beschrieben, wieder aufgenommen werden. Die Anzahl der abonnierten sowie publizierten Topics ist nicht beschränkt [ROSb].

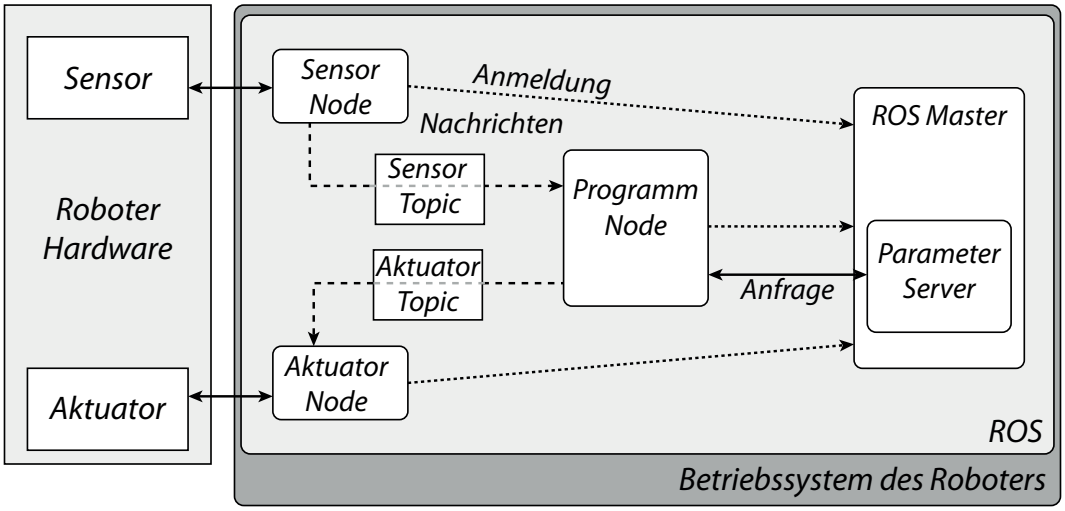

Abbildung 2.1: Kommunikation zwischen Knoten, Hardware und dem ROS Master

2.2 OpenCV

Bei OpenCV handelt es sich um eine freie Bibliothek für Bildverarbeitung, die ursprünglich von *Intel* entwickelt wurde. Im Bereich der Computer Vision stellt OpenCV eines der wichtigsten Softwarepakte dar und ist sehr umfangreich. Der Name OpenCV steht tatsächlich für Open Computer Vision.

Die Algorithmen dieser Bibliothek sind für die unterschiedlichsten Aufgaben im Bereich der Bildverarbeitung entwickelt. Eine mögliche Aufgabe, die mit OpenCV gelöst werden kann, ist beispielsweise die bereits erwähnte Berechnung von 3D-Punkten aus visuellen Daten (Kapitel 1 Abschnitt 1.1).

Die Menge der Aufgaben in der Bildverarbeitung, die mit OpenCV gemeistert werden können ist groß. Einige dieser Aufgaben werden im Verlauf der Arbeit vorgestellt.

2.3 PCL - Point Cloud Library

Bei der PCL handelt es sich ebenfalls um eine freie Bibliothek. Die Bibliothek war ursprünglich ein Projekt des ROS, ist mittlerweile aber eigenständig. Die PCL wurde speziell für die Verarbeitung von Mengen dreidimensionaler Punkte entwickelt, ist aber auch im Stande mit anderen Dimensionen umgehen zu können. Diese Mengen von Punkten werden auch als Punktwolken bezeichnet.

Die PCL stellt verschiedenste Algorithmen für die diese Datenmengen zur Verfügung. Mit der PCL ist es zum Beispiel möglich Punktwolken effektiv zu segmentieren. Dies stellt aber nur einen kleinen Bruchteil von den eigentlichen Fähigkeiten dieser Bibliothek dar.

Speziell bei der Verwendung von 3D-Sensoren unter ROS stellt die PCL einen elementaren Bestandteil für die Verarbeitung der Messdaten dar. Etliche, dort definierte Datentypen und Strukturen werden in ROS genutzt [RC11].

2.4 CGAL - Computational Geometry Algorithms Library

Das letzte große Softwarepaket, was bei der Implementierung dieser Arbeit zum Einsatz kommt, ist CGAL. CGAL ist eine Bibliothek für die Lösung einer enormen Menge geometrischer Probleme.

Dabei deckt CGAL teilweise Problembereiche ab, für die bereits Lösungen in der PCL oder in OpenCV enthalten sind. CGAL bietet jedoch weitere Algorithmen für bestimmte Probleme an, die weder von OpenCV noch von der PCL gelöst werden können.

Bei der Implementierung dieser Arbeit war genau dies der Fall [cga].

3 Grundlagen zur Fusion von Kamera- und 3D-Laserscanner-Daten

3.1 Fusion von 3D-Laserscanner und Kamera

Bevor mit den über 3D-Laserscanner und Kamera aufgenommenen Daten gearbeitet werden kann, müssen diese zunächst in Beziehung zu einander gebracht werden. Die später beschriebenen Interpolationsverfahren arbeiten mit Pixelkoordinaten. Diese beschreiben die Position eines Pixels auf der Bildebene.

Transformation zwischen Koordinatensystemen

Üblicherweise liefert der Treiber für einen 3D-Laserscanner Punktwolken mit 3D-Positionen. Der Ursprung des Koordinatensystems, in dem die Punkte dargestellt sind, ist gleichzeitig die Position des Lasersensors. Der erste Arbeitsschritt zur Vereinigung von Kamera- und Laserdaten ist also zunächst die Transformation vom Laserscanner-Koordinatensystem in das der Kamera.

Um diese Abbildung auszuführen sind für gewöhnlich eine Rotationsmatrix und ein Translationsvektor nötig.
Zunächst werden die Punkte um den Ursprung des Laserscanner-Koordinatensystems mithilfe der Matrix rotiert. Anschließend sorgt die Translation dafür, dass die Punkte an die korrekte Position im Kamera-Koordinatensystem verschoben werden. Damit ist die Darstellung in diesem Koordinatensystem gültig.

Berechnung der Transformation

Die Transformation für die genannten 3D-Punkte hängt letztendlich vom Aufbau der Sensorik auf dem Roboter ab. Daher gibt es zwei Möglichkeiten, wie man die Transformation erhalten kann. Entweder sind die genauen Dimensionen sowie die Winkel im Aufbau bekannt oder man muss über ein Programm eine Kalibrierung von Kamera und Laserscanner durchführen.

Im Zuge dieser Arbeit wurde auf die zweite Methode zurückgegriffen und mit der *Matlab Calibration Toolbox* umgesetzt. Das Programm arbeitet dabei mit einem Schachbrett, einer sogenannten *planaren Struktur*. Ausser der Berechnung der Transformation von Laserscanner zu Kamera, werden Eigenschaften der Kamera bestimmt [matb]. Das Besondere an einem Schachbrett ist, dass es aus vielen, kleinen Quadraten besteht, die eine reguläre Struktur bilden. Anhand dieser Struktur lassen sich Verzerrungen durch die Linse leicht erkennen und können durch die Toolbox errechnet werden. Ausserdem werden Berechnungen über die interne Geo-

metrie angestellt. Die Eigenschaften der internen Geometrie und die Verzerrungskoeffizienten der Linse bilden die *intrisischen* Parameter. Diese Parameter werden für eine genaue Berechnung der Position und Abbildung der Punkte eines Objektes gebraucht.

Für die Bestimmung der Transformation von Laserscanner zu Kamera werden verschiedene Referenzframes verwendet. Genauere Informationen über den gesamten Vorgang sind in der *Matlab Calibration Toolbox Dokumentation* nachzulesen [mata].

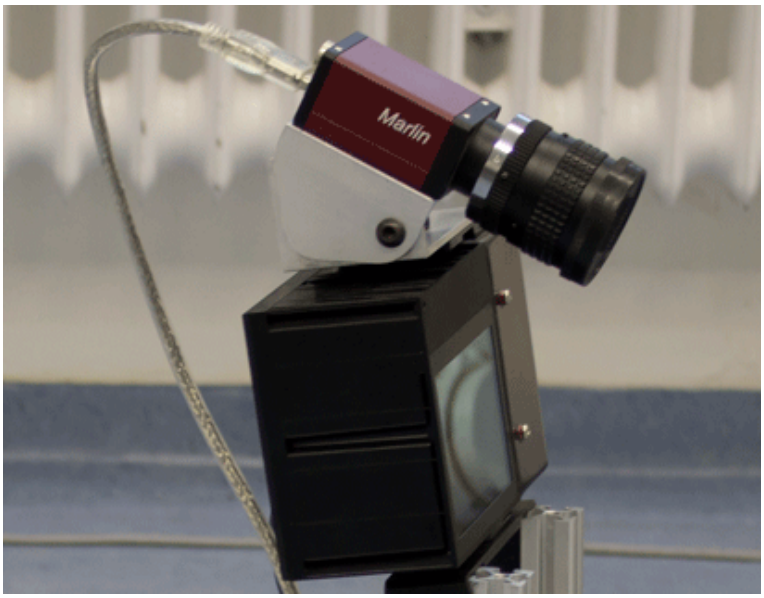

Abbildung 3.1: Anordnung von 3D-Laserscanner und Kamera

3.2 Kameramodell und intrinsische Kameraparameter

Ideales Kameramodell

Das hier relevante Kameramodell leitet sich aus dem Modell einer idealen Lochkamera ab. Die Umstellung des Lochkameramodells auf diese Form bewirkt, dass die Berechnungen für Projektionen einfacher werden. Das umgeformte Modell bleibt zum Lochkameramodell gleichwertig [BK08b]. Im Wesentlichen wird das genutzte *ideale Kameramodell* durch das Kamerakoordinatensystem und einer *Bildebene* beschrieben. Die Bildebene hat dabei die Brennweite f als Abstand zum Ursprung. Die Brennweite ist der Abstand zwischen Objektiv beziehungsweise Linse und dem digitalen Aufnahmechip der verwendeten Kamera.

Die z-Achse des Kamerakoordinatensystems, die auch *optische Achse* genannt wird schneidet den *Bildbereich* der Kamera in seinem Mittelpunkt, dem *Bildmittelpunkt* (siehe Abb. 3.2). Der Bildbereich ist rechteckig und liegt in der Bildebene. In ihm liegen alle für die Kamera sichtbaren Punkte. Die x-Achse läuft entlang der Breite und die y-Achse entlang der Höhe dieser Ebene.

Der Ursprung des Kamerakoordinatensystems o bildet das Projektionszentrum. Möchte man einen Punkt auf die Bildebene projezieren, dann muss eine Gerade durch das Projektionszentrum und den abzubildenden Punkt gelegt werden. Erhält man anschließend einen Schnittpunkt

mit der Bildebene, dann stellt eben dieser die Projektion dar. Wird die Ebene nicht geschnitten, existiert auch keine Projektion. Liegt die Projektion ausserhalb des Bildbereiches, dann ist diese für die Kamera nicht sichtbar.

Im idealen Kameramodell ist es möglich eine Beziehung zwischen einem Punkt P und seiner Projektion p auf der Bildebene herzustellen.

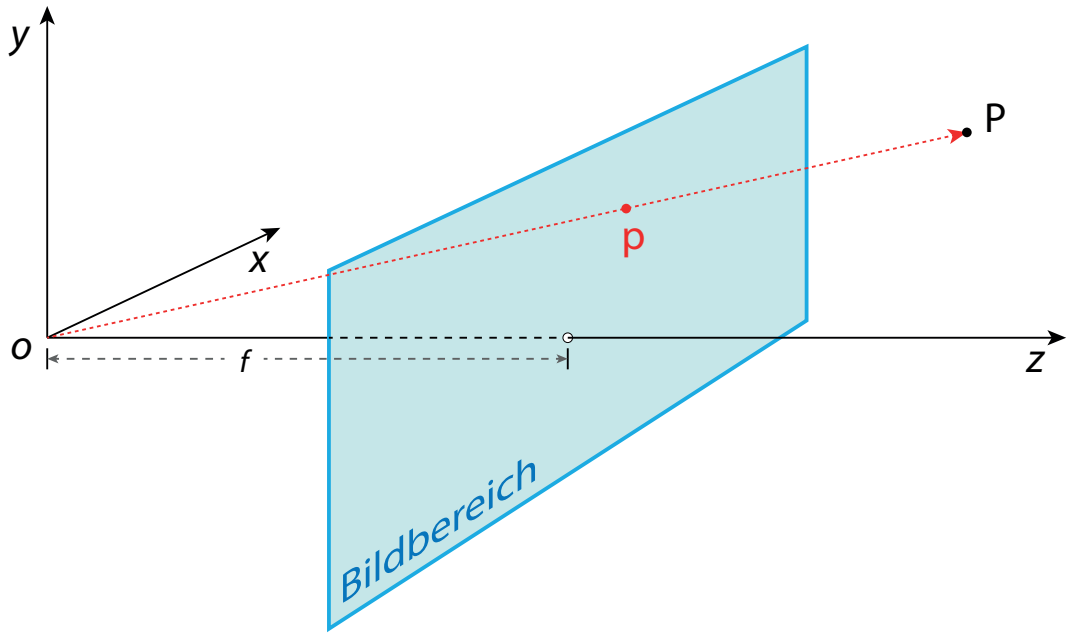

Abbildung 3.2: Kameramodell mit Projektion $p(x'_p, y'_p, f)^T$ eines Punktes $P(x_p, y_p, z_p)^T$.

Sei $P = (x_p, y_p, z_p)^T$ und seine Projektion $p = (x'_p, y'_p, f)^T$. Die z-Koordinate ist logischerweise gleich der Brennweite f, da die Bildebene den Abstand f zum Ursprung des Kamerakoordinatensystems hat. Über den Strahlensatz lässt sich dann die Beziehung zwischen Punkt und Projektion wie folgt charakterisieren:

- Für die x-Koordinate von Punkt und Projektion gilt:

$$\frac{x'_p}{f} = \frac{x_p}{z_p} \Rightarrow x'_p = \frac{x_p}{z_p} \cdot f \tag{3.1}$$

- Für die y-Koordinate von Punkt und Projektion gilt:

$$\frac{y'_p}{f} = \frac{y_p}{z_p} \Rightarrow y'_p = \frac{y_p}{z_p} \cdot f \tag{3.2}$$

Angepasstes Kameramodell

Das ideale Kameramodell stellt nur die Grundlage für das Modell der tatsächlichen Kamera dar. Es ist davon auszugehen, dass keine perfekte Kamera produziert werden kann. In irgendeiner Form werden immer wieder Abweichungen auftreten, zum Beispiel der Versatz des bildgebenden Mikrochips. Für das oben beschrieben Modell bedeutet dies, dass die optische Achse den

Bildbereich in der Bildebene nicht mehr perfekt im Bildmittelpunkt schneidet. Das Kameramodell kann in dieser Hinsicht über zwei Parameter c_x und c_y angepasst werden. Die beiden Parameter beschreiben jeweils den Versatz in y-Richtung und x-Richtung.

Ein weiterer Punkt ist, dass bei vielen Kameras kein quadratischer, bildgebender Mikrochip verbaut ist. In das Modell müssen daher zwei verschiedene Brennweiten f_x und f_y übernommen werden. [BK08c]

Die Beziehung zwischen Punkt P und seiner Projektion p lautet dann wie folgt:

- Für die x-Koordinate von Punkt und Projektion gilt:

$$x'_p = \frac{x_p}{z_p} \cdot f_x + c_x \tag{3.3}$$

- Für die y-Koordinate von Punkt und Projektion gilt:

$$y'_p = \frac{y_p}{z_p} \cdot f_y + c_y \tag{3.4}$$

Diese Beziehungen lassen sich in einer für die Kamera spezifischen Matrix festhalten, *der Kameramatrix*. Diese kann direkt vom Bibliotheken wie OpenCV benutzt werden, was bei der Implementierung auch der Fall ist.

Die Darstellung hat für alle Punkte P und deren Projektion p folgende Form:

$$p = \begin{pmatrix} x'_p \\ y'_p \\ 1 \end{pmatrix} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \frac{x_p}{z_p} \\ \frac{y_p}{z_p} \\ 1 \end{pmatrix} \tag{3.5}$$

Diese Darstellung verwendet homogene Koordinaten. Genauer bedeutet das für den Punkt $P = (x_p, y_p, z_p)^T$, dass er sich über Normierung durch z_p in die Form $P = (\frac{x_p}{z_p}, \frac{y_p}{z_p}, 1)^T$ bringen lässt. Ist dies getan, lässt sich die Kameramatrix für die Projektion offensichtlich direkt anwenden [BK08c]. Alternativ zu c_x und c_y lässt sich auch die Position des *Bildmittelpunktes* angeben, da die beiden Parameter unmittelbar von diesem abhängen.

Mit der Kameramatrix und damit mit den Parametern c_x, c_y, f_x, f_y ist bereits ein Teil der *intrinsischen* Parameter abgedeckt.

Verzerrungskoeffizienten

Im Abschnitt 3.1 wird verdeutlicht, dass die Verzerrung durch die Kameralinse eine wichtige Rolle spielt. Es gibt allerdings auch Verzerrungen, die nicht durch die Linse hervorgerufen werden. Die Eigenschaften der auftretenden Verzerrungen sind in den *Verzerrungskoeffizienten* festgehalten.

Bei vielen Kameras ist die Verzerrung deutlicher, bei anderen weniger deutlich zu erkennen. Tatsache ist, dass sie bei jeder Kamera auftritt. Im Grunde kommen Verzerrungen deswegen zu Stande, dass es nicht möglich ist eine perfekte Linse oder perfekte Anordnung der Bauteile in einer Kamera zu schaffen.

Abbildung 3.3: Unterschiedlich starke Verzerrung bei verschiedenen Kameras

Die Verzerrungskoeffizienten, die für eine korrekte Berechnung der Projektion eines Punktes benötigt werden, beziehen sich daher auf die *radiale* und auf die *tangentiale* Verzerrung [BK08d].

Die radiale Verzerrung beschreibt die Verzerrung, die durch die ungleichmäßige Krümmung der Linse entsteht. Die tangentiale Verzerrung bezieht sich auf die Ausrichtung des bildgebenden Mikrochips der Kamera.

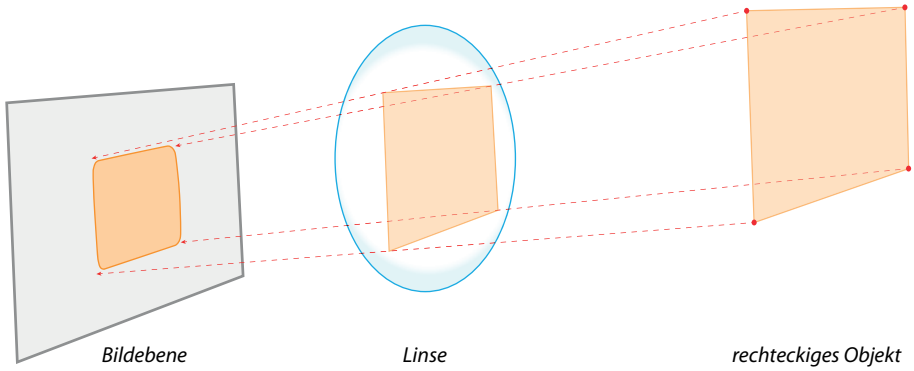

Abbildung 3.4: Radiale Verzerrung bei der Projektion eines viereckigen Objekts durch die Linse

Der Effekt der durch die radiale Verzerrung auftritt wird anhand der Abbildung 3.4 deutlich. Man kann annehmen, dass die Verzerrung am Ursprung der optischen Achse, auf dem Bildgeber, gleich 0 ist. Dieser Punkt liegt direkt unter der Mitte der Linse. Alle Punkte, die weiter aussen liegen werden in der Regel mehr verzerrt. Um eine Korrektur der nach aussen hin zunehmenden Verzerrung berechnen zu können, wird eine Annäherung über die Taylor-Entwicklung durchgeführt Dabei werden die ersten Terme dieser Entwicklung genutzt [BK08d]. Die Berechnung der Korrektur für eine Projektion p sieht wie folgt aus:

- Für die x-Koordinate von Punkt und Projektion gilt:

$$x'_{p\ korrigiert} = x'_p \cdot (1 + k_1 \cdot r^2 + k_2 \cdot r^4 + k_3 \cdot r^6) \tag{3.6}$$

- Für die y-Koordinate von Punkt und Projektion gilt:

$$y'_{p\ korrigiert} = y'_p \cdot (1 + k_1 \cdot r^2 + k_2 \cdot r^4 + k_3 \cdot r^6) \tag{3.7}$$

Bei k_1, k_2 und k_3 handelt es sich um bereits für die Kamera errechnete intrinsische Parameter.

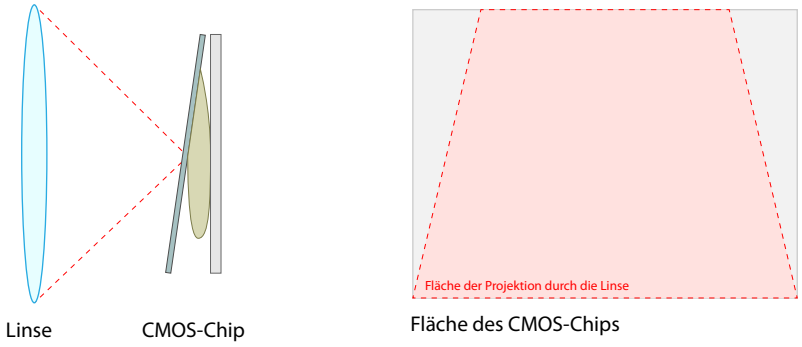

Abbildung 3.5: links: in einer Kamera verbauter Bildgeber mit Linse, nicht gerade geklebt
rechts: auf dem Bildgeber ankommende Projektion durch die Linse

Der durch die tangentiale Verzerrung hervorgerufene Effekt kann dadurch charakterisiert werden, dass das Bild, das vom Sensor aufgenommen wird, nicht rechteckig, sondern trapezförmig wird. Es ist genau dann der Fall, wenn der Sensor nicht parallel zur angebracht Linse ist, ausgelöst durch einen Produktionsdefekt (siehe Abb. 3.5) [BK08d]. Ähnlich wie bei der radialen Verzerrung kann auch hier eine Korrekturformel angewendet werden. Treten beide Verzerrungstypen auf, müssen diese Formeln verrechnet werden. Beim Parameter r handelt es sich bei dieser Korrekturformel um den selben, wie bei der ersten Korrektur. Für die Korrektur der beiden Koordinaten x'_p, y'_p einer Projektion p müssen folgende Berechnungen angestellt werden:

- Für die x-Koordinate von Punkt und Projektion gilt:

$$x'_{p\ korrigiert} = x'_p \cdot (2p_1 y'_p + p_2(r^2 + 2(x'_p)^2)) \tag{3.8}$$

- Für die y-Koordinate von Punkt und Projektion gilt:

$$y'_{p\ korrigiert} = y'_p \cdot (p_1(r^2 + 2(y'_p)^2) + 2p_2 x'_p) \tag{3.9}$$

Die Parameter p_1 und p_2 sind, wie im Fall der ersten Korrektur k_1, k_2 und k_3, intrinsische Parameter, die vor Verwendung der Korrekturformel bekannt sein müssen.

Zusammenfassend gesagt, bestehen die intrinsischen Parameter einer Kamera aus einer Projektionsmatrix und den fünf Koeffizienten, die die Verzerrung beschreiben. Ohne diese Parameter ist eine sinnvolle Projektion von Punkten auf ein Bild nicht zu berechnen.

3.3 Abbildung Messpunkte auf die Bildebene

Nachdem die über den 3D-Lasersensor aufgenommenen Punkte über eine Transformation in das Kamerakoordinatensystem übertragen worden sind, können diese auf die Bildebene projeziert werden. Zur Durchführung der Projektion wird die Kameramatrix benutzt. Im Regelfall

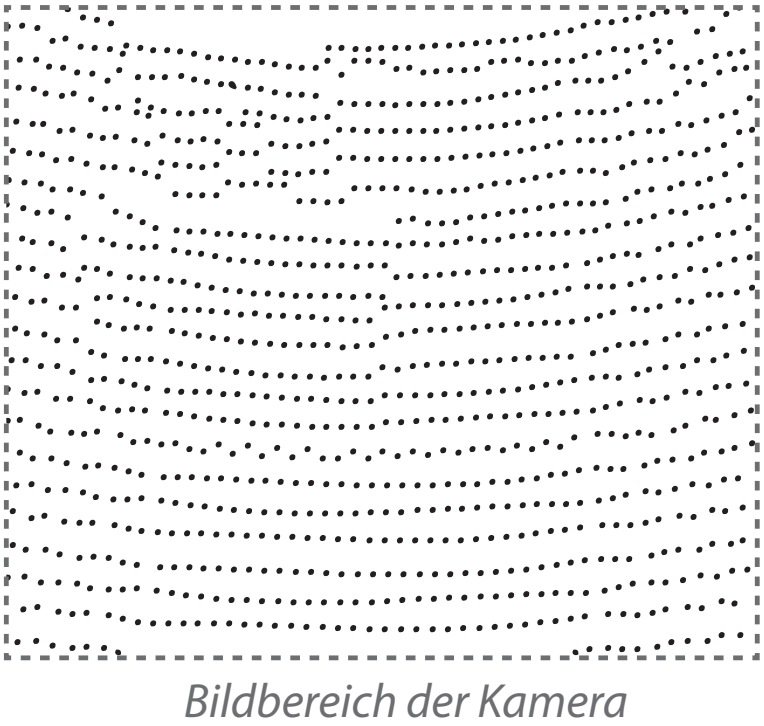

Abbildung 3.6: Projezierte Messwerte auf Bildebene

ist bei einer Aufnahme eines Bildes über die Kamera noch keine Korrektur eingerechnet. In diesem Fall treten die bereits besprochenen Effekte auf. Da es sich bei den Messpunkten nur um dreidimensionale Koordinaten im Raum handelt, können sie nicht einfach durch die Linse abgebildet werden. Man muss die Verzerrungseffekte durch die Linse nachträglich einrechnen, da in diesem Fall auf dem noch verzerrten Bild gearbeitet wird.

Die Projektion der Messpunkte ist notwendig, um zu ermitteln, mit welcher Position beziehungsweise welchem Pixel der Messwert korrespondiert. Diese Korrespondenz ist von fundamentaler Bedeutung für die an späterer Stelle beschriebenen Interpolationsverfahren. Diese Verfahren benötigen für die Interpolation einige Punkte auf der Bildebene mit Tiefeninformationen.

Liegen Messpunkte ausserhalb des Sichtbereichs der Kamera und werden projeziert, dann liegen ihre Projektion zwar auf der Bildebene, aber ausserhalb des Bildbereichs. Diese Punkte werden nur in speziellen Fällen oder gar nicht gebraucht.

Rückabbildung in den dreidimensionalen Raum

Die Rückabbildung kann erst durchgeführt werden, wenn für jeden Pixel des Bildes ein Tiefenwert über eine Interpolation bestimmt worden ist. Die Ausführung der Rückabbildung bedient sich dabei der intrinsischen Parameter. Zunächst wird für jeden Bildpunkt ein normierter Richtungsvektor berechnet, der anschließend mit dem interpolierten Tiefenwert multipliziert wird. Im Grunde genommen ist es die Invertierung der Projektion (siehe 3.2). Die Projektion p eines Punktes P wurde bereits mit Formel 3.5 definiert.

Zwischen der Rückabbildung und der Projektion gibt es allerdings einen entscheidenden Un-

terschied. Im Falle der Projektion ist die Tiefenkoordinate z_p von P bekannt. Da zu diesem Zeitpunkt die interpolierten Tiefenwerte bekannt sind, stellt dies aber kein Problem dar.

Sei der Punkt $q = (x_q, y_q, 1)^T$ ein Punkt auf der Bildebene und Q seine Abbildung $\in \mathbb{R}^3$ im Kamera-Koordinatensystem und Q' der Richtungsvektor für q. Für den Punkt q sollten eventuelle Korrekturen bereits in Betracht gezogen sein. Dann kann, durch Umformung der Projektion, die Rückabbildung auf folgende Art und Weise beschrieben werden [BK08e]:

$$Q' = \begin{pmatrix} \frac{1}{f_x} & 0 & -\frac{c_x}{f_x} \\ 0 & \frac{1}{f_y} & -\frac{c_y}{f_y} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_q \\ y_q \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{x_q}{f_x} - \frac{c_x}{f_x} \\ \frac{y_q}{f_y} - \frac{c_y}{f_y} \\ 1 \end{pmatrix} \tag{3.10}$$

Damit man aus Q' den tatsächlichen Punkt Q erhält muss nun noch mit dem interpolierten Tiefenwert i multipliziert werden:

$$Q = Q' \cdot i = \begin{pmatrix} (\frac{x_q}{f_x} - \frac{c_x}{f_x}) \cdot i \\ (\frac{y_q}{f_y} - \frac{c_y}{f_y}) \cdot i \\ i \end{pmatrix} \tag{3.11}$$

Damit ist die Berechnung der Rückprojektion für den Punkt Q abgeschlossen.

4 Grundlagen zur Interpolation von Tiefenwerten

Im vorherigen Kapitel wird besprochen, wie sich Kamera und 3D-Laserscanner-Daten über die Projektion der Messpunkte vereinigen lassen. Jeder Pixel im Kamerabild soll durch Interpolation einen Tiefenwert erhalten und lässt sich damit auf einen dreidimensionalen Punkt im Kamerakoordinatensystem abbilden. Bevor jedoch die einzelnen Verfahren betrachtet werden können, sollten zunächst die Grundbausteine für die Interpolation besprochen werden. Die in dieser Arbeit verwendeten Tiefeninterpolationsverfahren stützen sich hauptsächlich auf bestimmte Nachbarschaftseigenschaften und diverse Gewichtungen. Mit Nachbarschaftseigenschaften sind zum Beispiel die *nächsten* oder die *natürlichen Nachbarn* gemeint. Die Gewichtungen werden für die Berechnung der interpolierten Tiefenwerte benötigt. In diesem Kapitel werden einige wichtige Grundlage für diese Gewichtungsverfahren besprochen.

4.1 Nachbarschaften

Nachbarn können auf unterschiedliche Arten gefunden und auf bestimmt werden. Einige Verfahren setzen auf die unmittelbare Nachbarschaft, andere berechnen diese unter Beachtung verschiedener Einflüsse. Für alle Nachbarschaften gilt jedoch, dass sie als Eingabepunkte die bereits projezierten Messwerte, ob im Bildbereich der Kamera oder nicht, erhalten.

Delauny Triangulierung - nächster Nachbar

Die *Delauny Triangulierung* stellt das Fundament einer Methode zur Ermittlung nächster Nachbarn dar. Bei der Delauny Triangulierung wird ein Netz aus Dreiecken zwischen den Eingabepunkten $\in \mathbb{R}^2$ gebildet. Die Algorithmen, die für die Berechnung der Triangulierung existieren, versuchen im Allgemeinen zu vermeiden, dass flache, langgezogene Dreiecke entstehen. Das bedeutet, dass versucht wird jeweils maximale Innenwinkel für ein Dreieck zu erhalten [BK08f]. Diese Eigenschaft lässt sich über die Umkreisbedingung formulieren:

Für jedes Dreieck in der Triangulierung gibt es einen Umkreis, der dessen drei Punkte schneidet. Für diesen Umkreis gilt, dass keine Knoten beziehungsweise Punkte in dessen Innerem liegen dürfen. Abbildung 4.1 zeigt, wie eine gültige Zerlegung aussehen kann. Die Umkreisbedingung ist genau dann erfüllt, wenn die Innenwinkel der Dreiecke maximal sind.

Die reine Triangulierung ist für das Finden eines nächsten Nachbarn noch nicht ausreichend. Deswegen muss zusätzlich die *Voronoi-Zerlegung* berechnet werden. Hierfür werden die Delauny-Kanten durch deren Orthogonalen in ihrem Mittelpunkt ausgetauscht. Eine Orthogonale wird dabei immer durch die ersten beiden Schnittpunkte mit Orthogonalen anderer Delauny-Kanten

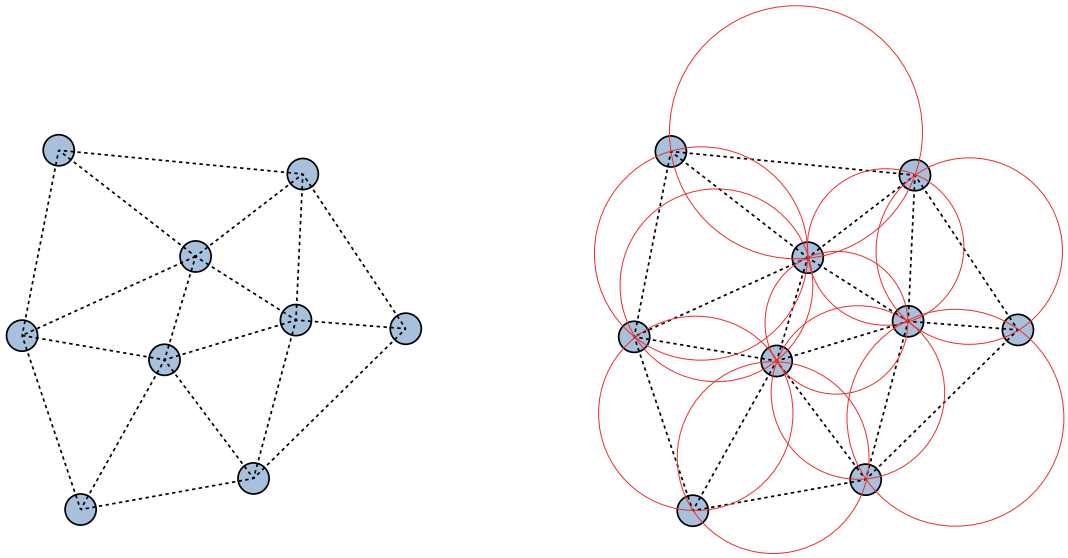

Abbildung 4.1: gültige Delauny-Zerlegung mit erfüllter Umkreisbedingung

begrenzt. Durch diesen Vorgang ergeben sich *Voronoi-Zellen*. Für zwei benachbarte Zellen gilt, dass die anliegende Zellwand genau gleich weit von den beiden zugehörigen Delauny-Knoten entfernt ist (siehe Abb. 4.2).

Für einen beliebigen Punkt $P \in \mathbb{R}^2$ kann nun innerhalb der Menge von Punkten der Triangulierung ein nächster Nachbar ausgemacht werden. Hierfür wird bestimmt in welche der generierten Voronoi-Zellen der Punkt P fällt. Der Knoten dieser Zelle ist dann der gesuchte nächste Nachbar.

Das Ermitteln mehrerer nächster Nachbarn innerhalb einer Delauny- beziehungsweise Voronoi-Zerlegung gestaltet sich leider nicht so einfach. Wenn ein nächster Nachbar ermittelt worden ist, müsste dieser Knoten aus der Zerlegung entfernt und diese neu berechnet werden. Anschließend ließe sich ein weiterer nächster Nachbar auf die gleiche Art und Weise ermitteln lassen, wie der Erste.

Konkret bedeutet das, dass das Berechnen mehrerer nächster Nachbarn über die Zerlegungen prinzipiell möglich ist, sich aber als äußerst unpraktisch darstellt.

Man kann diesbezüglich einen Kompromiss schließen. Da in der Delauny-Zerlegung alle Knoten mit einander verbunden sind, lassen sich die nächsten Nachbarn für einen Knoten sehr einfach ermitteln. Für diesen müssen nur die mit ihm verbundenen Knoten in Betracht gezogen werden. Bei entsprechender Implementierung kann sogar direkt der Delauny-Graph für die Ermittlung genutzt werden [BK08g]. Die Methodik bildet zwar nur eine Approximation, lässt sich allerdings schnell und einfach verwenden, ohne dass eine Zerlegung neu berechnet werden muss (Abb. 4.3 links).

Es gibt wie bereits am Anfang erwähnt unterschiedliche Algorithmen, die die Delauny-Zerlegung und damit die Voronoi-Zerlegung berechnen können. Sie unterscheiden sich sowohl von der internen Vorgehensweise, als auch von der Komplexität ihrer Struktur. Daraus ergibt sich, dass sie unterschiedliche Laufzeiten entwickeln. Die schnellsten Algorithmen für diese Berechnung benötigen für eine Eingabe der Größe n eine Laufzeit von $\mathcal{O}(n \log \log n)$. Der in dieser Arbeit verwendete Algorithmus von der Bibliothek OpenCV benötigt $\mathcal{O}(n^2)$ [BK08f].

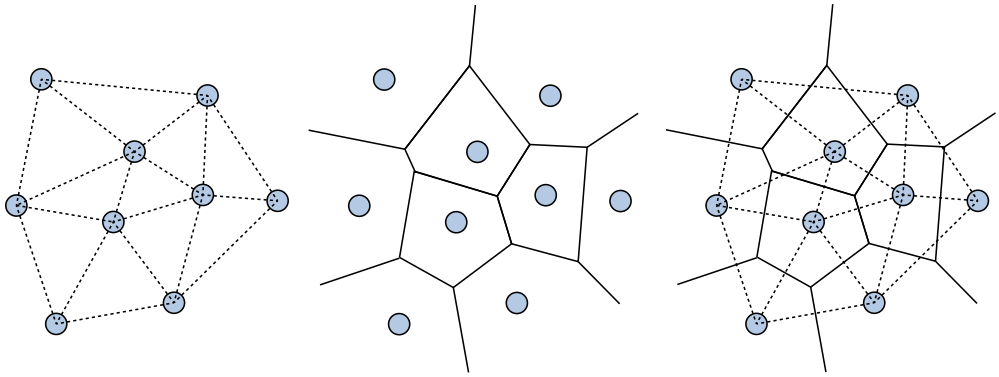

Abbildung 4.2: Delauny-Triangulierung und Voronoi-Zerlegung

Natürliche Nachbarn

Die *natürliche Nachbarschaft* wird maßgeblich über die Voronoi-Zerlegung bestimmt. Dabei existiert eine für eine gewisse Anzahl an Eingabepunkten berechnete Zerlegung. Sollen für einen beliebigen Punkt $P \in \mathbb{R}^2$ dessen natürliche Nachbarn ermittelt werden, muss P zunächst in die Voronoi-Zerlegung eingefügt werden. Anschließend wird für P eine Zelle in die Voronoi-Zerlegung eingefügt, was eine lokale Veränderung hervorruft (siehe Abb. 4.3 rechts).

Die natürlichen Nachbarn von P sind nun jene Knoten in der Zerlegung, deren Zellen die neu eingefügte Zelle von P berühren.

P ist nicht der einzige Punkt, für den die natürlichen Nachbarn aus der Ursprungsmenge der Zerlegung berechnet werden sollen. Daher wird die Zerlegung nicht dauerhaft verändert und P wird auch nicht in die Menge der Eingabepunkte der Zerlegung aufgenommen.

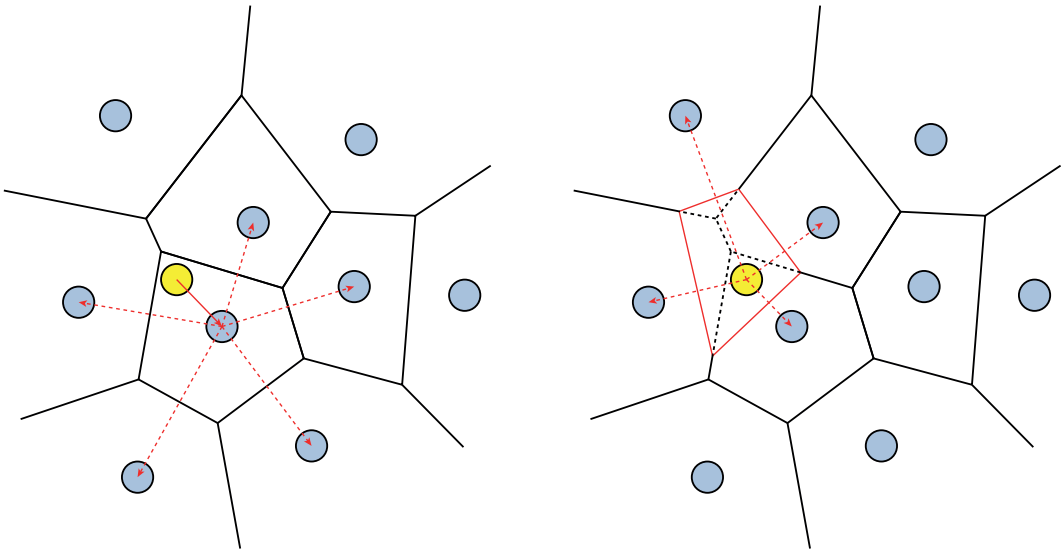

Abbildung 4.3: Einfügen eines Punktes. **links**: nächste Nachbarn des nächsten Knoten der Zerlegung. **rechts**: Natürliche Nachbarn des eingefügten Punktes.

Für die Berechnung der natürlichen Nachbarn wird bei dieser Arbeit ein Algorithmus der Bibliothek CGAL angewandt. Die Implementierung des Algorithmus stützt sich dabei auf verschiedene wissenschaftliche Veröffentlichungen [Flö11] zu natürlichen Nachbarn und den dazu gehörigen Interpolationen. Wie lang der Algorithmus jedoch zur Berechnung der Ausgangszerlegung benötigt ist leider nicht bekannt.

k' nächste Nachbarn

Neben nächster Nachbarschaft und natürlicher Nachbarschaft, gibt es noch die Möglichkeit genau k' *nächste Nachbarn* zu bestimmen. Der wesentliche Unterschied dieses Verfahrens zu den bereits besprochenen ist, dass immer genau k' viele Nachbarn ausgemacht werden. Die Anzahl der Nachbarn k' kann dabei größer als die Anzahl der natürlichen oder nächsten Nachbarn sein, aber auch kleiner. Im Grunde hängt dies von der Wahl von k' ab.

Dieses Verhalten lässt sich an einem kleinen Beispiel verdeutlichen. Bildet man eine Menge von Eingabepunkten auf ein Bild ab, ist in dieser Fläche die Anzahl der natürlichen oder nächsten Nachbarn zunächst durch die Dichte der Punkte bestimmt. Es lässt sich jedoch festhalten, dass ein Punkt P am Rand in der Regel weniger Nachbarn dieser Form besitzt als ein Punkt P', der in der Mitte des Bildes liegt.

Bei der Suche nach k' nächsten Nachbarn würde man aber für beide Punkte definitiv die gleiche Menge an Nachbarn erhalten.

Um für einen beliebigen Punkt k' nächste Nachbarn zu ermitteln, dient im Regelfall die Distanz dazu, festzustellen, welcher der Eingabepunkte dazu gehört. Daher lässt sich das Verfahren ein wenig verändern. Statt nach genau k' vielen nächsten Nachbarn zu suchen, kann auch nach genau k' vielen Nachbarn in einem bestimmten Radius gesucht werden. Wird k' dabei groß genug gewählt, werden alle Nachbarn in diesem Radius gefunden. Gibt es mehr Nachbarn in diesem Radius, werden die k' nächsten ausgewählt. Sind weniger als k' viele Nachbarn im Radius enthalten, dann werden nur diese ausgewählt.

Zieht man an dieser Stelle wieder einen Vergleich zu den nächsten oder natürlichen Nachbarn, ergibt sich auch hier ein Unterschied. Es ist nicht gesagt, dass die Menge von Eingabepunkten im Raum gleich verteilt ist. Das bedeutet, dass ein nächster Nachbar unter Umständen eine etwas größere Distanz zu einem Punkt, für den gesucht werden soll, haben kann. So kann es vorkommen, dass dieser nächste Nachbar außerhalb des gewählten Radius liegt und damit nicht mehr zu der Menge der umliegenden Eingabepunkte gezählt wird.

Das Ergebnis der Suche nach genau k' nächsten Nachbarn stellt nur eine Approximation für tatsächliche Nachbarschaften dar. Das Gleiche gilt für k' nächste Nachbarn innerhalb eines bestimmten Radius. Je nach Wahl der Parameter kann das Ergebnis der Suche deutlich von der tatsächlichen Nachbarschaft abweichen.

Die Suche nach den k' nächsten Nachbarn kann und wird häufig über sogenannte *k-d-Bäume* realisiert. Ein solcher Baum ist ein k-dimensionaler Suchbaum, dessen innere Knoten die Position von Hyperebenen speichern, die zur Unterteilung des Raumes $\mathbb{R}^k$ dienen. Die Menge von Punkten $\in \mathbb{R}^k$, aus der die Nachbarn für beliebige andere Punkte ermittelt werden sollen, wird auf die Blätter des Baumes verteilt. Die Verzweigungen des Baumes formulieren dabei, in welchem Unterraum sich der jeweilige Punkt aufhält.

Sollen für einen beliebigen Punkt die k' nächsten Nachbarn gefunden werden, so muss der

Baum durchlaufen und die entsprechenden Unterräume ermittelt werden. k-d-Bäume können für beliebige Dimensionen k verwendet werden.

An dieser Stelle wird bereits mit Punkten, projeziert auf eine Bildebene, gearbeitet. Diese Ebene und die Punkte selbst sind zweidimensional. Deswegen reicht es 2-d-Bäume zu betrachten. Abb. 4.4 zeigt eine beispielhafte Zerlegung einer solchen zwei dimensionalen Ebene, in der Punkte enthalten sind. Die eingezeichneten, farbigen Linien stellen dabei sogenannte *Splitgeraden* dar. Da diese Geraden immer parallel zu einer der Achsen des Koordinatensystem der Ebene sind, reicht es aus nur die x- beziehungsweise y-Koordinate zu speichern [Kle05]. Die Unterteilung in Aufenthaltsräume wird solange vorgenommen, bis jeder Punkt in gerade einem solchen enthalten ist.

In der Darstellung zeichnet sich zu dem ab, dass diese Splitgeraden in einer bestimmten Reihenfolge festgelegt worden sind. In diesem Fall wurde zunächst die rote, dann die blauen und im letzten Schritt die beiden grünen Geraden eingefügt. Für die Punkte $P_i | i = 1,...,6$ ist am Ende der Bearbeitung eine eindeutige Zuordnung möglich.

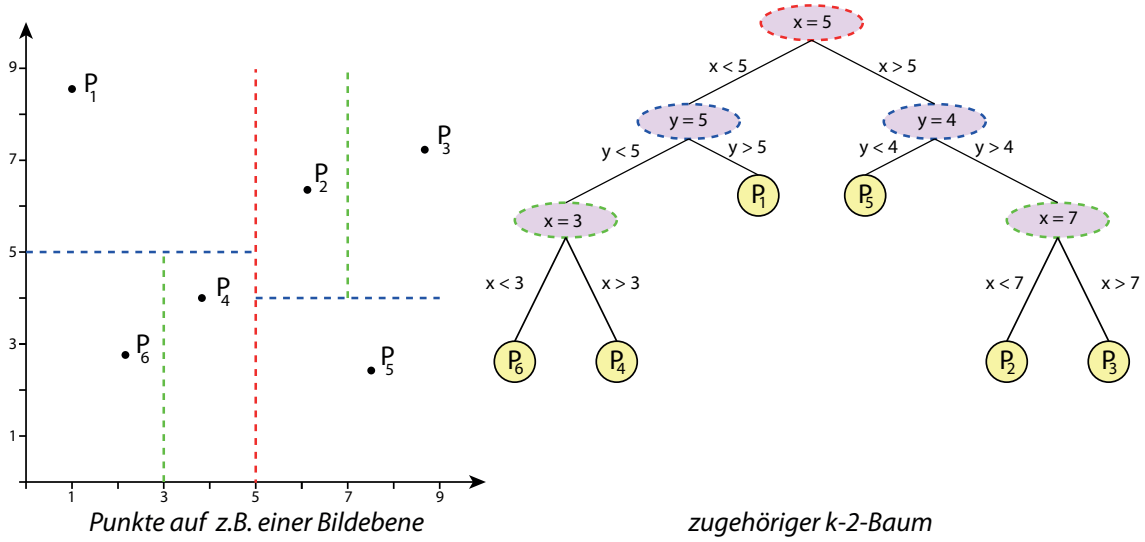

Abbildung 4.4: Einteilung einer Menge von Punkten in einer Ebene über einen k-d-Baum.

Das Resultat der Einteilung ist ein 2-d-Baum auf der rechten Seite der Abbildung 4.4. Wie bereits erwähnt, werden die Koordinaten der Geraden in den inneren Knoten festgehalten. Eine linksseitige Verzweigung beschreibt, dass die x- oder y-Koordinate eines Punktes kleiner war als die für die Gerade gespeicherte x- oder y-Koordinate. Entsprechend war der Wert einer Koordinate bei rechtsseitiger Verzweigung größer. Sind die Bedingungen für die Geraden im Baum kodiert, dann werden die Eingabepunkte, entsprechend ihrem Aufenthaltsraum, auf die Blätter verteilt.

In der Regel benötigt der Aufbau eines 2-d-Baumes bei einer Eingabe der Größe n eine Laufzeit von $\mathcal{O}(n \log n)$.

Nächste Nachbarn in einer geordneten Datenstruktur

Bei den bereits vorgestellten Nachbarschaften, werden die Eingabedaten zuerst in eine bestimmte Datenstruktur überführt, zum Beispiel die Delauny-Zerlegung. Anschließend wird die

Suche auf diesen Datenstrukturen durchgeführt.

Da die Eingabepunkte einzeln in diese Datenstrukturen eingegeben und innerhalb von diesen sinnvoll sortiert werden, spielt die Reihenfolge dieser Punkte keine Rolle.

Erhält man die Eingabepunkte bereits in einer geordneten Struktur, beispielsweise in einer Matrix oder einem Array, lässt sich diese bereits benutzen, um Nachbarschaften zu approximieren.

Viele 3D-Laserscanner besitzen einen rechteckigen oder gar quadratischen Messbereich, in dem die Messpunkte normalerweise gleichmäßig verteilt sind (Abb. 4.5 rechts).

Sind alle Messwerte ermittelt, dann werden die Messdaten in einem Vektor oder einer Punktwolke zurückgegeben.

Die Messpunkte sind normalerweise gleichmäßig verteilt, weswegen sich eine matrixähnliche Struktur abzeichnet. Sind die Breite und die Höhe der Einteilung des Messbereichs bekannt, lassen sich verhältnismäßig einfach die Punktpositionen im Vektor von Messdaten bestimmen.

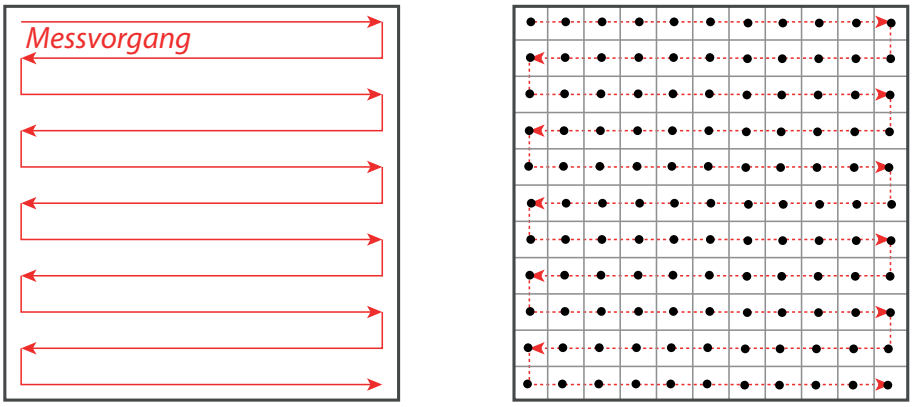

Abbildung 4.5: Messzyklus im Messbereich und Messpunktverteilung

Da an dieser Stelle, wie bereits erwähnt, mit Punktpositionen auf einer Bildebene gearbeitet wird, muss für das Gelingen dieser Suche nach Nachbarn die Struktur der Daten erhalten werden. Das bedeutet, dass die Projektionen der Messpunkte genau in der gleichen Reihenfolge oder Art und Weise gespeichert werden müssen.

Wie bereits in Kapitel 3 im Abschnitt 3.3 besprochen wurde, können Messpunkte jedoch außerhalb des Bildbereichs der Kamera liegen. Diese Punkte müssen jedoch für die Erhaltung der Ordnung in der Datenstruktur mit einbezogen werden.

Für das Ergebnis der Suche von Nachbarn, für einen beliebigen Punkt des Bildbereichs, innerhalb der Datenstruktur ist eine Nachbearbeitung nötig. Dabei muss jeder gefundene Nachbar darauf überprüft werden, ob er innerhalb oder außerhalb des Bildbereichs liegt.

Die Suche nach nächsten Nachbarn innerhalb einer geordneten Datenstruktur stellt eine Approximation für die Suche nach tatsächlichen nächsten Nachbarn dar. Die Annahme die getroffen wird ist, dass wenn eine Nachbarschaft in der Datenstruktur vorliegt, auch eine Nachbarschaft unter den Projektionen existiert. Dabei wird außer Acht gelassen, welche Positionen die Projektionen der Messpunkte tatsächlich haben. Dadurch kann eine Abweichung entstehen.

Folgender Ablauf liegt der Suche nach nächsten Nachbarn in der Datenstruktur zugrunde:

1. Finden des nächsten Nachbarn.
2. Auslesen der Nachbarn in der Datenstruktur.
3. Überprüfung der gefundenen Punkte auf die Lage im Bezug auf den Bildbereich.

Abbildung 4.6 zeigt, welche Punkte sich in der Nachbarschaft anderer Messpunkte befinden. Es handelt sich dabei um die vom 3D-Laserscanner gelieferte Datenstruktur.

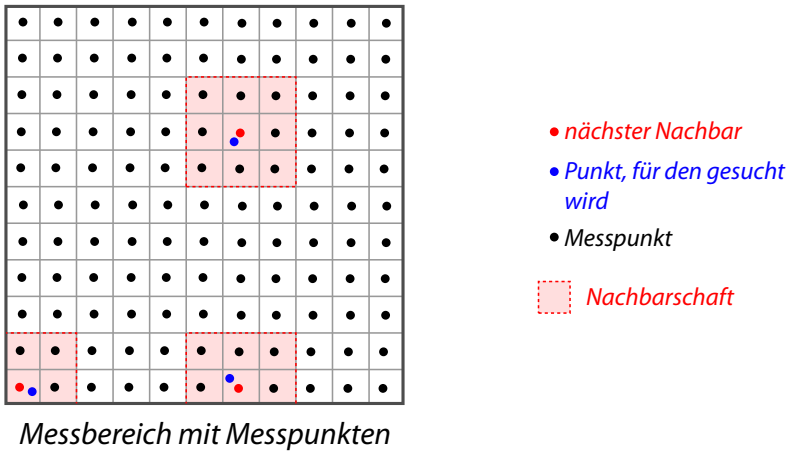

Abbildung 4.6: Nachbarschaft unter Messpunkten

Ob diese Suche von Nachbarn angewendet werden kann, hängt grundsätzlich davon ab, wie die Messdaten gespeichert sind. Liefert ein 3D-Laserscanner die Daten in wechselnder Reihenfolge, lässt sie sich gar nicht anwenden.

Der wesentliche Vorteil dieser Approximation ist, dass keine weitere Laufzeit benötigt wird, um die Daten in eine andere Struktur, wie einen k-d-Baum oder eine Delauny-Zerlegung zu überführen. Lediglich das Finden des ersten nächsten Nachbarn zu einem Punkt nimmt etwas Zeit in Anspruch. Ist dieser Nachbar erst einmal gefunden, lassen sich die restlichen äußerst schnell ermitteln. Ausschlaggebend dafür ist die Möglichkeit der Auswahl über ein festes Schema auf einer linearen Datenstruktur.

Der einzige Nachteil dieser Approximation ist, dass die Genauigkeit aufgrund der getroffenen Annahmen niedriger ausfällt als bei anderen Suchen.

4.2 Schreibweisen und Varianzen

Die Interpolationsverfahren arbeiten mit vielschichtigen Daten. Beispielsweise besteht ein Pixel aus einem Bild nicht aus nur einem, sondern aus vielen Werten. Außer der Position, ist in einem Pixel zusätzlich die Farbe gespeichert, die auch wieder zerlegt werden kann.

Bevor im nächsten Kapitel näher auf die Interpolationsverfahren eingegangen werden kann, müssen zunächst eine einheitliche Schreibweise sowie einige Konstanten definiert werden. Die Schreibweise bezieht sich dabei auf Pixel, abgebildete Messwerte und Farben. Zu den Konstanten, die immer wieder in den Interpolationsverfahren auftauchen, zählen die Farb- und Vertei-

lungsvarianz.

Schreibweisen

- Ein Pixel $P_j = (X_j, Y_j, C_J)$ besteht aus zwei Koordinaten X_j und Y_j sowie aus einem Farbwert C_J.
- Ein Farbwert C_j gliedert sich in drei Werte C_j^1, C_j^2 und C_j^3 auf. Diese Werte stehen zum Beispiel für rot, grün und blau, wenn der *RGB-Farbraum* genutzt wird.
- Ein auf die Bildebene, in den Bildbereich abgebildeter Messwert $R_i = (X_i, Y_i, r_i, C_i)$ ist im Grunde ein um einen Tiefenwert r_i erweiterter Pixel.
- Ein interpolierter Messwert $R_j^* = (X_j, Y_j, r_j^*, C_j)$ auf der Bildebene im Bildbereich ist ein um einen interpolierten Tiefenwert erweiterter Pixel.

Wie sich im Kapitel 3 in Abschnitt 3.3 gezeigt hat, genügt es an dieser Stelle für einen abgebildeten Messwert den Tiefenwert r_i zu speichern.

Tiefenwerte werden nur für Pixelpositionen berechnet, die noch nicht von einem abgebildeten Messwert belegt sind [ATL07].

Verteilungs- und Farbvarianz

Für die Berechnung einiger Gewichtungen ist es nötig verschiedene Varianzen einzubeziehen. Die Varianzen geben jeweils die Streuung der Farb- beziehungsweise Positionswerte der Messpunkte an.

Verteilungsvarianz

Die Verteilungsvarianz σ_p bezieht sich auf die Streuung der Messwerte im Bildbereich. Sei M die Menge aller Positionen $m_i = (X_i, Y_i)$ im Bildbereich, die durch einen projezierten Messpunkt belegt werden. Dann wird zunächst der Mittelwert $m_{mittel} = (X_{mittel}, X_{mittel})$ der Positionen berechnet:

$$m_{mittel} = \frac{\sum_{i=0}^{|M|} m_i}{|M|} \tag{4.1}$$

Ist P_{pos} die Menge aller Positionen von Pixeln $p_i = (X_i, Y_i)$ im Bildbereich, dann gilt für die Verteilungsvarianz σ_p:

$$\sigma_p = \frac{\sum_{i=0}^{|P_{pos}|} (m_{mittel} - p_i)^2}{|P_{pos}| - 1} \tag{4.2}$$

Wobei es sich bei $(m_{mittel} - p_i)^2$ um das Skalarprodukt des aus der Differenz resultierenden Punktes mit sich selbst handelt.

Farbvarianz

Die Berechnung der Farbvarianz folgt einem sehr ähnlichen Muster und gibt an, wie stark die Streuung der Farbwerte der abgebildeten Messpunkte ist. Sei F die Menge aller zu Messpunkten gehörenden Farbwerte $F_i = (C_1, C_2, C_3)$, so ist der Mittelwert C_{mittel} mit $C_{mittel} = (C^1_{mittel}, C^2_{mittel}, C^3_{mittel})$:

$$C_{mittel} = \frac{\sum_{i=0}^{|F|} F_i}{|F|} \tag{4.3}$$

Ist F_p die Menge aller Farben der im Bildbereich liegenden Pixel und $C_i = (C_1, C_2, C_3)$ eine Farbe, dann gilt für die Farbvarianz σ_c:

$$\sigma_c = \frac{\sum_{i=0}^{|F_p|} (C_{mittel} - C_i)^2}{|F_p| - 1} \tag{4.4}$$

Auch bei $(C_{mittel} - C_i)^2$ handelt es sich um das Skalarprodukt des Differenzvektors mit sich selbst.

5 Interpolationsverfahren

Nach dem alle Grundlagen für die Interpolationsverfahren vorgestellt und erörtert sind, kann auf diese im Detail eingegangen werden. Den Interpolationsverfahren liegen verschiedene Gewichtungen zur Berechnung der Werte zugrunde.

Bei den Gewichtungen muss dabei zwischen zwei Kategorien unterschieden werden. Es gibt Gewichtungen, die direkt in die Berechnung der interpolierten Tiefenwerte mit einfließen und es gibt solche, die sich auf die Berechnung der Tiefenwerte indirekt auswirken.

Die Gewichtungen, die direkt in die Berechnung mit einfließen, lassen sich wiederum in verschiedene Kategorien unterteilen. Dabei gibt es verschiedene Ansätze, die Flächeninhalte und Abstände einbeziehen. Diese Ansätze können wiederum über eine Gewichtung durch die Farbe erweitert werden.

Ziel der Interpolationsverfahren ist es für die übrige Menge von Bildpunkten, auf die kein Messwert des 3D-Laserscanners abgebildet worden ist, Tiefenwerte zu berechnen. Diese Tiefenwerte werden anschließend, über die im Kapitel 3 unter Abschnitt 3.3 besprochene Rückabbildung, wieder auf 3D-Punkte im Kamera beziehungsweise Laserkoordinatensystem abgebildet.

5.1 Nearest Range Reading - NR

Das erste und im weitesten Sinne einfachste Interpolationsverfahren ist das *Nearest Range Reading* oder *der nächste Messwert* zu einem Bildpunkt [ATL07].

Die Interpolation in diesem Verfahren stützt sich auf eine Gewichtung, die sich nur indirekt auf die Berechnung des Tiefenwertes auswirkt. Die Gewichtung stellt vielmehr die Wahrscheinlichkeit dar, dass ein abgebildeter Messpunkt die korrekte Auswahl für einen beliebigen Bildpunkt darstellt.

Ist ein Messwert R_i für einen beliebigen Bildpunkt P_j gewählt, dann wird dieser durch einen interpolierten Messwert R_j^* ersetzt. R_j^* übernimmt dabei die Tiefeninformation von R_i.

Die Wahrscheinlichkeit ist dabei proportional zu einer Exponentialfunktion, die sowohl den quadratischen Abstand, als auch die Verteilungsvarianz mit einbezieht. Die Verteilungsvarianz wird bereits in Kapitel 4 unter Abschnitt 4.2 behandelt und definiert. Sie gibt eine Aussage darüber, wie sich die Streuung der Messpunkte verhält.

Die Wahrscheinlichkeit $p(P_j, R_i)$, dass für einen Bildpunkt beziehungsweise Pixel P_j ein Messwert R_i als am Besten geeigneter ausgewählt wird, ist wie folgt definiert:

$$p(P_j, R_i) \propto e^{-\frac{(X_j - X_i)^2 + (Y_j - Y_i)^2}{\sigma_p^2}} \tag{5.1}$$

Wie sich zeigt lässt dieses Verfahren die Farbe von Bildpunkt P_j und vom abgebildeten Messpunkt R_i außen vor. Würde die Verteilung von Messpunkten auf der Bildebene immer dieselbe sein, dann würde das Verfahren trotz wechselnder Farbbilder die Messpunkte auf dieselben Bildpunkte verteilen. Aufgrund von beispielsweise Messfehlern ist davon nicht auszugehen.

Die Bildebene beziehungsweise der Bildbereich der Kamera ist allerdings wichtig als Bezug. Ohne diesen Bereich gäbe es keine Verbindung zwischen Punkten, für die interpoliert werden soll, und den gemessenen Werten des 3D-Laserscanners.

Anstatt alle Messwerte auf die Wahrscheinlichkeit hin zu überprüfen, ob sie als bester Messwert in Frage kommen, kann an dieser Stelle die Suche nach dem nächsten Nachbarn getätigt werden (Kapitel 4 Abschnitt 4.1). Für den nächsten Nachbarn sollte der Abstand am geringsten und somit die Wahrscheinlichkeit am höchsten sein.

5.2 Nearest Range Reading Consindering Color - NRC

Das zweite im Zuge dieser Arbeit implementierte Interpolationsverfahren ist das *Nearest Range Reading Considering Color* oder *der nächste Messwert unter Beachtung der Farbe* [ATL07].

Dieses Verfahren stellt eine Erweiterung des NR Interpolationsverfahren dar. Dabei wird für einen beliebigen Punkt P_j ein Messwert R_i gesucht, der möglichst in der Nähe liegt und dabei eine ähnliche Farbe besitzt. Anschließend wird P_j durch einen interpolierten Messwert R_j^* ersetzt, wobei dieser den Tiefenwert von R_i übernimmt.

Die Exponentialfunktion, von der die Wahrscheinlichkeit abhängt, dass ein Messpunkt der beste zu wählende ist, muss so erweitert werden, dass der Farbunterschied mit in Betracht gezogen wird.

Dafür lässt sich zunächst eine Funktion definieren, zu der die Wahrscheinlichkeit $p_c(P_j, R_i)$ proportional ist. p_c ist die Wahrscheinlichkeit für den besten zu wählenden Messpunktes in Abhängigkeit des Farbunterschieds:

$$p_c(P_j, R_i) \propto e^{-\frac{||C_j - C_i||^2}{\sigma_c^2}} \tag{5.2}$$

Die Exponentialfunktion, die im NR Verfahren Anwendung findet, wird dann wie folgt erweitert:

$$p(P_j, R_i) \propto e^{-\frac{(X_j - X_i)^2 + (Y_j - Y_i)^2}{\sigma_p^2} - \frac{||C_j - C_i||^2}{\sigma_c^2}} \tag{5.3}$$

Damit wird bei der Berechnung der Wahrscheinlichkeit für einen Messpunkt R_i für einen Bildpunkt P_j die Farbe mit einbezogen.

Im Unterschied zum NR Verfahren dienen Bildebene und darin liegender Bildbereich nicht mehr nur als gemeinsamer Bezug, sondern werden direkt einbezogen.

Für die Suche nach dem optimalen Messpunkt R_i für einen beliebigen Bildpunkt P_j müssen der Farbunterschied und die Distanz gleichermaßen gering bleiben. Wird die Distanz zu hoch sinkt die Wahrscheinlichkeit. Es ist daher weniger sinnvoll alle Messpunkte zu überprüfen.

Beim NR Interpolationsverfahren kommt die Suche nach dem nächsten Nachbarn unter den Messpunkten zum Einsatz. Nur diesen zu Suchen wäre sinnlos. Es ist zwar die höchste Wahrscheinlichkeit im Bezug auf die Distanz garantiert, jedoch nicht, dass der Farbabstand gering ist.

Letztendlich ist die beste Methode für die Ermittlung von R_i für einen Punkt P_j, nach den benachbarten Messpunkten im unmittelbaren Umkreis zu suchen.

In Kapitel 4 werden einige Verfahren beziehungsweise Datenstrukturen erwähnt, die dieser Aufgabe gerecht werden können:

- Die Erweiterung des Verfahren zur Bestimmung des nächsten Nachbarn 4.1.
- Das Verfahren zu Bestimmung der k nächsten Nachbarn 4.1.
- Die Erweiterung dieses Verfahrens zur Bestimmung der k nächsten Nachbarn in einem bestimmten Umkreis.
- Die Bestimmung nächster Nachbarn in einer geordneten Datenstruktur 4.1.

Damit sollte ein Messpunkt ermittelt werden können, so dass die Wahrscheinlichkeit, dass er optimal ist, am höchsten ausfällt.

5.3 Multi Linear Interpolation - MLI

Mit der *Multi Linear Interpolation* wird das erste Verfahren vorgestellt, das die Gewichtungsfaktoren direkt für die Berechnung der interpolierten Tiefenwerte mit einbezieht [ATL07].

Dieses Verfahren hat mit dem NR Interpolationsverfahren eine Gemeinsamkeit. Für die Interpolation von Tiefenwerten wird die Farbe nicht in Betracht gezogen. Das bedeutet, wenn die Abbildung der Messpunkte auf die gleiche Stelle auf der Bildebene im Bildbereich geschehen würde, dann würden für jeden Bildpunkt bei der Berechnung die gleichen Messpunkte einbezogen werden.

Für die Gewichtung der Tiefenwerte bei der Berechnung gibt es zwei unterschiedliche Hauptgruppen. Die Gewichtung über die Distanz und die Gewichtung über die Flächeninhalte, die in einem Voronoi-Diagramm vorliegen.

Die Interpolation eines Tiefenwertes für einen beliebigen Bildpunkt P_j erfolgt über die Tiefenwerte einer Anzahl umliegender Messpunkte R_{P_j}. In Abhängigkeit des für das Finden der Messpunkte verwendeten Suchverfahrens, kann entweder die Distanz- oder die Flächengewichtung angewandt werden. Für das Finden der umliegenden Messpunkte können wieder die Verfahren aus Kapitel 4 verwendet werden.

In den folgenden beiden Abschnitten wird erläutert, wie die Tiefenwerte der einzelnen umliegenden Messpunkten mit der entsprechenden Gewichtung verrechnet werden und den interpolierten Tiefenwert für P_j ergeben.

Gewichtung über Distanzen

Die erste Variante zur Berechnung der Tiefenwerte bedient sich der Gewichtung über die Distanz. Dabei hat ein Tiefenwert einen größeren Anteil am interpolierten Wert, wenn der dazugehörige Messpunkt R_i näher am Bildpunkt P_j liegt, für den der Wert berechnet werden soll.

Zunächst sei die Distanz zwischen einem beliebigen Bildpunkt P_j und einem für P_j gewählten Messwert R_i der Wert d_i mit:

$$d_i(P_j, R_i) = \sqrt{(X_i - X_j)^2 + (Y_i - Y_j)^2} \tag{5.4}$$

Jetzt muss ein Normalisierungsfaktor N_d definiert werden, so dass ein interpolierter Tiefenwert nie größer werden kann, als der maximale Tiefenwert der umliegenden Messpunkte. Sei R_{P_j} die Menge aller umliegenden Messpunkte R_i für einen beliebigen Bildpunkt P_j. Dann gilt für N_d:

$$N_d(P_j, R_{P_j}) = \sum_{i=0}^{|R_{P_j}|} d_i \tag{5.5}$$

Nachdem die Summe über alle Distanzen der Messpunkte zum Punkt P_j berechnet ist, kann der Tiefenwert für P_j aus den Tiefenwerten (r_i) der Messpunkte interpoliert werden. Für den interpolierten Tiefenwert r_j^* für P_j gilt:

$$r_j^* = \sum_{i=0}^{|R_{P_j}|} \frac{(1 - \frac{d_i}{N_d})}{|R_{P_j}| - 1} \cdot r_i \tag{5.6}$$

P_j wird nun wieder durch den interpolierten Tiefenwert r_j^* erweitert und wird zu R_j^*.

Gewichtung über Flächeninhalte

Grundlage zur Benutzung der Gewichtung über Flächeninhalte zur Berechnung des interpolierten Tiefenwertes für einen Bildpunkt P_j ist das Verfahren zur Bestimmung natürlicher Nachbarn.

Bei der Suche nach natürlichen Nachbarn wird letztendliche eine Voronoi-Zelle für P_j zwischen den umliegenden Messpunkten R_i eingefügt. In Kapitel 4 unter Abschnitt 4.1 wurde bereits angedeutet, dass die Schnittflächen der alten und der lokal neu berechneten Zerlegung eine wichtige Rolle spielen.

Auf Abbildung 5.1 sind die umliegenden Messpunkte $R_{1,2,3,4}$ und der Bildpunkt P_j beziehungsweise der für ihn berechnete Messpunkt R_j^* dargestellt. Die gestrichelten Linien bezeichnen den Verlauf der Ausgangszerlegung und beschreiben die Schnittflächen für die Messpunkte. Alle Schnittflächen A_i zusammen geben den Flächeninhalt $A_{R_j^*}$ der für P_j eingefügten Voronoi-Zelle.

$A_{R_j^*}$ stellt an dieser Stelle den Normierungsfaktor dar. Die Summe der gewichteten Tiefenwerte der Nachbarn darf auch hier nicht größer werden als der maximale, unter den Nachbarn vorkommende, Tiefenwert.

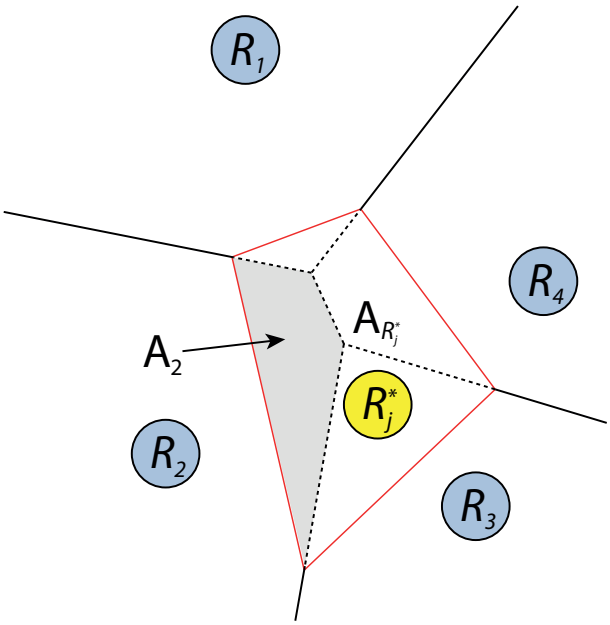

Abbildung 5.1: Natürliche Nachbarn von P_j beziehungsweise des interpolierten Messpunkts R_j^* für P_j

Die Menge R_{P_j} ist an dieser Stelle die Menge aller natürlichen Nachbarn von P_j unter den Messwerten R_i. Das Gewicht w_i des Tiefenwerts r_i eines in der Menge R_{P_j} liegenden Messpunktes lässt sich wie folgt definieren:

$$w_i(R_j^*) = \frac{A_i}{A_{R_j^*}} \tag{5.7}$$

Wenn die Gewichte der Tiefenwerte r_i berechnet ist, dann gilt für den interpolierten Tiefenwert von P_j beziehungsweise R_j^* folgendes:

$$r_j^* = \sum_{i=0}^{|R_{P_j}|} w_i \cdot r_i \tag{5.8}$$

Damit ist der Tiefenwert für P_j berechnet und P_j kann mit R_j^* ersetzt werden.

5.4 Multi Linear Interpolation Considering Color - LIC

Nachdem mit dem MLI Verfahren wieder eine Interpolation vorgestellt wird, folgt mit der *Multi Linear Interpolation Considering Color* beziehungsweise der *multilinearen Interpolation unter Beachtung der Farbe* das zweite Verfahren, das für die Interpolation eine zusätzliche Gewichtung über die Farbe benutzt [ATL07].

So wie beim MLI Verfahren handelt es sich um ein Verfahren, dass die Gewichtung direkt mit in die Tiefenwerte der für einen Bildpunkt gefundenen Messpunkte einrechnet.

Im Wesentlichen baut das LIC Verfahren auf dem MLI Verfahren auf. Ähnlich wie bei der NR und NRC Interpolation wird die Gewichtung der Tiefenwerte erweitert.

Anders als bei Verfahren, die mit einer Wahrscheinlichkeit die Auswahl eines Messpunktes treffen, dürfen die Gewichtungen an dieser Stelle nicht direkt multipliziert werden. Das Problem daran ist, dass die interpolierten Werte verfälscht werden. Das gestaltet sich in der Regel so, dass die interpolierten Werte zu klein werden.

Bevor auf die Kombination der Gewichtungen aus der MLI eingegangen werden kann, muss zunächst eine brauchbare Definition für eine Gewichtung über die Farbe gefunden werden.

Gewichtung über Farbe

Die Farbgewichtung soll direkt in der Berechnung eines interpolierten Tiefenwerts für einen Punkt P_j im Bildbereich verwendbar sein. Das bedeutet, dass die Gewichtung so gestaltet sein muss, dass sie die Tiefenwerte nicht verfälscht.

Es ist möglich die Definition der Farbgewichtung aus dem NRC Verfahren 5.2 so zu erweitern, dass sie sich für die direkte Berechnung eignet. In der Definition für die Wahrscheinlichkeit zur Auswahl eines Messpunktes R_i für einen beliebigen Bildpunkt P_j in Abhänigkeit der Farbe 5.2 ist eine Exponentialfunktion enthalten.

Diese Exponentialfunktion wird für die Farbgewichtung im LIC Verfahren die Grundlage bilden. Sei nun p'_c die Funktion für einen Messpunkt R_i und einen beliebigen Bildpunkt P_j, die angibt, welchen Abstand die Farben der beiden Punkte im Bildbereich besitzt. So gilt für p'_c:

$$p'_c(P_j, R_i) = e^{-\frac{||c_i - c_j||^2}{\sigma_c^2}} \tag{5.9}$$

Bevor das endgültige Gewichtung eines Farbunterschieds für die direkte Berechnung definiert werden kann, muss ein Normierungsfaktor eingeführt werden. Entsprechend sei die Menge R_{P_j}, die Menge aller zu einem beliebigen Bildpunkt P_j gewählten Messpunkte R_i. Dann gilt für den Normierungsfaktor W_c:

$$W_c(P_j, R_{P_j}) = \sum_{i=0}^{|R_{P_j}|} p'_c(P_j, R_i) \tag{5.10}$$

Damit lässt sich das Gewicht eines Farbunterschieds für P_j und ein R_i folgendermaßen formulieren:

$$w_i^c(P_j, R_i) = \frac{p'_c(P_j, R_i)}{W_c} \tag{5.11}$$

In den folgenden zwei Abschnitten wird gezeigt, wie die Kombination von Farb- und Distanz- oder Flächengewichtung konkret beschrieben wird.

Gewichtung über Distanzen und Farbe

Wie bereits erwähnt, handelt es sich beim LIC Verfahren um eine Erweiterung des MLI Verfahrens. Die Gewichtung über die Distanz und die Auswahl der benachbarten beziehungsweise umliegenden Messpunkte R_i für einen Bildpunkt P_j funktioniert also genau gleich.

Für eine bessere Übersicht in der Formel zur Gewichtung, sei w'_i der Gewichtungsfaktor für den Abstand eines Messpunktes R_i zu einem Punkt P_j, wobei R_{P_j} die Menge aller umliegenden

Messpunkte ist. Für w_i' gilt:

$$w_i'(P_j, R_i) = \frac{(1 - \frac{d_i}{N_d})}{|R_{P_j}| - 1} \tag{5.12}$$

Die Interpolation eines Tiefenwertes r_j^* für einen beliebigen Bildpunkt P_j für die Menge der umliegenden Messpunkte R_{P_j}, ist wie folgt zu formulieren:

$$r_j^* = \sum_{i=0}^{|R_{P_j}|} \frac{w_i' + w_i^c}{2} \cdot r_i \tag{5.13}$$

Die Berechnung ist gültig, da die der jeweiligen Gewichtungen nur den Maximalwert 1 erreichen können. Da anschließend wieder halbiert wird, liegt die Gesamtgewichtung auch zwischen 0 und 1.

Gewichtung über Flächeninhalte und Farbe

Auch bei der Flächengewichtung kann für das LIC Verfahren die bereits vorgestellte Gewichtung aus dem MLI Interpolationsverfahren übernommen werden.

Die Erweiterung der Flächengewichtung durch den Farbunterschied ist so zu verwirklichen, wie bei der Distanzgewichtung.

Interpoliert man also für einen beliebigen Bildpunkt P_J mit der Menge R_{P_J} der natürlichen Nachbarn einen Tiefenwert r_j^*, dann gilt:

$$r_j^* = \sum_{i=0}^{|R_{P_j}|} \frac{w_i + w_i^c}{2} \cdot r_i \tag{5.14}$$

6 Implementierung

Wie im Kapitel 2 bereits angekündigt wird, ist die Implementierung der Interpolationsverfahren für die Framework ROS umgesetzt. Des Weiteren wird unter Abschnitt 2.1 auf die allgemeine Funktionsweise von ROS eingegangen. Zur Wiederholung, für ROS implementierte Programme nennen sich Knoten oder Nodes.

Der im Zuge dieser Arbeit realisierte Knoten trägt den Namen *RGBD_Creator* und stellt sämtliche, vorgestellte Algorithmen zur Interpolation zur Verfügung. Der Name RGBD_Creator rührt daher, dass zunächst ein Tiefenbild interpoliert wird, was im Anschluss auf 3D-Punkte abgebildet wird. Die interpolierte Punktwolke befindet sich im 3D-Laserscanner-Koordinatensystem.

6.1 Struktur der Software

In der internen Struktur lässt sich der RGBD_Creator-Node in zwei größere Teile zerlegen. Diese Teile wären zum einen das Hauptprogramm und zum anderen die Interpolationsalgorithmen.

Das Hauptprogramm sorgt dabei für die Datenaufnahme, Datenvorbereitung, Datenspeicherung und die Veröffentlichung der Ergebnisse. Die Berechnungen werden durch die Algorithmen durchgeführt.

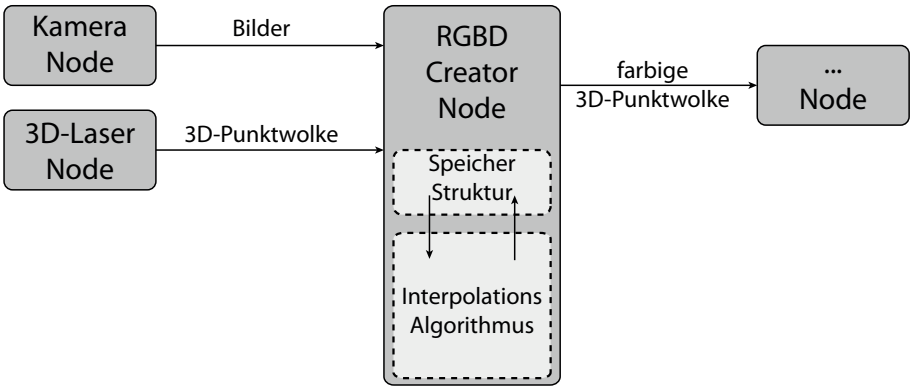

Abbildung 6.1: Struktur des RGBD_Creators

Für die Speicherung von Daten im Hauptprogramm gibt es für die Algorithmen eine Besonderheit. Beim Programmstart werden einmalig feste Werte, in Form von Vektoren oder Matrizen, geladen. Diese sollten nicht verändert werden. Das heißt, dass ein Algorithmus diese Daten nur lesen darf.

Das Gleiche gilt für die zwischengespeicherten Bild- und Laserscannerdaten. Würde etwas an diesen Daten geändert werden, dann hätte das eine Verfälschung der Interpolationsergebnisse zur Folge.

Wie in Kapitel 2 angesprochen, werden in dieser Arbeit unterschiedliche Bibliotheken für die Berechnung einer Interpolation benutzt. Es ist daher nötig die abgebildeten Messpunkte für die Interpolationsalgorithmen als unterschiedliche Typen zu speichern. Die Interpolationsalgorithmen können sich dann die benötigten Daten eigenständig aus dem Hauptprogramm laden.

Damit die Bild- und Laserscannerdaten aufgenommen werden können, sind zwei *Callbacks* definiert. Callbacks sind Methodenaufrufe, die getätigt werden, wenn ein Signal in irgendeiner Form eintrifft.

Damit diese beiden Methoden aufgerufen werden, muss das Hauptprogramm die richtigen Nachrichtenkanäle (Topics) abonnieren. Hierfür werden in ROS sogenannte *Subscriber* benutzt.

Hat ein Algorithmus die Interpolation abgeschlossen, dann muss die resultierende, farbige 3D-Punktwolke publiziert werden. Die Publikation findet auf einem bestimmten Topic, also Nachrichtenkanal, statt. In ROS werden dafür sogenannte *Publisher* verwendet.

6.2 Implementierung der Interpolationsalgorithmen

Im Verlauf dieser Arbeit werden verschiedene Möglichkeiten zur Bestimmung umliegender, abgebildeter Messpunkte für beliebige Bildpunkte vorgestellt. Darauf aufbauend existieren unterschiedliche Möglichkeiten zur Gewichtung bei der Berechnung der Interpolation.

Zunächst sollte auf ein paar allgemeine Eigenschaften der Implementierung dieser Algorithmen eingegangen werden.

Struktur und Funktion der Interpolationsalgorithmen

Wichtigste Eigenschaft der Implementierung der Interpolationverfahren ist die einheitliche Benutzbarkeit. Diese Eigenschaft stellt eine erhebliche Erleichterung für die Implementierung der restlichen Programmfunktionen dar und erspart lästige Überprüfungen während das Programm aktiv ist.

Aus diesem Grund haben alle Implementierungen der Algorithmen ein einheitliches Interface beziehungsweise eine Basisklasse erhalten. Diese bietet dem Hauptprogramm lediglich die eine Methode zur Verarbeitung der Daten an.

Ein weiterer Vorteil dieser Vorgehensweise ist, dass jeder Algorithmus im Hauptprogramm auf die selbe Art und Weise gespeichert werden kann. Den Ursprung findet diese Art der Implementierung im *Strategy Pattern* [Wika].

Speicherung von Daten

Abgesehen von einigen Parametern für die Interpolation speichert eine Instanz eines Interpolationsalgorithmus keine Daten. Die Ergebnisse werden in den, im Hauptprogramm vorgesehenen, Speicherorten abgelegt.

Alle Eingabedaten, in unterschiedlichen Datentypen, kann ein Interpolationsalgorithmus vom

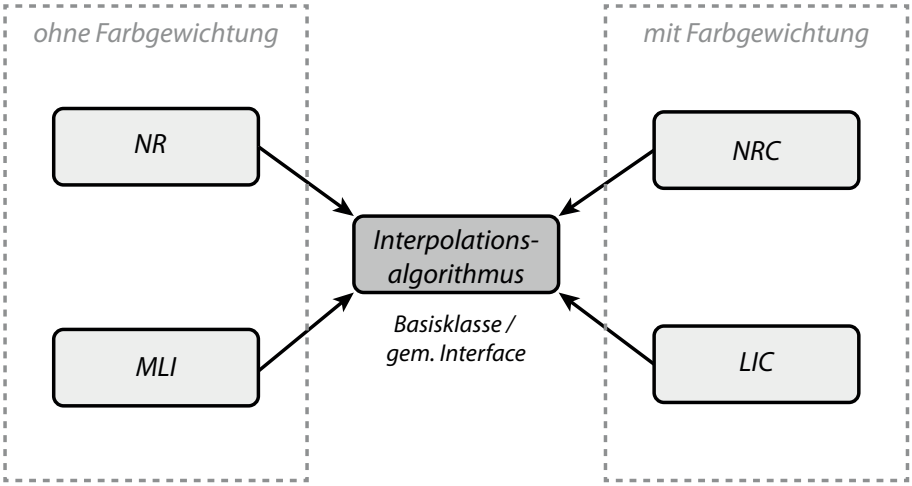

Abbildung 6.2: Basisklasse / gemeinsames Interface

Hauptprogramm abfragen. Hierfür wird eine Referenz bei der Instanziierung übergeben.

Laden eines Algorithmus

Das Laden eines Algorithmus wird über eine statische Methode in der Basisklasse ermöglicht. Das bedeutet, dass die Basisklasse zur Ausführung dieser Methode nicht instanziiert werden muss.

Die Methode erwartet dabei eine Zeichenkette, sprich den Namen eines Algorithmus. Die für einen Algorithmus in ROS hinterlegten Parameter werden bei der Erstellung eines Interpolationsalgorithmus gesetzt.

Algorithmen und ihre Datenstrukturen

In Kapitel 5 werden die verschiedenen Interpolationen vorgestellt. Es existieren dabei vier übergeordnete Ansätze. Dazu zählen das NR, NRC, MLI und LIC Verfahren.

Innerhalb der Verfahren sind wieder unterschiedliche Gewichtungen und Suchen nach umliegenden Messpunkten möglich.

Für die vier Ansätze existiert jeweils eine Klasse. Innerhalb der Implementierung dieser Klassen können unterschiedliche Datenstrukturen und Gewichtungen genutzt werden.

Im vorigen Abschnitt unter Absatz 6.2 wird bereits erwähnt, dass die Interpolationsalgorithmen unterschiedliche Zeichenketten als Bezeichnung haben. Die Zeichenkette kodiert dabei den Interpolationsansatz und die für diesen verwendete Datenstruktur. Welche Rolle diese Zeichenketten beim Laden des Programms spielen, wird in Abschnitt 6.3 erläutert.

Zunächst soll aber definiert werden, was sich hinter den Namen verbirgt. Dazu wird für die jeweilige Datenstruktur die zugehörige Bibliothek angegeben:

NR Verfahren (Nearest Range Reading)

- *NN_SUBDIV*: Delauny-Triangulierung (OpenCV), Bestimmung des nächsten Nachbarn.
- *NN_K_TREE*: k-d-Baum (PCL), auch Bestimmung des nächsten Nachbarn.

NRC Verfahren (Nearest Range Reading Considering Color)

- *NN_COLOR*: k-d-Baum (PCL), Bestimmung einer beliebigen Anzahl umliegender Messpunkte
- *NN_COLOR_RADIUS*: k-d-Baum (PCl), Bestimmung einer beliebigen Anzahl umliegender Messpunkte in einem Radius
- *NN_COLOR_SUBDIV*: Delauny-Triangulierung (OpenCV), Erweiterung der Suche nach dem nächsten Nachbarn für umliegende Messpunkte.
- *NN_COLOR_GRID*: Suche auf einer geordneten Datenstruktur für umliegende Messpunkte

MLI Verfahren (Multi Linear Interpolation)

- *MLI_DISTANCE*: Delauny-Triangulierung (OpenCV), Erweiterung der Suche nach dem nächsten Nachbarn, Interpolation über Distanzgewichtung
- *MLI_DISTANCE_K*: k-d-Baum (PCL), Bestimmung einer beliebigen Anzahl umliegender Messpunkte, Interpolation über Distanzgewichtung
- *MLI_DISTANCE_GRID*: Suche auf einer geordneten Datenstruktur für umliegende Messpunkte, Interpolation über Distanzgewichtung
- *MLI_CGAL_NN*: Delauny-Triangulierung (CGAL), Bestimmung natürlicher Nachbarn, Interpolation über Flächengewichtung

MLI Verfahren (Multi Linear Interpolation)

- *LIC_DISTANCE*: Siehe MLI_DISTANCE, zusätzlich Farbgewichtung
- *LIC_DISTANCE_K*: Siehe MLI_DISTANCE_K, zusätzlich Farbgewichtung
- *LIC_DISTANCE_GRID*: Siehe MLI_DISTANCE_GRID, zusätzlich Farbgewichtung
- *LIC_CGAL_NN*: Siehe MLI_CGAL_NN, zusätzlich Farbgewichtung

6.3 Programmparameter

Für den RGBD_Creator-Knoten und seine Interpolationsverfahren existieren verschiedene Parameter. An dieser Stelle sollen Name und Funktion vorgestellt werden.

Dazu wird zunächst darauf eingegangen, wie Parameter in ROS zu setzen sind.

ROS-Launch-File

Alle Knoten in ROS lassen sich mit einem sogenannten *Launch-File* starten. Ein Launch-File ist als eine Art Startskript aufzufassen. Es wird mit dem Befehl *roslaunch* ausgeführt.

Das ROS Launch-File ist in erster Linie ein XML-Dokument, in dem geschrieben steht, welche Knoten mit welchen Parametern gestartet werden sollen. Abbildung 6.3 zeigt den Ausschnitt aus einem Launch-File für den RGBD_Creator-Knoten und soll zeigen, wie Parameter gesetzt werden.

```
<launch>
    ...
   <param name="camera"        value="/camera/image_color" />
   <param name="laser"         value="/points2" />
   <param name="algorithm"     value="NN_K_TREE" />
    ...
   <param name="neighbours"    value="15" />
   <param name="publish_topic" value="/points2rgb" />
    ...
</launch>
```

Abbildung 6.3: Ausschnitt eines Launch-Files für den RGBD_Creator-Knoten

Wie komplette Launch-Files für einen oder mehrere Knoten geschrieben werden, ist der ROS Dokumentation [ROSc] zu entnehmen.

Parameter für den RGBD_Creator-Knoten

Für Interpolationsverfahren lassen sich eine Vielzahl von Parametern setzen. Darunter sind Parameter für den Programmknoten selbst und für die Interpolationsalgorithmen direkt. In Tabelle 6.1 sind die wichtigsten Parameter für den Programm-Knoten zu finden, um eine gültige Interpolation berechnen zu lassen.

Name	**Datentyp**	**Werte**
laser	string	3D-Laserscanenr Topic Name
camera	string	Kamera Topic Name
algorithm	string	siehe 6.2
neighbours	int	wählbar > 0
radius	double	wählbar > 0
pubish_topic	string	nach Wahl
intrinsics	OpenCV Matrix	gültiger Pfad
distortion	OpenCV Matrix	gültiger Pfad
rotation	OpenCV Matrix	gültiger Pfad
translation	OpenCV Matrix	gültiger Pfad

Tabelle 6.1: wichtigste Parameter des RGBD_Creator-Knoten

Zu den angegebenen Parametern sollte noch etwas angemerkt werden.

Zwei dieser Parameter stehen im Bezug auf die Suche nach umliegenden Messpunkten in einem k-d-Baum. *neighbours* beschreibt im diesen Fall die Anzahl der zu suchenden, umliegenden Messpunkte. Wird mit dem Verfahren NN_COLOR_RADIUS interpoliert, stellt *radius* den Wert für den Radius der Suche nach umliegenden Messpunkten dar.

Der Parameter *intrinsics* steht für die Kameramatrix und *distortion* für die Distorionskoeffizienten. *rotation* und *translation* sind für die Transformation von Laserkoordinatensystem zu Kamerakoordinatensystem. Die Matrizen werden in für OpenCV angelegten XML-Dokumenten [BK08h] angegeben und können direkt über den Pfad geladen werden.

6.4 Ablauf der Datenverarbeitung

In diesem Abschnitt soll deutlich gemacht werden, welche Arbeitsschritte innerhalb des Programms für die Interpolation einer 3D-Punktwolke nötig sind.

Programmstart

Beim Programmstart müssen zunächst alle für die Datenverarbeitung wichtigen Parameter geladen werden. Dabei kann zwischen Parametern unterschieden werden, die direkt für das Hauptprogramm und für die Interpolation gebraucht werden.

Es müssen verschiedene Vektoren und Matrizen in einem OpenCV Format geladen werden. In Kapitel 3 werden bereits nötige Parameter für die Fusion von Kamera und Laserscanner vorgestellt.

Darunter befinden sich ein Vektor mit Distortionskoeffizienten, die Kameramatrix und die Transformation von Laserscanner zu Kamera. Diese Daten werden im Hauptprogramm gespeichert.

Da während die Interpolationsverfahren 3D-Punkte interpoliert werden, wird beim Programmstart einmalig eine Matrix von Richtungsvektoren für die Bildpunkte im Bildbereich der Kamera berechnet. Der Interpolationsalgorithmus kann mit diesen Richtungsvektoren und den interpolierten Tiefenwerten die 3D-Position aus einem Bildpunkt im Kamerakoordinatensystem berechnen [BK08e].

Weil die interpolierten Punktwolken im Laserscanner-Koordinatensystem dargestellt werden sollen, wird während der Programminitialisierung die Rücktransformation von Kamera nach Laserscanner-Koordinatensystem berechnet und gespeichert. Auf die sich ergebende Matrix hat ein Interpolationsalgorithmus lesenden Zugriff.

Datenverarbeitung für ankommende Bild- und Laserscannerdaten

Während beim Programmstart einmalig wiederverwendbare Daten berechnet werden, müssen bestimmte Berechnungen für alle ankommenden Daten wiederholt werden.

Zu diesen Arbeitsschritten zählen die Abbildung der Messdaten auf die Bildebene, deren Speicherung, unter Verwendung unterschiedlicher Datentypen, und die Berechnung des Farb- und

Positionsmittelwerts der Messpunkte. Die Mittelwerte werden später in den Interpolationsalgorithmen für die Bestimmung der Farb- und Verteilungsvarianz benötigt.

Interpolation

Das Hauptprogramm startet über einen Methodenaufruf den Interpolationsalgorithmus. Dabei übergibt es dem Algorithmus die Speicherorte für Visualisierung und die Speicherung der 3D-Punkte.

Der Algorithmus führt darauf hin die Interpolation, unter Verwendung verschiedener Datenstrukturen, durch.

Rückabbildung und -transformation

Um Zeit zu sparen und nicht alle berechneten Tiefenwerte noch einmal zu durchlaufen, läuft dieser Arbeitsschritt im Algorithmus direkt nach der Berechnung der Interpolation für einen einzelnen Bildpunkt ab.

Publikation der interpolierten Punktwolke

Nach dem ein Interpolationsalgorithmus mit der Bearbeitung der Bildpunkte fertig ist, kann das Hauptprogramm die resultierende, interpolierte Punktwolke auf einem Nachrichtenkanal (Topic) publizieren.

7 Ergebnisse

Eine der Aufgaben bei der Zielsetzung dieser Arbeit ist es, die implementierten Interpolationsverfahren auf Genauigkeit und Laufzeit zu überprüfen.

Die Genauigkeit eines Interpolationsverfahrens wird darüber bestimmt, wie hoch die Abweichung von interpolierten Tiefenwerten zu tatsächlichen Tiefenwerten ist. Für die Überprüfung steht ein Datensatz zur Verfügung, der Farb- und dazugehörige Tiefenbilder bereitstellt. Die Laufzeitanalyse wurde unter normalen Betriebsbedingungen durchgeführt.

7.1 Datensatz für die Auswertung

Für die Auswertung kommt der *Middlebury Datensatz 2006 (half-size)* [SH07] zum Einsatz. Der Datensatz besteht aus mehreren Bildsätzen, die für eine bestimmte Szene zwei verschiedene Blickwinkel besitzen. Die Farbbilder für jeden Blickwinkel sind mit drei verschiedenen Beleuchtungen und drei verschiedenen Belichtungszeiten entstanden.

Für die beiden Blickwinkel existiert jeweils eine *Disparity Map*. Die Disparity Map ist ein Schwarzweißbild mit gleicher Auflösung wie die Farbbilder. Die Tiefeninformationen wurden auf diesem Datensatz über ein Stereokamerasystem ermittelt und die Disparity Map gibt die Abweichung zwischen den beiden Kameras an.

Mit den nötigen Informationen zu den Kameras, wie Brennweite und Abstand lässt sich aus der Disparity Map ohne Probleme ein Tiefenbild der gleichen Auflösung berechnen. Diese Angaben werden natürlich beim Middlebury Datensatz zur Verfügung gestellt.

7.2 Auswertungsstrategie

Für die Auswertung wird eine bestimmte Menge von Punkten aus dem Tiefenbild ausgewählt und als Messpunkte in einen Interpolationsalgorithmus eingegeben. Dieser interpoliert dann für die restlichen Bildpunkte die Tiefenwerte.

Zwischen den tatsächlichen Tiefenwerten r_i und den interpolierten r_i^*, wird dann der Abstand berechnet, wobei dieser Abstand als Fehler err_i bezeichnet wird. r_i und r_i^* gehören zum gleichen Bildpunkt. Für die ursprünglichen Messpunkte wird der Fehler nicht berechnet, da die Ausgangswerte nicht von den Interpolationsalgorithmen geändert werden.

$$err_i = |r_i - r_i^*| \tag{7.1}$$

Die Auswahl der Messpunkte erfolgte über das Muster einer Abbildung von Messpunkten eines 3D-Laserscanners, wie es in Abbildung 3.6 in Kapitel 3 gezeigt wird. Die Auflösungen der Bil-

der im Datensatz unterschieden sich teilweise um ein paar Pixel. Die Anzahl der ausgewählten Messpunkte lag dabei etwa um 1000 - 1100.

Standardabweichung und Fehler

Die Standardabweichung wird an dieser Stelle benötigt, um ein Maß für die Streuung der Fehlerwerte zu erhalten.

Zunächst muss über alle Fehlerwerte err_i ein Mittelwert err_{mittel} berechnet werden. Die Anzahl der interpolierten Tiefenwerte und damit die Anzahl der zu betrachtenden Fehlerwerte sei hierfür n.

Dann gilt für die Standardabweichung std_{err}:

$$std_{err} = \sqrt{\frac{\sum_{i=0}^{n}(err_{mittel} - err_i)^2}{n-1}} \tag{7.2}$$

Computersystem für die Auswertung

Die erreichten Laufzeiten für die jeweiligen Verfahren stehen in Zusammenhang mit der Systemleistung. Das für die Ermittlung der Laufzeiten verwendete System, entspricht von seiner Leistung in etwa dem, was auf einem mobilen, autonomen Roboter zum Einsatz kommen kann.

Von Interesse sind an dieser Stelle jedoch nur Prozessor und Hauptspeicher. Es kommen ein Core 2 Duo T9400 mit 2.53 GHz und 4 GB Arbeitsspeicher zum Einsatz.

7.3 Auswertung der Verfahren

Bei der Auswertung werden die Verfahren unter den bereits festgelegten Benennungen geführt. Diese werden in Kapitel 6 Abschnitt 6.2 aufgelistet.

Laufzeit

Die in Tabelle 7.1 aufgelisteten Laufzeiten beziehen sich nur auf die Dauer der Berechnung der Interpolation und nicht auf die Datenvorbereitung. Der mittlere Wert für die Dauer der Datenvorbereitung lag konstant bei ca. $4.4ms$. Abbildung 7.1 macht auf einen Blick deutlich, dass die Laufzeiten sehr unterschiedlich sind.

Die Anstiege der Laufzeiten sind auf unterschiedliche Gründe zurückzuführen. Zunächst wird die Laufzeit dadurch größer, dass ab dem NRC Verfahren nach mehreren umliegenden Messpunkten gesucht werden muss.

Ein weiterer Faktor ist die Gewichtung über die Farbe. Diese macht es nötig, dass im Fall von NRC und LIC auf mehreren Datenstrukturen Zugriffe durchgeführt werden müssen. In den beiden Fällen ist das der Zugriff auf ein Farbbild.

Beim Vergleich von NR und NRC sowie MLI und LIC zeichnet sich in der Grafik ab, dass die jeweiligen Verfahren mit Farbgewichtung etwa eine halbe Sekunde langsamer sind. Die Zeit ergibt sich aus der Dauer für die Berechnung und Verrechnung der Farbgewichtung.

Der übermäßige Anstieg bei MLI_CGAL_NN und LIC_CGAL_NN lässt sich darauf zurückführen, dass der Aufwand zur Berechnung von Flächeninhalten deutlich höher ist, als die direkte Verwendung von Distanzwerten.

Im Allgemeinen zeichnet sich ab, dass die Verfahren, die auf OpenCV zurückgreifen zu den schnelleren gehören.

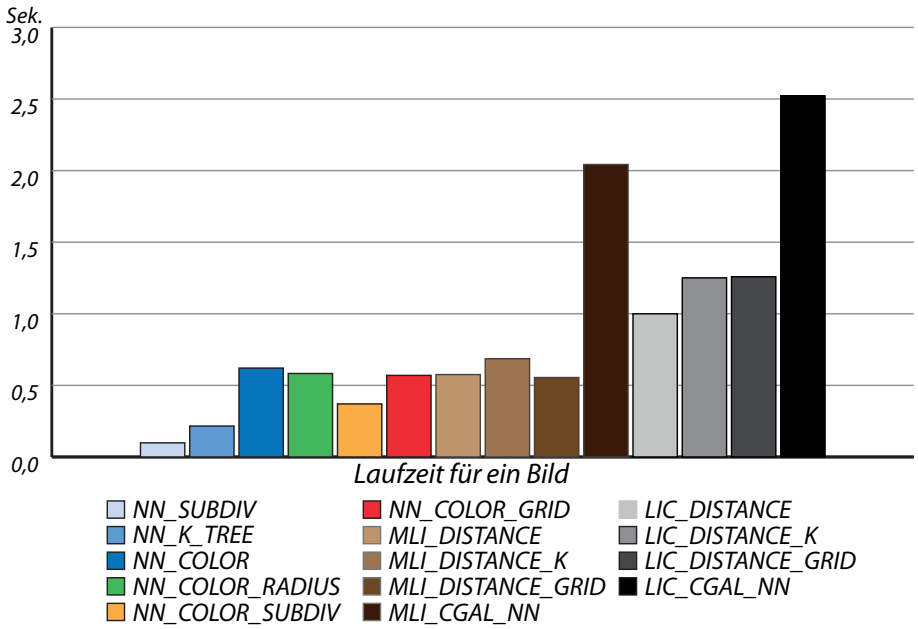

Abbildung 7.1: Laufzeiten aller implementierten Verfahren

Verfahren	**Laufzeit in Sekunden**
NN_SUBDIV	0.099
NN_K_TREE	0.216
NN_COLOR	0.621
NN_COLOR_RADIUS	0.583
NN_COLOR_SUBDIV	0.370
NN_COLOR_GRID	0.570
MLI_DISTANCE	0.575
MLI_DISTANCE_K	0.686
MLI_DISTANCE_GRID	0.5547
MLI_CGAL_NN	2.043
LIC_DISTANCE	0.99
LIC_DISTANCE_K	1.251
LIC_DISTANCE_GRID	1.258
LIC_CGAL_NN	2.523

Tabelle 7.1: Mittlere Laufzeiten der Interpolationsverfahren

Ergebnisse auf Testdaten

Alle Verfahren wurden mit den gleichen Einstellungen und der gleichen Messpunktverteilung auf dem Datensatz getestet.

Genutzte Einstellungen

Einstellungen werden für die Interpolationen nur für die k-d-Bäume vorgenommen.

Die für die Verfahren genutzten Einstellungen waren folgende:

- Anzahl der zu umliegenden Messpunkte: 8
- Radius für die umliegenden Messpunkte: 15

Ergebnisse und ihre Bedeutung

Im Unterschied zu der wissenschaftlichen Arbeit [ATL07], in der die Interpolationsverfahren vorgestellt werden, sind die hier aufgeführten Ergebnisse ungefiltert. Sogenannte Ausreißer, also Tiefenwerte, die an bestimmten Stellen zu stark abweichen, wurden nicht entfernt oder ignoriert.

Jede aufgeführte Grafik steht im Zusammenhang mit einem bestimmten Interpolationsansatz. In den Säulendiagrammen ist die Standardabweichung nicht, wie üblich, direkt an den Säulen für die Fehlerwerte abgetragen. Die Abweichungen fallen bei jedem Verfahren recht groß aus, weswegen es sinnvoller ist, diese in eigenen Säulen darzustellen.

Bei allen Verfahren fällt auf, dass eine sehr hohe Standardabweichung auf den Fehlern existiert. Der Wert liegt in allen Fällen deutlich höher als der Fehlerwert selbst. Dieser Umstand lässt sich relativ leicht erklären.

Ausschlaggebend für die Qualität der Interpolation ist die Anzahl der Messpunkte, die in den Bildbereich abgebildet werden. Speziell in der Situation, wenn sich Objekte vor einem tieferen Hintergrund befinden, kommt es an den Kanten zu Interpolationsfehlern. Es ist möglich, dass für Bildpunkte, die eigentlich zum Hintergrund gehören, Punkte auf den im Vordergrund stehenden Objekten gewählt werden. Für den einzelnen Bildpunkt können sich sehr große Fehler ergeben.

Diese großen Einzelfehler erhöhen die Standardabweichung, obwohl der Durchschnittsfehler deutlich kleiner ist.

Ein weiterer Punkt, der bei der Betrachtung der Ergebnisse auffällt, ist, dass der Durchschnittsfehler des MLI und des LIC Verfahrens größer sind, als die von NR und NRC. Dieser Effekt ist darauf zurückzuführen, dass die MLI beziehungsweise LIC Interpolation Tiefenwerte berechnet, während NR und NRC lediglich gemessene Werte zur Interpolation auswählen.

Angenommen, dass es nur ganz hohe und ganz niedrige Messwerte gäbe. Wird mit dem NR oder NRC Verfahren interpoliert, bestünde das resultierende Tiefenbild immer noch aus ganz hohen oder ganz niedrigen Werten. Mit dem MLI oder LIC Verfahren würden sich Werte dazwischen ergeben.

Das MLI und das LIC Verfahren legen förmlich ein Tuch aus interpolierten Werten über die

Umgebung. Die Verfahren sind für die Interpolation von detailreichen Umgebungen nicht besonders gut geeignet, liefern jedoch gute Werte für homogene Umgebungsstrukturen.

Das günstigste Verfahren

Viele Verfahren liefern, gemessen an der durchschnittlichen Tiefe von 2.3 Meter, gute Werte. Die Abweichungen sollten immer in Relation zu dieser betrachtet werden. Die Verfahren haben je nach Anwendung Vor- und Nachteile. Man sollte man sich vor Augen führen, dass die Ergebnisse unter Verwendung von etwa 1000 Messwerten entstanden sind. Dabei wurde für etwa 300000 Bildpunkte interpoliert.

Nichts desto Trotz heben sich zwei Verfahren besonders hervor. Dazu gehört das NN_SUBDIV Verfahren auf der Basis der Delauny-Triangulierung von OpenCV. Es besitzt eine relativ geringe Abweichung bei der höchsten Geschwindigkeit.

Ein deutlich langsameres Verfahren, jedoch mit der kleinsten Abweichung, ist das NN_COLOR. Es profitiert von der Farbgewichtung und der Möglichkeit mehr als nur die nächsten, umliegenden Messpunkte in die Interpolation einzubeziehen.

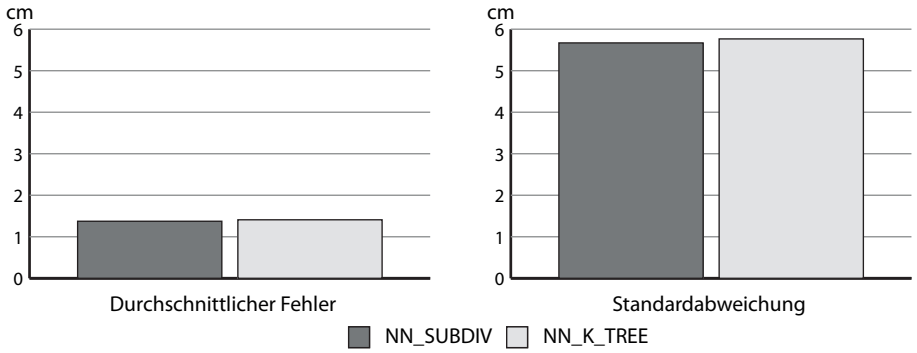

Abbildung 7.2: Abweichung des NR Interpolationsalgorithmus

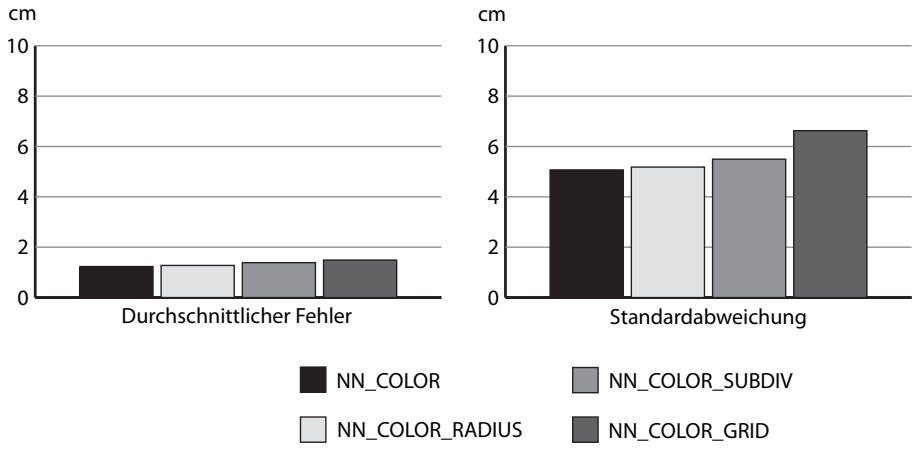

Abbildung 7.3: Abweichung des NRC Interpolationsalgorithmus

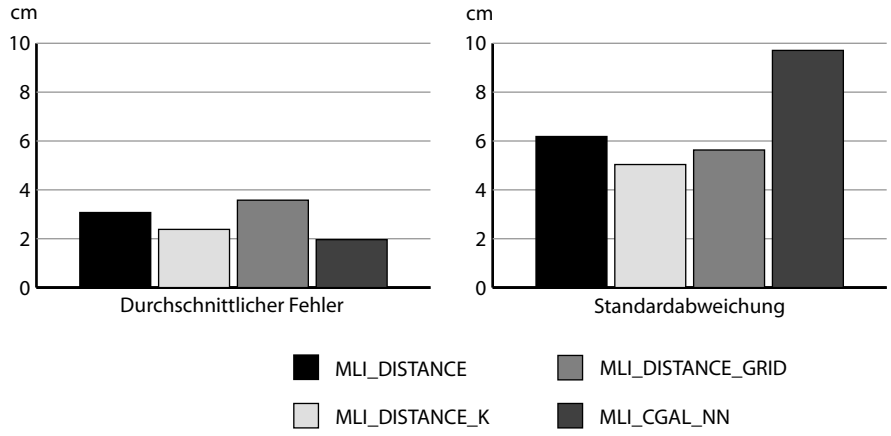

Abbildung 7.4: Abweichung des MLI Interpolationsalgorithmus

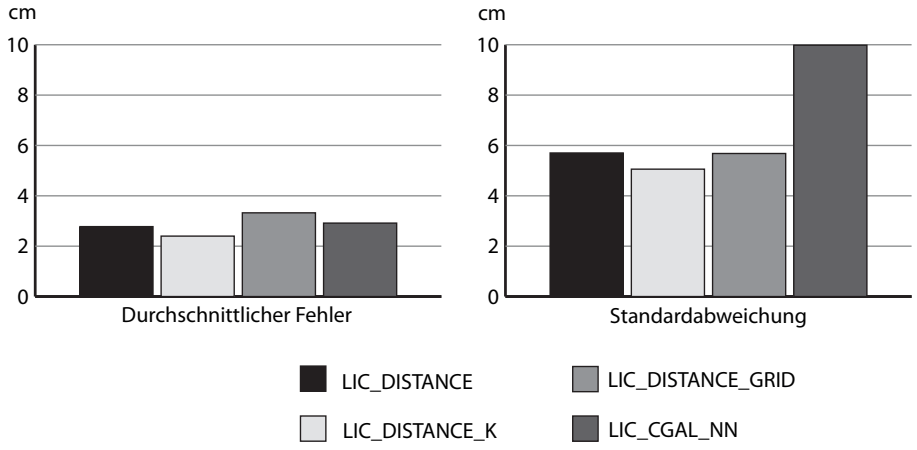

Abbildung 7.5: Abweichung des LIC Interpolationsalgorithmus

Verfahren	**Fehlerdurchschnitt**	**Standardabweichung**
NN_SUBDIV	1.374	5.670
NN_K_TREE	1.409	5.766
NN_COLOR	1.231	5.070
NN_COLOR_RADIUS	1.276	5.182
NN_COLOR_SUBDIV	1.386	5.496
NN_COLOR_GRID	1.489	6.627
MLI_DISTANCE	3.072	6.148
MLI_DISTANCE_K	2.382	5.037
MLI_DISTANCE_GRID	3.578	5.632
MLI_CGAL_NN	1.962	9.709
LIC_DISTANCE	2.777	5.702
LIC_DISTANCE_K	2.400	5.059
LIC_DISTANCE_GRID	3.246	5.681
LIC_CGAL_NN	2.917	9.982

Tabelle 7.2: Mittlere Fehler und Standardabweichung der Interpolationsverfahren

Beispiel von Tiefenbildern und Punktwolken einer Umgebung

Dieser Abschnitt zeigt eine Umgebungsaufnahme und die dazugehörigen Tiefenbilder vom NN_COLOR und vom LIC_DISTANCE_K Verfahren. Des Weiteren sind die dazugehörigen Punktwolken abgebildet. An diesen Punktwolken werden die im vorigen Abschnitt besprochenen Effekte deutlich.

Die Aufnahmen sind mit einer VGA-Kamera und einem 3D-Laserscanner entstanden, der eine Auflösung von 59 mal 29 vielen punkten besitzt. Effektiv konnten circa 1000 Messpunkte für die Interpolation benutzt werden.

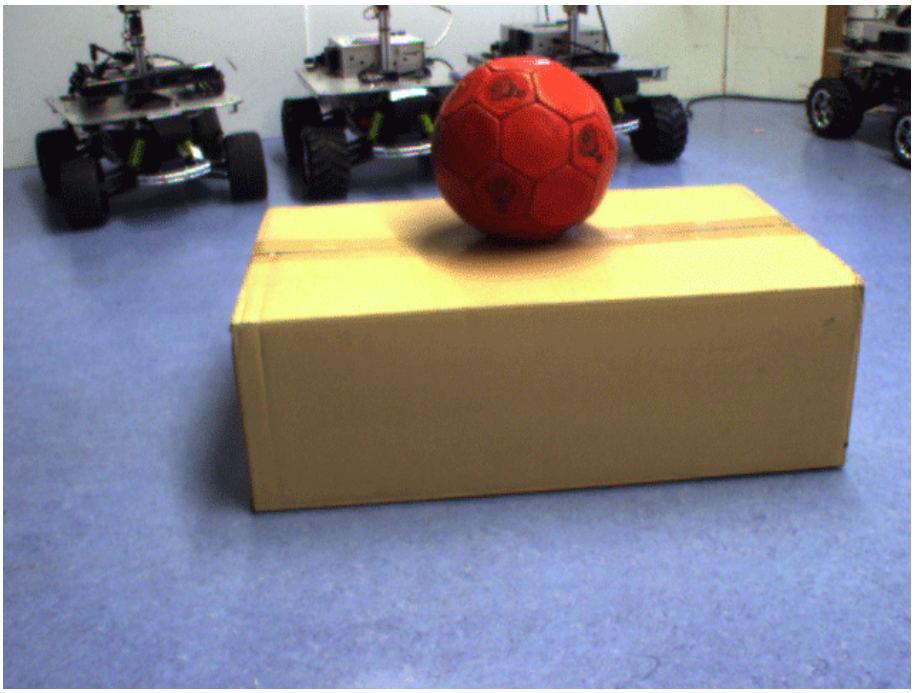

Abbildung 7.6: Bild der Umgebung

Abbildung 7.7: Tiefenbild des NN_COLOR Verfahrens

Abbildung 7.8: Tiefenbild des LIC_DISTANCE_K Verfahrens

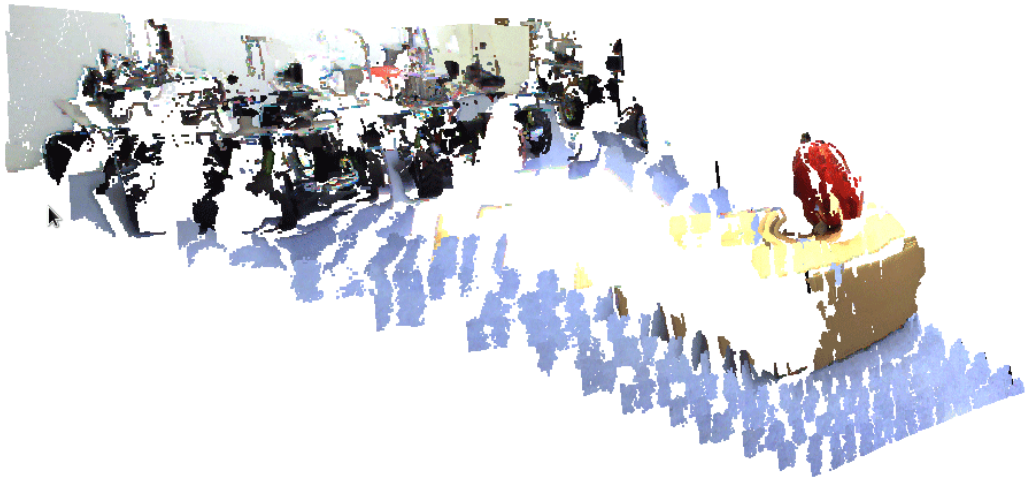

Abbildung 7.9: Interpolierte 3D-Punktwolke des NN_COLOR Verfahrens

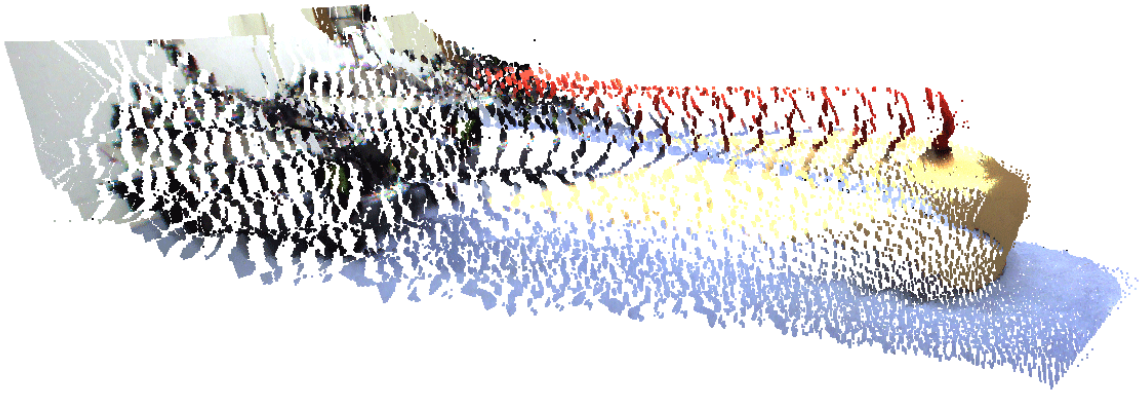

Abbildung 7.10: Interpolierte 3D-Punktwolke LIC_DISTANCE_K Verfahrens

8 Zusammenfassung und Ausblick

Die Zielsetzung, niedrig aufgelöste 3D-Laserscannerdaten unter Verwendung von Bilddaten zu verbessern, kann durch aus als erfolgreich betrachtet werden. Zwar sind die implementierten Interpolationsverfahren nicht im Stande dazu ein perfektes Abbild der Umgebung zu schaffen, trotzdem liefern sie eine gute Annäherung. Die interpolierten Daten sind dabei wesentlich umfangreicher, als die Menge von Daten, die durch die reine Verwendung eines Laserscanners ermittelt werden können. Im Fall der Tests, auf dem in dieser Arbeit verwendeten Datensatz, war es möglich beinahe die dreihundertfache Menge von Tiefenwerten zu produzieren.

Aus den Tests der Interpolationsverfahren hat sich ergeben, dass diese für bestimmte Umgebungen besser oder eben schlechter geeignet sind. Ihre Funktion wird in der Regel kaum von den in einer Umgebung herrschenden Bedingungen eingeschränkt. Trotz wechselnder Blickwinkel und Helligkeiten, waren sie im Stande gute Ergebnisse zu liefern.

Die Stabilität gegenüber wechselnden Lichtverhältnissen, verschafft der Kombination aus Kamera und Laserscanner entscheidende Vorteile gegenüber anderen Sensoren. Die Kinect von Microsoft lässt sich zum Beispiel nicht sinnvoll in Außenbereichen betreiben, weil der Sensor durch Sonnenlicht gestört wird.

Nachteil einiger Interpolationsalgorithmen ist allerdings, dass sie eine gewisse Systemleistung für eine geringe Laufzeit voraussetzen. Auf mobilen, autonomen Robotern, kann diese Systemleistung nicht immer garantiert werden.

Allerdings wird dieser Umstand nicht von Dauer sein. Es werden immer kleinere, leistungsfähigere und gleichzeitig sparsamere Prozessoren entwickelt. Seit geraumer Zeit ist es möglich parallele Berechnungen schnell und effizient auf Grafikprozessoren durchzuführen. Dahingehend werden auch immer mehr Programmbibliotheken, wie OpenCV oder die PCL, weiterentwickelt.

Zu dem wurde in dieser Arbeit nur ein Bruchteil der existierenden Verfahren vorgestellt. Es existieren sicherlich weitere leistungsfähige Verfahren zur Interpolation von Tiefenwerten.

Letzten Endes wird man immer wieder versuchen Verfahren zu entwickeln, die es ermöglichen aus wenigen, verschiedenen Daten, so viel wie möglich zu Ermitteln. Dabei werden Interpolationen häufig eine Rolle spielen.

Literaturverzeichnis

[ATL07] Andreasson, Henrik, Rudolph Triebel und Achim Lilienthal: *Noniterative Vision-based Interpolation of 3D Laser Scans, volume 76*. In: *of Studies in Computational Intelligence*, Seiten 83–90. Springer, 2007.

[BK08a] Bradski, Gary und Adrian Kaehler: *Learning OpenCV*, Seiten 415 – 417. O'Reilly, 2008.

[BK08b] Bradski, Gary und Adrian Kaehler: *Learning OpenCV*, Seiten 371 – 372. O'Reilly, 2008.

[BK08c] Bradski, Gary und Adrian Kaehler: *Learning OpenCV*, Seiten 373 – 374. O'Reilly, 2008.

[BK08d] Bradski, Gary und Adrian Kaehler: *Learning OpenCV*, Seiten 375 – 377. O'Reilly, 2008.

[BK08e] Bradski, Gary und Adrian Kaehler: *Learning OpenCV*, Seiten 396 – 397. O'Reilly, 2008.

[BK08f] Bradski, Gary und Adrian Kaehler: *Learning OpenCV*, Seiten 300 – 302. O'Reilly, 2008.

[BK08g] Bradski, Gary und Adrian Kaehler: *Learning OpenCV*, Seiten 305 – 308. O'Reilly, 2008.

[BK08h] Bradski, Gary und Adrian Kaehler: *Learning OpenCV*, Seite 84. O'Reilly, 2008.

[cga] CGAL, *Computational Geometry Algorithms Library*. http://www.cgal.org.

[Flö11] Flötotto, Julia: *2D and Surface Function Interpolation*. In: *CGAL User and Reference Manual*. CGAL Editorial Board, 3.8 Auflage, 2011. http //www.cgal.org/Manual/3.8/doc_html/cgal_manual/packages.html#Pkg Interpolation2.

[Kle05] Klein, Rolf: *Algorithmische Geometrie*, Seiten 126 – 128. Springer, 2005.

[mata] *Camera Calibration Toolbox for Matlab*. http://www.vision.caltech.edu/bouguetj/calib_doc/.

[matb] *Camera Calibration Toolbox for Matlab - Camera Parameters*. http://www.vision.caltech.edu/bouguetj/calib_doc/htmls/parameters.html.

[RC11] Rusu, Radu Bogdan und Steve Cousins: *3D is here: Point Cloud Library (PCL)*. In: *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.

[ROSa] *Robot Operating System*. http://www.ros.org.

[ROSb] *Robot Operating System - Concept*.
http://www.ros.org/wiki/ROS/Concepts.

[ROSc] *Robot Operating System - Launchfiles*.
http://www.ros.org/wiki/roslaunch/XML.

[SH07] Scharnstein, D. und H. Hirschmüller: *Evaluation of cost functions for stereo matching*. IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2007.

[Wika] *Strategy Pattern*.
http://en.wikipedia.org/wiki/Strategy_pattern.

[Wikb] *TOF-Kamera*.
http://de.wikipedia.org/wiki/TOF-Kamera.