ALBERT-LUDWIGS-UNIVERSITÄT
FREIBURG
INSTITUT FÜR INFORMATIK

Arbeitsgruppe Autonome Intelligente Systeme

Prof. Dr. Wolfram Burgard

Unüberwachtes Lernen von 3D Modellen für nicht stationäre Objekte auf volumetrischen Daten

Diplomarbeit

Michael Ruhnke

November 2008

ALBERT-LUDWIGS-UNIVERSITÄT FREIBURG
FAKULTÄT FÜR ANGEWANDTE WISSENSCHAFTEN

Institut für Informatik
Arbeitsgruppe Autonome Intelligente Systeme
Prof. Dr. Wolfram Burgard

Unüberwachtes Lernen von 3D Modellen für nicht stationäre Objekte auf volumetrischen Daten

Diplomarbeit

Verfasser: Michael Ruhnke

Abgabedatum: 13. November 2008

Erstgutachter: Prof. Dr. Wolfram Burgard
Zweitgutachter: Prof. Dr. Bernhard Nebel
Betreuer: Dr. Giorgio Grisetti

Erklärung

Hiermit erkläre ich, dass ich diese Abschlussarbeit selbständig verfasst habe, keine anderen als die angegebenen Quellen/Hilfsmittel verwendet habe und alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten Schriften entnommen wurden, als solche kenntlich gemacht habe. Darüber hinaus erkläre ich, dass diese Abschlussarbeit nicht, auch nicht auszugsweise, bereits für eine andere Prüfung angefertigt wurde.

(Michael Ruhnke)
Freiburg, den 13. November 2008

Danksagung

Mein Dank gilt all denen, die mich beim Schreiben dieser Diplomarbeit unterstützt haben. Besonderer Dank geht an Prof. Dr. Wolfram Burgard für die Inspiration zu dieser Arbeit und die Möglichkeit diese umzusetzen. Dr. Giorgio Grisetti möchte ich für die Betreuung meiner Arbeit und die freundliche Unterstützung danken. Auch bei Rainer Kümmerle und Bastian Steder möchte ich mich für die aufgewendete Zeit, die vielen hilfreichen Ratschläge und Diskussionen bedanken, sowie für die intensive Durchsicht der schriftlichen Ausarbeitung danken. Ein besonderes Dankeschön geht auch an meine Freundin Heidi, für die zeitintensive Durchsicht dieser Arbeit.

Inhaltsverzeichnis

1 Einleitung

Mobile Roboter werden schon heute für eine Vielzahl an Aufgaben eingesetzt. Zum einen weil manche Aufgaben sehr gefährlich sind, wie zum Beispiel die Suche nach Verschütteten in Erdbebengebieten oder auch das Entschärfen von Bomben und Mienen. Der Verlust eines Roboters ist wesentlich einfacher zu verkraften als der eines Menschen. Generell sollen mobile Roboter das alltägliche Leben komfortabler und einfacher machen. Beispiele für Aufgabenstellungen, die den Alltag erleichtern können, sind Rasenmähen und Staubsaugen oder eine elektronische Einparkhilfe für Autos. Um dieses Ziel zu erreichen, sollen mobile Roboter ihre Aufgaben möglichst selbständig (autonom) erledigen.

Zentrale Themen der Forschung sind die Exploration von unbekannten Umgebungen, das Aufbauen einer internen Repräsentation der Umgebung (Karte) und das Navigieren auf einer solchen Karte. Das Problem für das Bauen von Umgebungskarten liegt darin, dass simultan die Sensormessungen in eine Karte integriert und die Position des Roboters in dieser Karte bestimmt werden muss, damit die Sensormessungen auch an der richtigen Position in die Karte eingefügt werden. Diese Problemstellung ist unter dem Namen *Simultaneous Localization and Mapping, SLAM (Simultane Lokalisierung und Kartengenerierung)* bekannt.
Die Umgebung wird in Karten sehr häufig nur als belegter und freier Bereich encodiert. Die Informationen, die in solchen Karten enthalten sind, reichen für die Navigation in der entsprechenden Umgebung aus. Die Umgebung besteht jedoch gewöhnlich aus einer Vielzahl an verschiedenen Objekten. Um eine weiterführende Interaktion von Robotern mit realen Objekten der Umgebung zu ermöglichen, müs-

sen diese modelliert werden. Das manuelle Erstellen von Modellen für alle Gegenstände, die einem mobilen Roboter begegnen können, ist wenig praktikabel. Analog zur Idee Karten aus Sensordaten eines Roboters zu erstellen, ist es auch plausibel, Modelle für real existierende Objekte aus den Sensordaten zu lernen. Um diese Aufgabenstellung soll es in dieser Arbeit gehen. Besondere Aufmerksamkeit wird darauf liegen, nicht jedes Objekt einzeln zu modellieren, sondern verschiedene Instanzen eines Objekts zu einem Modell zu kombinieren.

1.1 Zielsetzung

Das Ziel dieser Arbeit ist es eine Software zu entwickeln, die dreidimensionale Modelle für Objekte aus den Laser-Range Daten eines Roboters lernt. Ein gelerntes Modell besteht aus einer 3D Punktwolke und repräsentiert im Idealfall die komplette Oberfläche eines Objektes. Da ein 3D Laser-Range-Scan nur eine Ansicht auf ein Objekt liefert, sollen möglichst komplette Modelle aus den zur Verfügung stehenden Teilansichten gelernt werden. Dabei sollen auch Teilansichten von verschiedenen Instanzen eines Objekttyps in ein Modell einfließen.
Daraus ergeben sich folgende zentrale Problemstellungen, mit denen wir uns in dieser Arbeit auseinander setzen werden:

- Welche Punkte gehören zu einem Objekt und wie gruppiert man Punkte zu einer Einheit?
- Welche Teilansichten gehören zu dem selben Objekt?
- Wie kombiniert man Teilansichten zu einem vollständigen Modell?
- Wie vermeidet man inkonsistente Modelle?

Im Verlauf dieser Arbeit werden unter anderem diese Fragen beantwortet und die hieraus entstandene Implementation vorgestellt.

1.2 Notationen

Zu Beginn einige Notationen, wie sie im Verlauf dieser Arbeit verwendet werden:

Matrizen

Matrizen werden durch Großbuchstaben oder auch große griechische Buchstaben dargestellt.
Beispiele: $A, R, \Omega, ...$

Vektoren

Vektoren werden durch fett gedruckte Kleinbuchstaben dargestellt.
Beispiele: $\mathbf{a}, \mathbf{p}, \mathbf{x}, ...$

Winkel

Winkel werden durch kleine griechische Buchstaben dargestellt.
Beispiele: $\alpha, \beta, \gamma, \theta, ...$
Weiterhin gelten die folgenden Konventionen:

- Zur besseren Lesbarkeit werden Winkel im Folgenden nicht im Bogenmaß, sondern im Gradmaß $(-180°, 180°]$ angegeben.
- Alle Winkel, auch Ergebnisse von Berechnungen, werden als korrekt auf $(-180°, 180°]$ normalisiert angenommen, ohne dass dies nochmals erwähnt wird.
- Die Subtraktion zweier Winkel α, β ist definiert als die jeweils kleinere Distanz zwischen den beiden Winkeln, d.h:
  Es sei $\delta = \alpha - \beta$ die normale Subtraktion der reellen Werte der Winkel.
  Dann gilt im Folgenden: $\alpha - \beta := \begin{cases} \delta & \text{falls } \delta \in (-180°, 180°] \\ \delta + 360° & \text{falls } \delta <= -180° \\ \delta - 360° & \text{falls } \delta > 180° \end{cases}$

1.3 Aufbau

Im folgenden Kapitel werden verwandte Arbeiten aus dem Themenbereich dieser Arbeit kurz vorgestellt. In Kapitel 3 werden die in der Arbeit verwendeten grundlegenden Techniken zu Themen wie Registrierung, Segmentierung, Merkmalsextraktion und Graph Partitionierung erläutert. Desweiteren wird eine geeignete effiziente Datenstruktur erläutert. In Kapitel 4 werden die Struktur der verwendeten Sensormessungen beschrieben und die nötigen Schritte, um eine Datenbank für Objektteilansichten zu erstellen. Anschließend wird in Kapitel 5 eine Technik zum globalen Registrieren von Punktwolken mit Hilfe von Tiefenbildern erläutert. Darauf aufbauend wird in Kapitel 6 ein Ansatz präsentiert, um aus den in Kapitel 4 extrahierten Teilansichten möglichst vollständige Modelle zu lernen. In Kapitel 7 werden die durchgeführten Experimente vorgestellt und deren Ergebnisse präsentiert. Abschließend werden in Kapitel 8 die Ergebnisse zusammengefasst, sowie die Schwierigkeiten und interessante Erweiterungen diskutiert.
Informationen zur verwendeten Hardware finden sich im Anhang.

2 Verwandte Arbeiten

Obwohl das automatische Lernen von Objektmodellen bisher wenig Aufmerksamkeit in der aktuellen Forschung erhalten hat, ist es mit dem Erkennen von Objekten in einer 3D Szene eng verbunden. Um beispielsweise zwei Teilansichten dem selben Objekt zuordnen zu können, muss eine gemeinsame Oberflächenstruktur in einem Teilbereich erkannt werden. In diesen Kontext gehören auch Arbeiten, welche die globale Ausrichtung zweier Punktwolken eines Objektes bestimmen. Auch hier werden identische Oberflächenstrukturen benutzt, um Punktwolken zu registrieren. In der Literatur nennt man dies globale Registrierung. Im Folgenden werden wir exemplarisch einen merkmals- und einen sphärenbasierten Ansatz für die globale Registrierung vorstellen:

Gelfand *et al.* [4] haben einen Ansatz präsentiert, um die globale Registrierung mit Hilfe von *Integral Volume Descriptors* als Merkmal zu berechnen. Eine Integral Volume Deskriptor ist ein eindimensionaler Wert, der von dem eingeschlossenen Volumen der Objektoberfläche an einem bestimmten Punkt abhängt. Ein *branch and bound* Algorithmus wird benutzt, um die bezüglich der Deskriptorendistanz optimalen Korrespondenzen der beiden Oberflächen zu finden und die beiden Oberflächen aneinander auszurichten.

Makadia *et al.* [5] benutzen *Extended Gaussian Images*, kurz EGIs, um zwei Oberflächen aneinander auszurichten. Für die Normale jedes Punktes aus einer Punktwolke wird der Schnittpunkt mit einer Sphäre um das Objekt berechnet. Die Sphäre wird in Zellen unterteilt und in jeder Zelle wird die Anzahl der Normalenschnittpunkte gespeichert. Ein EGI ist dann das Histogramm der Normalenschnittpunkte auf dieser Sphäre. Da es viele Lösungen für die Ausrichtung von zwei Punktwolken zueinander geben kann, wird ein Verifikationsschritt eingeführt, um schlechte Lösungen abzulehnen. Die Verifikation basiert auf zwei Eigenschaften, zum einen wird die Ähnlichkeit der Normalen im Überlappungsbereich berechnet, zum anderen wird die Sichtbarkeit eines jeden Punktes von seiner Observationsposition geprüft. Diese beiden Eigenschaften werden auch in dieser Arbeit Verwendung finden.

Neben Integral Volume Dekriptoren gibt es eine Vielzahl an möglichen Merkmalen, die sich für volumetrischen Daten eignen. Neben den in dieser Arbeit verwendeten Tiefenbildern wäre noch folgender Ansatz zu hervozuheben:

Johnson *et al.* [11] schlagen die Verwendung von so genannten *Spin Images* (Drehbilder) als Merkmale für die Objekterkennung vor. Dabei ist ein *Spin Image* eine 2D Repräsentation der Oberfläche, die einen 3D Punkt umgibt. Dafür ist es notwendig, ein 3D Gittermodell der Daten zu erstellen, da die Oberflächen in einer Punktwolke nicht vollständig bekannt sind. *Spin Images* gelten als sehr robuste Merkmale zum Finden von korrespondierenden Punkten auf 3D Daten.

Folgende Arbeiten konzentrieren sich auch auf das Lernen von Objektmodellen:

Triebel *et al.* [18] haben einen Ansatz zum beaufsichtigten Lernen von 3D Modellen vorgestellt, der auf Assoziativen Markov Netzwerken (AMN) basiert. Im ersten Schritt werden Punkte in einem Trainigsdatensatz mit dem zugehörigen Objekttyp beschriftet. Danach werden alle Punkte in einer 3D Karte klassifiziert bzw. mit dem entsprechenden Objekttyp beschriftet. Einer der gravierenden Nachteile ist, dass in dem Testdatensatz alle zu lernenden Objekttypen vorhanden sein müssen. Das Lernen neuer Objekttypen ist nicht möglich und die Objekttypen müssen von Hand beschriftet werden.

Einen unbeaufsichtigten Ansatz zum Lernen von Objektmodellen auf vollständigen 2D gitterbasierten Karten präsentieren Anguelov *et al.* [1]. Ziel ist es aus verschiedenen Objektinstanzen, für jedes vorkommende Objekt ein Modelltemplate zu lernen. Dabei werden nicht stationäre Objekte als Objektinstanzen aus n Karten extrahiert. Mit Hilfe eines *expectation maximization* (EM) Algorithmus wird zuerst die optimale Anzahl der Modelltemplates bestimmt und in einem weiteren Schritt wird für jedes Template eine 2D Repräsentation gelernt.

Im Prinzip kann dieser Ansatz auch für 3D Karten und Objekte erweitert werden. Eine Einschränkung dieses Ansatzes ist jedoch, dass die zu lernenden Objekte vollständig sichtbar sein müssen. In einer normalen Umgebung sind aber viele Objekte nicht zugänglich und können daher nicht von allen Seiten wahrgenommen werden. In den Experimenten wurden nur Objekte verwendet, die mitten im Raum standen. Sieht der Roboter aber zu unterschiedlichen Zeitpunkten verschiedene Ansichten des selben Objektes oder aber zum gleichen Zeitpunkt mehrere Instanzen des gleichen Objekttyps, ist das Lernen eines möglichst kompletten 3D Modells aus diesen Teilansichten erstrebenswert.

3 Grundlagen

In diesem Kapitel werden einige grundlegende Techniken beschrieben, die im weiteren Verlauf dieser Arbeit Verwendung finden werden. Zuerst wird die Datenstruktur kd-Baum vorgestellt.

3.1 kd-Baum

Ein kd-Baum (Bentley 1975)[21, 19] ist eine Datenstruktur, um k-dimensionale Datensätze in einer Baumstruktur zu organisieren. Eine Baumdatenstruktur besteht aus einem Wurzelknoten, inneren Knoten und Blättern (äußere Knoten), die mit Kanten verbunden werden. Abbildung 3.1 zeigt eine solche Baumdatenstruktur. Es gibt zwei Varianten von kd-Bäumen: homogene und inhomogene. Homogene kd-Bäume speichern in jedem Knoten einen Datensatz, während bei der inhomogenen Variante die inneren Knoten nur Schlüssel enthalten und die Blätter Verweise auf Datensätze speichern. Im Folgenden werden die Konstruktion aus einem Datensatz, die Nächste-Nachbarn-Suche und die Umkreissuche erläutert.

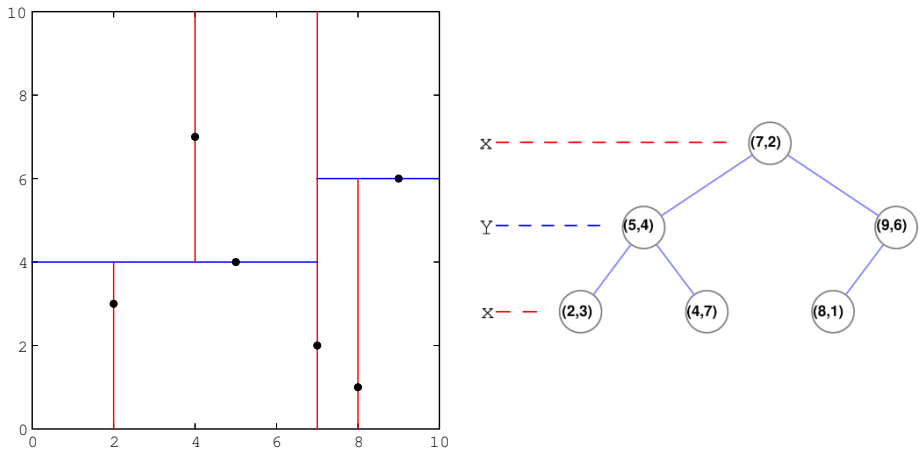

Abbildung 3.1: 2D Beispiel für einen kd-Baum. Die linke Grafik zeigt die geometrische Einteilung einer Punktwolke mit Hyperebenen und die rechte Grafik den dazu gehörenden kd-Baum.

3.1.1 Konstruktion

Um für einen Datensatz einen vollständigen kd-Baum aufzubauen, wird die Punktmenge rekursiv mit Hilfe von Hyperebenen unterteilt, bis in allen Punktmengen nur noch ein Element enthalten ist und somit jedem Punkt ein Blattknoten im kd-Baum zugewiesen ist. In den inneren Knoten werden dabei die Parameter der Hyperebenen gespeichert.
In jeder Rekursion wird eine andere Dimension für die Hyperebene ausgewählt. Beispielsweise könnte die Hyperebene des Wurzelknotens die X-Koordinate aufteilen, dessen Kinderknoten die Y-Koordinate und deren Kinderknoten wiederum die Z-Koordinate.
Um nun die Punktmenge in jeder Rekursion zu unterteilen, berechnet man für alle Punkte den Median bezüglich der Koordinate der ausgewählten Achse. In dem aktuellen Knoten wird der Median und die aufteilende Dimension gespeichert.

3.1.2 NN-Suche

Einer der großen Vorteile von kd-Bäumen ist die schnelle Suche nach dem Nächsten Nachbarn (NN) zu einem k-dimensionalen Punkt P. Im naiven Ansatz müsste die euklidische Distanz für alle Punkte zu P berechnet werden, aber durch die Struktur von kd-Bäumen kann die Berechnung der euklidischen Distanz für einige Teilbäume gespart werden. Die Suche nach dem NN wird dabei als Tiefensuche auf einem Baum implementiert. In jedem Schritt wird eine Approximation für den NN gespeichert. Der Algorithmus terminiert, sobald es keinen Punkt mehr geben kann, der näher an P liegt als der approximative NN. Ausgehend von dem Wurzelknoten wird immer der Kinderknoten ausgewählt, dessen Hyperrechteck den Punkt P enthält. Ein Hyperrechteck entspricht in 2D einem Rechteck und in 3D einem Quader. In der letzten Iteration erhält man eine minimale Region in der P liegt. Für alle Elternknoten dieser Region überprüft man nun, ob der jeweilige Bruderknoten im kd-Baum einen Punkt enthalten kann, der näher an P liegt.
Ausgehend von dem Abstand des momentan besten, approximativen NN zu P als Radius, kann man eine Hypersphäre (im dreidimensionalen Raum eine Kugel) um den Punkt P konstruieren. Alle noch möglichen Kandidaten für den NN liegen in dieser Sphäre. Um zu testen ob ein Bruderknoten noch einen Punkt enthalten kann, der näher an P liegt als der aktuelle approximative NN, muss man nur testen ob die Hypersphäre um P das Hyperrechteck des Bruderknotens schneidet. Ist dies nicht der Fall braucht man den Teilbaum für den Bruderknoten nicht weiter betrachten. Gibt es eine Überschneidung müssen für den Bruderknoten und alle seine Kinderknoten, die die Hypersphäre auch schneiden, die euklidischen Distanzen zu P berechnet werden. Sollte ein Punkt gefunden werden, der näher an P liegt, muss der approximative NN angepasst werden. Sind alle Knoten überprüft oder abgelehnt worden, kennt man den NN und die euklidische Distanz.

3.1.3 Umkreissuche

Die Umkreissuche kann als Spezialfall der NN-Suche interpretiert werden. Analog zur NN-Suche wird für einen Punkt P im kd-Baum das Hyperrechteck gesucht, in dem dieser Punkt liegt. Im Unterschied zur NN-Suche werden alle Punkte gesucht, die in einem festen Umkreis liegen. Daher müssen für alle Elternknoten die Nachbarn betrachtet werden, deren Hyperrechteck sich mit der Hypersphäre mit festem Radius schneiden. Alle Punkte, die in dieser Hypersphäre liegen, werden zu einer Liste hinzugefügt. Sind alle überschneidenden Hyperrechtecke durchsucht, wird die Punktliste als Ergebnis zurückgeliefert.

Ein kd-Baum lässt sich für n Punkte in $\mathcal{O}(nlog(n))$ konstruieren, wobei der Speicherplatzbedarf in $\mathcal{O}(n)$ liegt. Diese Datenstruktur eignet sich besonders für Suchanfragen, wie zum Beispiel Nächste Nachbarn oder Umkreissuche. Hierbei ist jedoch eine Einschränkung, dass die Effizienz von kd-Bäumen mit der Höhe der Dimension k sinkt.

3.2 ICP Algorithmus

Der Iterative Closest Point (ICP) [22, 23] Algorithmus ist die Standardtechnik, um für zwei Punktwolken $\mathcal{D}$ und $\mathcal{M}$ eine Transformation zu berechnen, die $\mathcal{D}$ in das Koordinatensystem von $\mathcal{M}$ überführt. Dies geschieht unter der Annahme, dass die beiden Punktwolken in einem Teilbereich die selbe Struktur abbilden, also eine Überlappung besitzen. Ziel ist es die Punkte aus $\mathcal{D}$, in den mit $\mathcal{M}$ überlappenden Bereich, auf die korrespondieren Punkte in $\mathcal{M}$ zu transformieren. Den Vorgang nennt man auch Registrierung der Punktwolken $\mathcal{D}$ und $\mathcal{M}$. ICP berechnet in jeder Iteration korrespondierende Punkte aus $\mathcal{D}$ und $\mathcal{M}$, was den Vorteil hat, dass die Korrespondenzen nicht aufwendig vorberechnet werden müssen und eventuell fehlerhafte Vorberechnungen zu einem schlechten Ergebnis führen. Durch die iterative Berechnung der Korrespondenzen, nähern sich diese auch einem lokalen Optimum. Danach wird eine Transformation für $\mathcal{D}$ berechnet, die die Summe der quadratischen Fehler zwischen den korrespondierenden Punktpaaren minimiert. Die berechnete Transformation wird dann auf $\mathcal{D}$ angewendet und die nächste Iteration beginnt, solange die Verringerung des quadratischen Fehlers in der aktuellen Iteration nicht unter einer bestimmten Schranke liegt oder die maximale Anzahl Iterationen erreicht ist.

3.2.1 Korrespondenzen finden

In Iteration k sollen für die Punkte aus der aktuellen Punktwolke P_k und $\mathcal{M}$ korrespondierende Punktpaare berechnet werden. In der ersten Iteration gilt $P_0 = \mathcal{D}$. Da die Punkte aus $\mathcal{D}$ in jeder Iteration transformiert werden, gibt es für jeden der k

Schritte ein P_k. Es gibt verschiedene Möglichkeiten diese Punktpaare zu bilden. Eine intuitive Möglichkeit ist es Punktpaare für die Punkte in $\mathcal{M}$ mit ihren Nächsten Nachbarn in P_k zu bilden. Benutzt man als Datenstruktur kd-Bäume, lassen sich die Korrespondenzen in $O(Nlog(M))$ berechnen, wobei N die Anzahl der Punkte in $\mathcal{D}$ und M die Anzahl der Punkte in $\mathcal{M}$ repräsentiert.

3.2.2 Transformationen berechnen

Ausgehend von zwei Punktmengen mit Korrespondenzen $\mathcal{X} = (x_1, ..., x_n)$ und $\mathcal{Y} = (y_1, ..., y_n)$, soll eine dreidimensionale Transformation berechnet werden, welche den mittleren quadratischen Fehler minimiert. Eine Transformation $T = (R, \mathbf{t})$ für eine Punktwolke besteht aus einem Translationsvektor $\mathbf{t}$ und einer Rotationsmatrix R. Für die Korrespondenzen $\mathcal{X}$, $\mathcal{Y}$ sollen nun R und $\mathbf{t}$ bestimmt werden, so dass folgende Gleichung minimiert wird:

$$e(R, \mathbf{t}) := \frac{1}{N} \sum_{i=1}^{N} \|y_i - (Rx_i + \mathbf{t})\|^2. \tag{3.1}$$

Es gibt verschiedene Methoden, um diesen Ausdruck zu minimieren. Im folgenden benutzen wir die Singulär Wert Zerlegung (*Singular Value Decomposition*, SVD) vorgestellt von Umeyama [24] (Umeyama 1991), um die Rotation zu berechnen.
Im ersten Schritt überführen wir die Gleichung 3.1 in eine Matrix Repräsentation, durch Einführung der $3 \times N$ Matrizen $X = [x_1, ..., x_N]$ und $Y = [y_1, ..., y_N]$. Dabei repräsentiert jeder Datenpunkt eine Spalte. Zusätzlich wird ein N-Dimensionaler Vektor h definiert als $h = (1, 1, ..., 1)^T$. Damit kann man Gleichung 3.1 in folgende Form umformen:

$$e(R, \mathbf{t}) := \frac{1}{N} \|Y - RX + \mathbf{t}h^T\|^2 \tag{3.2}$$

Als nächstes wird eine Normalisierungsmatrix $K = I - (\frac{1}{N})hh^T$ eingeführt und X durch $XK + (\frac{1}{N})Xhh^T$ und Y durch $YK + (\frac{1}{N})Yhh^T$ substituiert. Dies führt zu

$$e(R, \mathbf{t}) = \frac{1}{N} \|YK - RX_K\|^2 + \|\mathbf{t}'\|^2, \tag{3.3}$$

wobei

$$\mathbf{t}' = -\frac{1}{N} Yh + \frac{1}{N} RXh + \mathbf{t}. \tag{3.4}$$

Das bedeutet, dass für die Minimierung $t' = 0$ sein muss.

$$\begin{aligned} \mathbf{t} &= \frac{1}{N} Yh - \frac{1}{N} RXh \\ &= \mu_Y - R\mu_X. \end{aligned} \tag{3.5}$$

μ_Y und μ_X sind die Mittelwerte der Punktwolken $\mathcal{X}$ bzw. $\mathcal{Y}$. Um die optimale Rotationsmatrix R zu berechnen, muss die Kreuzkovarianzmatrix Σ_{XY} definiert werden als

$$\Sigma_{XY} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \mu_Y)(x_i - \mu_X)^T. \tag{3.6}$$

Des Weiteren sei UDV^T die Singulärwertzerlegung von Σ_{XY}. Mit Hilfe des Lemma von Umeyama [1991] wird R eindeutig bestimmt, falls der Rang von Σ_{XY} mindestens zwei ist. In diesem Fall wird R berechnet mit

$$R = USV^T, \tag{3.7}$$

wobei

$$S = \begin{cases} I & \text{falls } det(\Sigma_{XY}) > 0 \\ & \text{oder falls } rank(\Sigma_{XY}) = 2 \wedge det(U)det(V) = 1 \\ diag(1, 1, ..., 1, -1) & \text{falls } rank(\Sigma_{XY}) = 2 \wedge det(U)det(V) = -1 \\ & \text{oder } det(\Sigma_{XY}) < 0. \end{cases} \tag{3.8}$$

Die Unterscheidungen in Gleichung 3.8 sind nützlich, falls der Rang der Kreuzkovarianzmatrix unzureichend ist. Ein Beispiel für so einen Fall ist, dass beide Punktwolken auf einer Ebene verteilt sind. In diesem Fall liefert Gleichung 3.8 eine eindeutige Rotation, obwohl die Rotation auf einer Ebene nicht eindeutig ist.

Zusammengefasst werden die Rotation und Translation zwischen zwei Punktwolken $\mathcal{X}$ und $\mathcal{Y}$ folgendermaßen berechnet:

- Berechnen der Mittelwerte μ_Y und μ_X
- Berechnen der Kreuzkovarianzmatrix Σ_{XY} wie in Gleichung 3.6 beschrieben.
- Berechnen der SVD von Σ_{XY}
- Berechnen von R mit Gleichungen 3.7 und 3.8
- Berechne $\mathbf{t}$ mit Gleichung 3.5

Vorgestellt wurde ICP von Besl und McKay 1992, wobei es zahlreiche Varianten und Implementierungen gibt, die sich durch die Wahl der Korrespondenzen und die Transformationsberechnung unterscheiden. In Besl und McKay [22] ist ein formaler Beweis angegeben, dass ICP in einem lokalen Optimum konvergiert. Im Gegensatz dazu finden Algorithmen für die globale Registrierung das globale Optimum, falls der überlappende Bereich groß genug ist. Die meisten dieser Algorithmen nutzen jedoch ICP für die Feinanordnung.

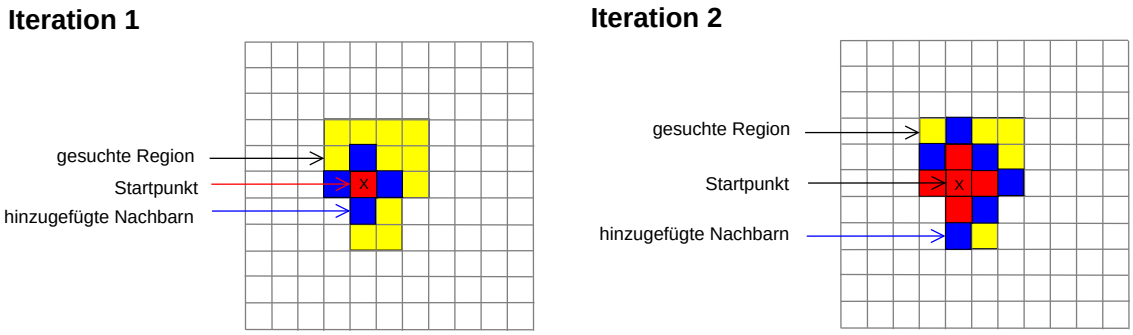

Abbildung 3.2: Das linke Bild zeigt die erste Iteration eines Region Growing Beispiels und das rechte Bild die zweite Iteration. Vom Startpunkt (X) soll eine Region $\mathcal{R}_k$ mit gleicher Farbe (gelb) segmentiert werden. Rote Zellen sind bereits in $\mathcal{R}_k$ enthalten und blaue Zellen wurden in der aktuellen Iteration hinzugefügt. Ziel ist es exakt den gelben Bereich mit roten Zellen zu überdecken.

3.3 Region Growing

Region Growing [25] ist eine Technik, um Punkte oder Pixel aus einem Datensatz in K verschiedene Gruppen einzuteilen. Die Grundannahme ist, dass zusammengehörende Punkte eine gemeinsame Eigenschaft besitzen und in einer zusammenhängenden Region liegen. Als Eigenschaft eignet sich jedes Attribut der Daten, mit dem eine Einteilung erfolgen kann. Auf Bildern wäre beispielsweise die Einteilung nach Farbe oder Helligkeit möglich. In einem 3D-Scan könnten Remissionswerte, Normalen der Punkte oder die Distanz der Punkte zueinander benutzt werden. Das Gruppieren von Punkten, deren Normalen in die gleiche Richtung zeigen, liefert als Resultat zusammenhängende Ebenen. Dabei wird in der Regel ein Treshold t eingeführt, um die Varianz der Eigenschaft zum Startpunkt zu definieren.
Angenommen $\mathcal{P}_\mathcal{D}$ sei die Menge der Punkte in einem Datensatz $\mathcal{D}$, dann besteht eine Region Growing Iteration k aus folgenden Schritten:

- Wahl eines Startpunktes $s \in \mathcal{P}_\mathcal{D}$.
- s und alle Nachbarn von s werden in die Punktgruppe der Region $\mathcal{R}_k$ eingefügt.
- Danach werden rekursiv alle Nachbarn in $\mathcal{R}_k$ hinzugefügt, welche die definierte gemeinsame Eigenschaft besitzen.

Gibt es keinen Nachbarn mit besagter Eigenschaft, ist $\mathcal{R}_k$ vollständig und $\mathcal{R}_{k+1}$ wird in der nächsten Iteration berechnet. Sind alle Punkte $p \in \mathcal{P}_\mathcal{D}$ auf die Regionen R_K aufgeteilt, terminiert der Algorithmus. Abbildung 3.2 illustriert das Verfahren an einem Beispiel. Es wird eine Startposition X gewählt und als Eigenschaft für die

Abbildung 3.3: Beispiele für Tiefenbilder, das linke Bild zeigt das Tiefenbild für einen kompletten 3D Scan. Das Tiefenbild vermittelt einen guten räumlichen Eindruck der Szene. Das rechte Bild zeigt ein Tiefenbild für eine Objektteilansicht.

Gruppierung soll die Farbe der Region dienen. Für die Beispielregion ist dies gelb. Die Zelle in der X liegt, ist bereits in der aktuellen Region $\mathcal{R}_k$ enthalten und daher rot eingefärbt. In der ersten Iteration werden alle Nachbarzellen von der Startzelle X überprüft, ob diese auch in der gelben Region liegen. Alle Nachbarzellen für die dies zutrifft, sind blau gekennzeichnet und werden in $\mathcal{R}_k$ eingefügt. In der zweiten Iteration sind wieder alle in $\mathcal{R}_k$ enthaltenen Zellen rot eingefärbt und die Nachbarzellen, die in dieser Iteration hinzugefügt werden sind wiederum blau eingefärbt. Nachbarzellen die außerhalb der gelben Region liegen, werden abgelehnt.

3.4 Tiefenbilder

Tiefenbilder [16] sind zweidimensionale Darstellungen von dreidimensionalen Punktwolken. Der Vorteil dieser vereinfachten Repräsentation ist, dass effiziente Verfahren der Bildverarbeitung, wie beispielsweise der Harris-Detektor, eingesetzt werden und deren Ergebnisse in die 3D Punktwolke übernommen werden können. Abbildung 3.3 zeigt zwei Tiefenbilder. Dabei sind nahe Bildpunkte dunkel und werden mit zunehmender Entfernung heller. Um ein Tiefenbild zu erstellen, muss zuerst eine Observationsposition p, mit einer Blickrichtung B bestimmt werden. Dabei sollen alle Punkte, die von p in Richtung B zu sehen sind auf eine Sphäre um p, mit Radius r, projiziert werden. Jeder Punkt der Punktwolke, wird auf die Bildsphäre projiziert und die Entfernung als Tiefenwert (Z-Buffer siehe Kapitel 4.3.1) in der Zielzelle gespeichert. Ist in einer Zelle bereits ein Tiefenwert gespeichert, wird dieser nur überschrieben falls der Tiefenwert kleiner ist als der des Vorgängers. Der Vorgänger wäre in diesem Fall nicht sichtbar aus der gewählten Beobachtungsposition. Die Auflösung des Tiefenbildes hängt von den Winkeln α, β und der Größe der Projektionssphäre ab.

Tiefenbilder eignen sich hervorragend, um die Informationen von 3D Punktwolken bezüglich einer Observationsposition zu repräsentieren. Ein weiterer Vorteil ist, dass

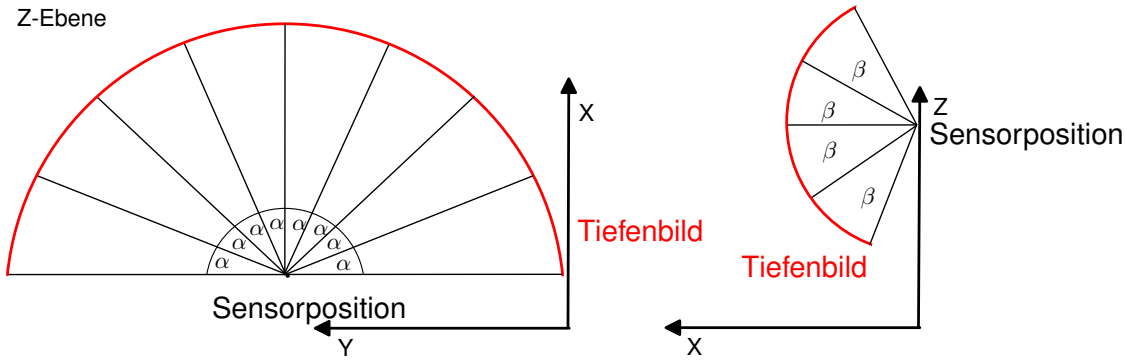

Abbildung 3.4: Das linke Bild zeigt die 2D Ansicht für eine Projektionsebene, also eine Zeile in einem Tiefenbild. Dabei hängt die Anzahl der Zellen in der Reihe von α ab. Diese Projektionsebene wird mit der Winkelschrittweite β, wie das rechte Bild zeigt, variiert. Jeder Punkt einer Puntkwolke wird auf diese Halbkugel projiziert und die Entfernung in der entsprechenden Zelle des Tiefenbildes gespeichert.

für Punkte in einem Tiefenbild die korrespondierenden Punkte in der Punktwolke berechnet werden können. Damit ist es möglich aus Tiefenbildern gewonnene Merkmale auf die Punktwolke zu projizieren. Wir werden diese Repräsentation sowohl für das Finden von ähnlichen Teilstrukturen von Objekten, als auch für das Bewerten von Hypothesen verwenden.

3.5 Interessante Regionen für Merkmale

Für das Finden ähnlicher Teilstrukturen sollen interessante Merkmale auf Tiefenbildern extrahiert werden. Dabei ist zu klären, was informationsreiche Regionen in Bildern sind und wie diese zu berechnen sind. Im Folgenden wird der Moravec-Detektor eingeführt und als Erweiterung der Harris-Detektor, mit dessen Hilfe die Regionen berechnet werden, auf denen die Merkmale in Tiefenbildern verteilt werden. Die extrahierten Merkmale können von Tiefenbildern direkt auf Punktwolken übertragen werden. Diese Merkmale auf Punktwolken werden im Folgenden zum Berechnen von Korrespondenzen für die globale Registrierung in Kapitel 5 verwendet.

3.5.1 Der Moravec-Detektor

Ein Ansatz zur Extraktion von informationsreichen Merkmalen auf Bildern ist der Moravec-Detektor [6, 7, 8, 10] (Hans Peter Moravec 1980). Informationsreiche Merkmale werden als Zentren von Bereichen mit geringer Selbstähnlichkeit definiert. Dies sind Bereiche, die möglichst unähnlich zu nahe liegenden, stark überlappenden Be-

reichen sind.
Um solche Bereiche für einen Bildausschnitt zu berechnen, wird dieser mehrmals leicht innerhalb des Bildes verschoben und dann mit dem „neuen Untergrund“ verglichen. Es sei

- $I(x, y)$ die Intensität des betrachteten Graubildes an der Stelle $(x, y)^T$.
- $(x_0, y_0)^T$ die linke obere Ecke des s_x Pixel breiten und s_y Pixel hohen rechteckigen Bildausschnittes.
- $(\Delta x, \Delta y)$ der Positionsunterschied zwischen verschobenem und unverschobenem Bildausschnitt.

Der Unterschied zwischen verschobenem und unverschobenem Bildausschnitt wird beim Moravec-Detektor mit der Summe der quadratischen Unterschiede berechnet:

$$S = \sum_{x=x_0}^{x_0+s_x} \sum_{y=y_0}^{y_0+y_x} (I(x + \Delta x, y + \Delta y) - I(x, y))^2 \tag{3.9}$$

Dabei spricht ein kleinerer Wert von S für eine große Ähnlichkeit der beiden Bildausschnitte. Als Verschiebungen wurden von Moravec
$(\Delta x, \Delta y) \in \{(1, 0), (1, 1), (0, 1), (-1, 1)\}$ betrachtet und S für jede dieser Verschiebungen berechnet. Als Bewertung des Ausschnittes wird der minimale Wert von S bezüglich der Verschiebungen betrachtet.
Ein Bild kann beispielsweise folgendermaßen auf Merkmale überprüft werden: Zunächst wird eine feste Ausschnittsgrösse gewählt. Anschließend wird jeder Pixel, der das Zentrum für solch einen Ausschnitt sein kann, mit obigem Verfahren geprüft. Eine Möglichkeit wäre es, die n besten Pixel mit den größten Werten für S als Merkmale auszuwählen.

Die Vorteile dieses Verfahrens sind die einfache Berechenbarkeit und die intuitive Verständlichkeit. Es gibt aber auch einige Nachteile. Das Verfahren ist beispielsweise anisotropisch, d. h. es ist abhängig von der Ausrichtung der potenziellen Merkmale im Bild. Ein einfaches Beispiel hierfür ist eine Kante in einem Bild, die parallel zu einer der überprüften Verschiebungen ist und dadurch eine niedrige Bewertung bekommen würde. Läge eine andere Kante nicht parallel zu einer der Verschiebungen, würde diese fälschlicherweise als besseres Merkmal erkannt werden. Ein weiterer Nachteil ist die fest gewählte Ausschnittsgröße, da dadurch nur Merkmale einer bestimmten Größe gefunden werden.

3.5.2 Der Harris-Detektor

Dank seiner starken Invarianz bezüglich Rotation und Beleuchtung, gehört der Harris-Detektor[6, 8, 10] (Harris&Stephens 1988) zu den populärsten Methoden, informationsreiche Merkmale aus Bildern zu extrahieren. Dabei wird die Idee des Moravec-Detektors weitergeführt, so dass auf das Rechnen mit verschobenen Bildausschnitten verzichtet werden kann und stattdessen die Gradienten der Pixelintensitäten zur Berechnung genutzt werden.

Betrachten wir hierzu erneut die Summe der quadratischen Unterschiede zweier verschobener Bildausschnitte (Gleichung 3.9):
$S = \sum_{x=x_0}^{x_0+s_x} \sum_{y=y_0}^{y_0+y_x} (I(x+\Delta x, y+\Delta y) - I(x,y))^2$

Um einen rotationsinvarianten Merkmalsdeskriptor zu erhalten, wird kein rechteckiger Bildausschnitt mehr verwendet, sondern ein radialer Bereich. Desweiteren scheint es sinnvoll, Veränderungen, die an zentralen Positionen eines Merkmalpunktes liegen, stärker zu bewerten als weiter außerhalb liegende. Dadurch wird das Bildrauschen aus der weiteren Umgebung weniger stark berücksichtigt. Damit wird die Robustheit des Merkmals erhöht. Als Gewichtung bietet sich eine Gaussverteilung an:

$$G(x, y, \sigma) = \frac{e^{-(x^2+y^2)/2\sigma^2}}{2\pi\sigma^2} \tag{3.10}$$

Die mit 3.10 angepasste gewichtete Summe der quadratischen Unterschiede ist dann:

$$S_G(x, y, \sigma) = \sum_{x',y'} G(x'-x, y'-y, \sigma) \cdot (I(x'+\Delta x, y'+\Delta y) - I(x', y'))^2 \tag{3.11}$$

Das Zentrum des zu beurteilenden Bildbereiches liegt dabei in (x, y). Dies ist auch die vermutete Merkmalsposition. Mit der Standardabweichung σ kann bestimmt werden, wie stark das weitere Umfeld berücksichtigt wird.

Für kleine $\Delta x, \Delta y$ lässt sich $I(x+\Delta x, y+\Delta y)$ mit den Gradienten (partiellen Ableitungen) der Intensitäten $\delta I/\delta x$ und $\delta I/\delta y$ linear mit der Taylor-Reihe erster Ordnung approximieren:
$I(x+\Delta x, y+\Delta y) \approx I(x,y) + \left(\frac{\delta I}{\delta x}(x,y) \cdot \Delta x + \frac{\delta I}{\delta y}(x,y) \cdot \Delta y\right)$

Eingesetzt in Gleichung 3.11 führt das zu:

$$
\begin{aligned}
S_G &\approx \sum_{x',y'} G(x'-x, y'-y, \sigma) \cdot \left(I(x',y') + \frac{\delta I}{\delta x}(x',y') \cdot \Delta x + \frac{\delta I}{\delta y}(x',y') \cdot \Delta y - I(x',y') \right)^2 \\
&= \sum_{x',y'} G(x'-x, y'-y, \sigma) \cdot \left(\frac{\delta I}{\delta x}(x',y') \cdot \Delta x + \frac{\delta I}{\delta y}(x',y') \cdot \Delta y \right)^2 \\
&= \sum_{x',y'} G(x'-x, y'-y, \sigma) \\
&\quad \cdot \left(\left(\frac{\delta I}{\delta x}(x',y') \cdot \Delta x \right)^2 + 2 \frac{\delta I}{\delta x}(x',y') \frac{\delta I}{\delta y}(x',y') \Delta x \Delta y + \left(\frac{\delta I}{\delta y}(x',y') \cdot \Delta y \right)^2 \right) \\
&= \Delta x^2 \sum_{x',y'} \left(G(x'-x, y'-y, \sigma) \cdot \left(\frac{\delta I}{\delta x}(x',y') \right)^2 \right) \\
&\quad + \Delta x \Delta y \sum_{x',y'} \left(G(x'-x, y'-y, \sigma) \cdot 2 \frac{\delta I}{\delta x}(x',y') \frac{\delta I}{\delta y}(x',y') \right) \\
&\quad + \Delta y^2 \sum_{x',y'} \left(G(x'-x, y'-y, \sigma) \cdot \left(\frac{\delta I}{\delta y}(x',y') \right)^2 \right) \\
&= \begin{pmatrix} \Delta x & \Delta y \end{pmatrix} \cdot M_H \cdot \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix}
\end{aligned}
\tag{3.12}
$$

Dabei ist M_H die Harris-Matrix, mit

$$
\begin{aligned}
M_H(x,y,\sigma) &= \begin{pmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{pmatrix} \overset{m_{12}=m_{21}}{=} \begin{pmatrix} m_{11} & m_{12} \\ m_{12} & m_{22} \end{pmatrix} \\
&= \sum_{x',y'} \left(G(x'-x, y'-y, \sigma) \cdot \begin{pmatrix} \left(\frac{\delta I}{\delta x}(x',y') \right)^2 & \frac{\delta I}{\delta x}(x',y') \cdot \frac{\delta I}{\delta y}(x',y') \\ \frac{\delta I}{\delta x}(x',y') \cdot \frac{\delta I}{\delta y}(x',y') & \left(\frac{\delta I}{\delta y}(x',y') \right)^2 \end{pmatrix} \right)
\end{aligned}
\tag{3.13}
$$

Eine Möglichkeit die Gradienten auf dem diskreten Bild zu approximieren, ist beispielsweise:
$\frac{\delta I}{\delta x}(x,y) = I(x+1,y) - I(x-1,y)$ und $\frac{\delta I}{\delta y}(x,y) = I(x,y+1) - I(x,y-1)$

Die Harris-Matrix M_H beschreibt die Struktur der Intensitäten im Umfeld des potentiellen Merkmals. Die Stärke der Intensitätsveränderungen in diesem Bereich wird durch die Größe der Eigenwerte λ_1, λ_2 dieser 2×2-Matrix beschrieben. Dank der Verwendung einer radialen Gewichtungsfunktion wie der Gaussverteilung, wird

eine rotationsinvariante Beschreibung der Merkmalsgüte erreicht.

Dabei sind drei verschiedene Fälle von Bedeutung:

- Beide Eigenwerte sind sehr klein, d. h. $\lambda_1 \approx 0$ und $\lambda_2 \approx 0$. Dies spricht für einen Bereich mit einheitlicher Intensität.
- Einer der Eigenwerte ist sehr klein, während der andere ein großer positiver Wert ist, d. h. $\lambda_1 \approx 0 \wedge \lambda_2 \gg 0$ oder $\lambda_1 \gg 0 \wedge \lambda_2 \approx 0$. Dies spricht für eine Kante. Die Richtung der Kante ist orthogonal zum Eigenvektor des größeren Eigenwertes.
- Beide Eigenwerte sind große positive Werte, d. h. $\lambda_1 \gg 0 \wedge \lambda_2 \gg 0$. Dies spricht für ein interessantes Merkmal.

Harris und Stephens [8] schlagen folgende Funktion für die Beurteilung der Qualität eines Merkmals vor:

$$Q = \lambda_1\lambda_2 - k(\lambda_1 + \lambda_2)^2 = \det(M_H) - k \cdot \mathrm{trace}^2(M_H) \tag{3.14}$$

Dabei ist $\det(M_H) = m_{11}m_{22} - m_{12}^2$ die Determinante der Harris-Matrix und $\mathrm{trace}(M_H) = m_{11} + m_{22}$ die entsprechende Spur. Auf diese Art müssen die Eigenwerte nicht extra berechnet werden, was deutlich effizienter ist. Mit dem Parameter k wird die Bewertung von Kanten gesteuert. Für negative Werte werden Kanten bevorzugt. Für kleine Werte um Null werden Bereiche mit einer einheitlichen Intensität bevorzugt und für hohe positive Werte werden interessante Merkmale berechnet.

Wurde M_H für jeden Bildpunkt mit einem festen σ berechnet, könnten die Merkmale dann auf einer Auswahl lokaler Maxima von Q basieren. Lokale Maxima Kennzeichnen sich dadurch, dass die umgebenden acht Pixel um eine Merkmalsposition niedrige Werte haben.

3.6 Single-Cluster Spectral Graph Partitioning

Der Single-Cluster Spectral Graph Patitioning (SCGP) Algorithmus von Olson *et al.* [14] eignet sich, um Graphen zu partitionieren. Ausgehend von einem Graphen mit N Hypothesen über die paarweise Konsistenz der Knoten, ist das Ziel einer Graph Partitionierung die Einteilung der Knoten in K Gruppen. Jede Partition soll eine maximale Anzahl an konsistenten Knoten enthalten. Nicht konsistente Knoten, im Folgenden auch Ausreißer genannt, sollen möglichst nicht in eine Partition. Das generelle Graph Partitionierungsproblem ist NP-hart. GOODSAC, RANSAC oder auch SCGP liefern ein approximatives Ergebnis in polynomieller Laufzeit. Wie der

Name Single-Cluster schon sagt, berechnet SCGP nur eine Partition. Alle Knoten die nicht zur berechneten Partition gehören, werden als Ausreißer angesehen.
Als Repräsentation des Graphen wird eine $N \times N$ Adjazenzmatrix A für die N Hypothesen aufgebaut, in der das Gewicht $C(i, j)$ für paarweise Konsistenz zwischen zwei Hypothesen i, j in die entsprechende Matrix Zelle eingefügt wird.

$$A_{ij} = C(i, j) \tag{3.15}$$

Die Matrix A ist symmetrisch, da die Konsistenz für ein Hypothesenpaar i, j gleich sein muss zu der Konsistenz für j, i also $C(i, j) = C(j, i)$, für beliebige $i, j \in N$. Ziel ist es nun, aus dieser Matrix eine Gruppe von Hypothesen zu ermitteln, die maximal konsistent zueinander ist. Für einen Graphen bedeutet das, dass eine Menge von Knoten gefunden werden soll, die mit großen Gewichten verbunden sind. Eine Menge von Knoten wird im Folgenden durch einen $N \times 1$ Indikatorvektor repräsentiert. Dabei ist ein Indikatorvektor $\mathbf{u}$ so aufgebaut, dass $u_i = 1$ ist, falls der i-te Knoten in $\mathbf{u}$ enthalten ist und $u_i = 0$ falls nicht. Da jeder Knoten enthalten sein kann oder nicht, gibt es 2^N mögliche Partitionierungen. Damit eine Menge von „maximal konsistenten" Hypothesen gefunden werden kann, muss ein Maß für das Vergleichen von zwei Hypothesenmengen definiert werden. Die gesuchte Metrik sollte folgende Eigenschaften haben:

- Die Metrik sollte mit der Anzahl von Kanten wachsen, die die in der Partition enthaltenen Knoten verbinden. Dies gibt einen Hinweis auf die wachsende Konsistenz der in der Partition enthaltenen Knoten.

- Die Anzahl der enthaltenen Knoten sollte sich negativ auswirken, so dass neue Knoten nur dann hinzugefügt werden, wenn sie die Konsistenz hinreichend erhöhen. Ohne diese Eigenschaft würde das Ergebnis immer alle vorhandenen Knoten enthalten.

Eine Möglichkeit für so eine Metrik ist der mittlere Konsens:

$$r(\mathbf{u}) = \frac{\mathbf{u}^T A \mathbf{u}}{\mathbf{u}^T \mathbf{u}} \tag{3.16}$$

wobei $\mathbf{u}^T A \mathbf{u}$ die Summe aller Kantengewichte des Teilgraphen ist, die in $\mathbf{u}$ enthalten sind und $\mathbf{u}^T \mathbf{u}$ ist die Anzahl der Knoten in diesem Teilgraphen.
Da es keine bekannte Lösung mit polynomieller Laufzeit gibt, mit der $r(u)$ maximiert werden kann, wenn $\mathbf{u}$ ein diskreter Indikatorvektor ist, wird eine approximative Lösung vorgeschlagen. Eine solche approximative Lösung kann berechnet werden, indem die Bedingungen für $\mathbf{u}$ gelockert werden und $\mathbf{u}$ jeden positiven realen Wert annehmen darf.
Die Extremwerte von $r(\mathbf{u})$ können berechnet werden, indem man den Gradienten

von $r(\mathbf{u})$ gleich Null setzt. Da A symmetrisch ist gilt:

$$\nabla r(\mathbf{u}) = \frac{A\mathbf{u}\mathbf{u}^T\mathbf{u} - \mathbf{u}^T A\mathbf{u}\mathbf{u}}{(\mathbf{u}^T\mathbf{u})^2} = \frac{A\mathbf{u} - r(\mathbf{u})\mathbf{u}}{\mathbf{u}^T\mathbf{u}} = 0 \tag{3.17}$$

$$A\mathbf{u} = r(\mathbf{u})\mathbf{u}. \tag{3.18}$$

Dies kann als Eigenvektor Problem aufgefasst werden mit einem Eigenwert $r(\mathbf{u})$. Der maximale erreichbare Wert ist der dominante Eigenwert der Matrix A, der auftritt wenn $\mathbf{u}$ der dominante Eigenvektor ist. Gleichung 3.16 ist auch bekannt als Rayleigh Quotient einer Matrix A. Unter der Bedingung, dass alle $A_{ij} > 0$, garantiert das Perron-Frobenius Theorem [28], dass der maximale Eigenwert von A und der Eigenvektor $\mathbf{u}$ beide positiv sind. Die Bedingung $A_{ij} > 0$ sollte nahezu jede Konsistenzfunktion erfüllen. Ein negativer Wert für eine Konsistenz erscheint jedenfalls wenig sinnvoll.
Der kontinuierliche Indikatorvektor kann als Gewicht der individuellen Hypothesen für die Mitgliedschaft in der optimalen Menge angesehen werden, falls eine Teilmitgliedschaft möglich ist. Da Teilmitgliedschaften im Allgemeinen nicht möglich sind, ist es erstrebenswert, den Indikatorvektor zu diskretisieren. Mit einem geeigneten Grenzwert t lässt sich der diskretisierte Vektor $\mathbf{v}$ berechnen mit:

$$\mathbf{v}_i(t) = \begin{cases} 1 & \text{falls } \mathbf{u}_i \geq t \\ 0 & \text{sonst.} \end{cases} \tag{3.19}$$

Falls die Größe der gewünschten Menge bekannt ist, kann t direkt berechnet werden, so dass $\mathbf{v}$ die gewünschte Anzahl Hypothesen enthält. Da dies im Allgemeinen nicht bekannt ist, muss t auf andere Art bestimmt werden. Dafür gibt es verschiedene Möglichkeiten. Eine davon ist es, der initialen Idee zu SCGP folgend, einen Grenzwert für $\mathbf{v}$ zu berechnen, der $r(\mathbf{v})$ maximiert.
Ein Grenzwert $t_{r_{max}}$ der $r(\mathbf{v})$ maximiert, kann in $\mathcal{O}(N^2)$ berechnet werden, wenn man für N mögliche Grenzwerte t $r(\mathbf{v})$ in $\mathcal{O}(N)$ berechnen kann. Um dies zu erreichen sortiert man zuerst die Elemente von $\mathbf{u}$ und berechnet inkrementell den Wert von $r(\mathbf{v})$ für absteigende t. Betrachtet man nun eine Iteration $n \in (1, ..., N)$, kann man $\mathbf{v}_n$ berechnen mit $\mathbf{v}_n = \mathbf{v}_{n-1} + \mathbf{w}_n$, wobei $\mathbf{w}_n$ ein Indikatorvektor ist, so dass $\mathbf{w}_j = 1$ falls $\mathbf{u}_j = t_n$. Ausgehend von Gleichung 3.16 betrachten wir ausschließlich den Zähler $\mathcal{Z}(\mathbf{v})$ von $r(\mathbf{v})$. Dann gilt:

$$\begin{aligned} \mathbf{v}_n &= \mathbf{v}_{i-1} + \mathbf{w}_n \\ \mathcal{Z}(\mathbf{v}_n) &= \mathbf{v}_n^T A \mathbf{v}_n \\ \mathcal{Z}(\mathbf{v}_n) &= (\mathbf{v}_{n-1} + \mathbf{w}_n)^T A (\mathbf{v}_{n-1} + \mathbf{w}_n) \\ &= \mathbf{v}_{n-1}^T A \mathbf{v}_{n-1} + \mathbf{v}_{n-1}^T A \mathbf{w}_n^T + \mathbf{w}_n^T A \mathbf{v}_{n-1} + \mathbf{w}_n^T A \mathbf{w}_n \end{aligned}$$

$$\mathcal{Z}(\mathbf{v}_n) = \mathcal{Z}(\mathbf{v}_{n-1}) + \sum_{i \in \mathbf{w}, j \in \mathbf{v}} 2A_{ij} + \sum_{i \in \mathbf{w}, j \in \mathbf{w}} A_{ij} \tag{3.20}$$

Für jeden Iterationsschritt, in dem der Grenzwert erhöht wird, muss Gleichung 3.20 berechnet werden, dies ist aber in $\mathcal{O}(N)$ möglich. Für N Iterationen ergibt sich damit eine Laufzeit von $\mathcal{O}(N^2)$.
Während man beweisen kann, dass der Indikatorvektor $\mathbf{u}$ optimal ist, gilt dies nicht für den diskreten Indikatorvektor $\mathbf{v}$. Das globale Optimum kann durchaus für keinen Grenzwert bezüglich $\mathbf{u}$ erreicht werden. Die Ergebnisse dieser approximativen Lösung, für das NP schwierige Graph Partitionierungsproblem, sind gewöhnlich sehr gut.

4 Sensordaten und Objekte

Wie bereits in der Einleitung erwähnt, sollen Objekte aus den realen Sensordaten eines Roboters gelernt werden. Durch die hohe Genauigkeit von Laser-Range-Findern (LRF), ist dieser Sensor erste Wahl für die Aufgabe. In den nächsten Abschnitten, soll es um Objekte und deren Extraktion aus den Sensordaten gehen.

4.1 Objekte

Objekte im Kontext dieser Arbeit sind zusammenhängende und in den Sensoren wahrnehmbare Gegenstände aus der realen Welt, also Stühle, Tische, einzelne Pflanzen usw. Objekte, die aus verschiedenen kleineren Objekten bestehen, werden nur als eine Objekteinheit betrachtet. Ziel der weiteren Betrachtungen ist es, aus den Sensordaten eines 3D-Laser-Range-Finders automatisch Modelle für real existierende Objekte zu lernen. Objekte lassen sich grob in folgende Klassen einteilen:

- **statische Objekte** oder auch **stationäre Objekte** stehen immer an der selben Stelle in der Welt. Entweder weil sie, wie beispielsweise Häuser, unbeweglich sind, oder aus dem Grund, dass sie in der Zeit, in der sie von einem Roboter wahrgenommen werden, nicht bewegt werden. Formal ist ein stationäres Objekt in n Observationen nur an einer Position zu finden.

- **nicht stationäre Objekte** sind bewegliche Objekte die zu n verschiedenen Zeitpunkten an m verschiedenen Positionen zu finden sind. In diese Kategorie gehören beispielsweise Stühle oder Mülleimer, aber auch Türen.

- **dynamische Objekte** bewegen sich während der Sensorabtastung. Beispiele hierfür sind Menschen oder Autos. Formalisiert dargestellt, hat ein dynamisches Objekt keine eindeutige Position in einer Observation.

- **verformbare Objekte** haben wie der Name schon sagt für n Observationen eine dynamische Oberfläche mit m Zuständen.

Die Einteilung mag ein wenig künstlich erscheinen, da sich alle Objekte mit genug Kraft bewegen oder verformen lassen. Auch müssen sich nicht stationäre Objekte bewegt haben, um einen neuen Standort zu erreichen. Doch lassen sich alle Objekte für eine Menge konkreter Observationen auf diese Art einteilen.

Durch die verwendete 3D-Scan Methode sind Scans für dynamische Objekte unbrauchbar, da die 2D Scanebenen durch Bewegung zueinander verschoben werden. Aus diesem Grund werden solche Objekte nicht betrachtet. Dagegen ist die Extraktion von nicht stationären Objekten ein besonders einfacher Fall, da hierfür die Unterschiede in zwei 3D Scans berechnet werden können. Auf diese Objekte werden wir uns daher im Folgenden beschränken. Das Erste Ziel dieses Ansatzes wird es sein, Objektteilansichten nicht stationärer Objekte aus 3D-Scans zu extrahieren.

4.2 3D-Scan

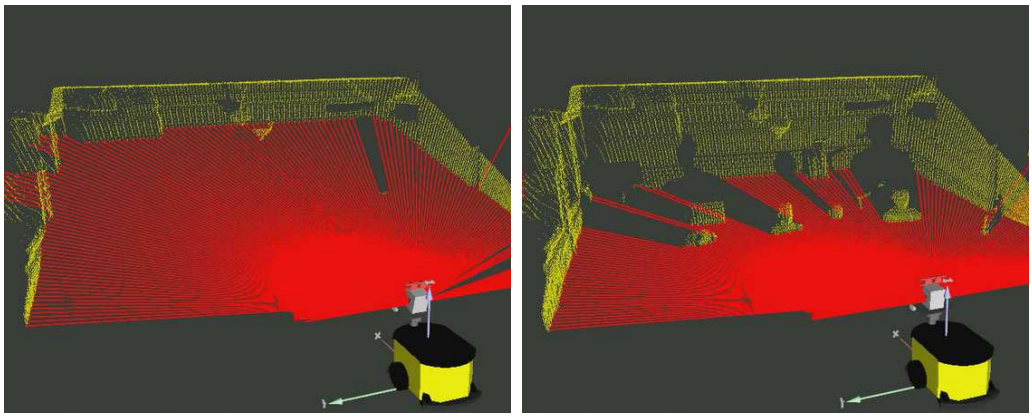

Abbildung 4.1: Entstehung eines 3D Scans durch Neigen eines 2D Laser-Range-Finders: Bilder von verschiedenen Zeitpunkten während eines Scanvorgangs. Die aktuellen Messstrahlen des LRF sind in rot und die entstehende Punktwolke in gelb eingezeichnet.

Um mit einem 2D Laser-Range-Finder (LRF) die Entfernungen für eine 3D Szene zu messen, muss die Messebene des Sensors variiert werden. Dies kann zum Beispiel durch die Neigung des Sensors erreicht werden. Die Daten, die in dieser Arbeit verwendet werden, wurden durch das Neigen eines LRF, mit Hilfe eines Pan-Tilt Gelenks, gemessen. Abbildung 4.1 zeigt dieses Prinzip.
Eine 3D Messung besteht in diesem Fall aus n 2D Scans mit variierenden Neigungswinkeln $\gamma_1, ..., \gamma_n$. Jeder dieser Scans besteht wiederum aus m einzelnen Entfernungsmessungen mit entsprechendem Winkel $\alpha_{1,l}, ..., \alpha_{m,l}$ $(l \leq n)$, innerhalb der Messebene. Jede Messung wird durch einen 3D Punkt mit der gemessenen Entfernung zum Sensor repräsentiert. Dabei ist die Position des Punktes durch den Neigungswinkel γ_i, den Winkel $\alpha_{i,l}$ und die Entfernung d wie folgt definiert:

$$\begin{pmatrix} x_{il} \\ y_{il} \\ z_{il} \end{pmatrix} = \begin{pmatrix} (\frac{\pi}{2} - \gamma_l)z_1 \\ cos(\frac{\pi}{2} - \gamma_l)z_1 \\ sin(\frac{\pi}{2} - \gamma_l)z_1 + z_0 \end{pmatrix} + d \begin{pmatrix} cos(\frac{\pi}{2} - \alpha_{i,l}) \\ sin(\frac{\pi}{2} - \alpha_{i,l}) + sin(\frac{\pi}{2} - \gamma_l) \\ -sin(\frac{\pi}{2} - \alpha_{i,l})cos(\frac{\pi}{2} - \gamma_l) \end{pmatrix}. \tag{4.1}$$

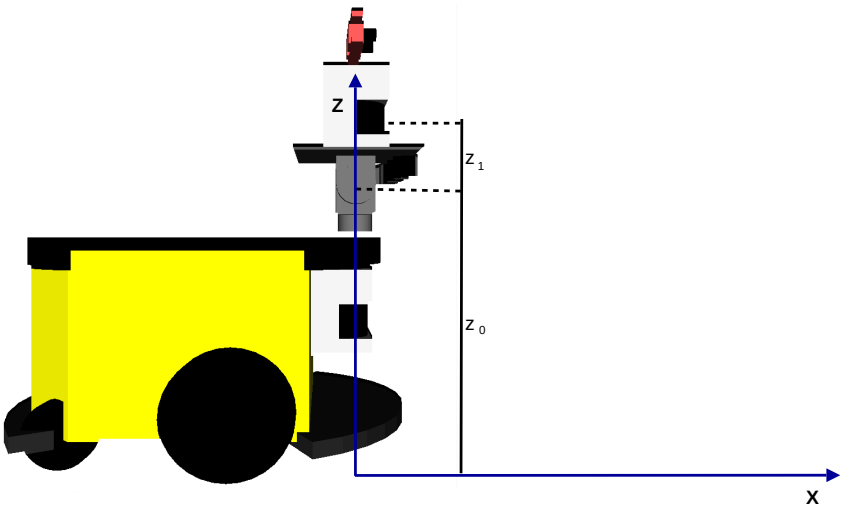

Abbildung 4.2: Die geometrische Ausrichtung der Neigungsebene mit Höhe z_0 und Abstand der Neigungsebene zu Sensor z_1.

Abbildung 4.2 zeigt die Neigungsebene auf dem Roboter mit Höhe z_0 und den Abstand der Neigungsebene zum Sensor z_1. Weitere Informationen zu Sensor und Hardware sind in den Anhang A.1 aufgeführt.
Die gemessene Menge an 3D Punkten wird in einer Punktwolke gespeichert. Als Datenstruktur für eine Punktwolke wurde der im Folgenden beschriebene kd-Baum ausgewählt, da er eine effiziente Berechnung für den nächsten Nachbarn und die Umkreissuche ermöglicht. Diese Eigenschaft ist vorteilhaft für mehrere in dieser Arbeit verwendeten Techniken.

4.3 Teilansichten für Objekte extrahieren

Um 3D-Modelle für nicht stationäre Objekte aus den Sensordaten zu lernen, benötigen wir als Eingabe mindestens zwei 3D-Scans, in denen Objekte verschiedene Standorte bzw. Positionen in der Welt haben. Wir nehmen im Folgenden an, dass die 3D-Scans $S_1, ..., S_N$ eine Überlappung haben und zu unterschiedlichen Zeiten aufgenommen wurden.
Damit Unterschiede in zwei 3D-Scans möglichst genau berechnet werden können, richten wir den 3D-Scan S_{n+1} mit Hilfe des in Kapitel 3.2 beschriebenen Iterative Closest Point Algorithmus an S_n aus. Im Folgenden sei $\mathcal{P}_n$ die Menge der Punkte in S_n und analog $\mathcal{P}_{n+1}$ die Menge der Punkte in S_{n+1}. Der Unterschied auf Punktebene zwischen S_n und S_{n+1} wird berechnet, indem alle Punkte $p \in \mathcal{P}_n$ in eine Punktwolke $\text{Diff}_{n,n+1}$ hinzugefügt werden, die eine der folgenden Eigenschaften erfüllen:

1. In der Sphäre mit Radius r um jeden Punkt $p \in \mathcal{P}_n$ liegt kein Punkt aus S_{n+1}

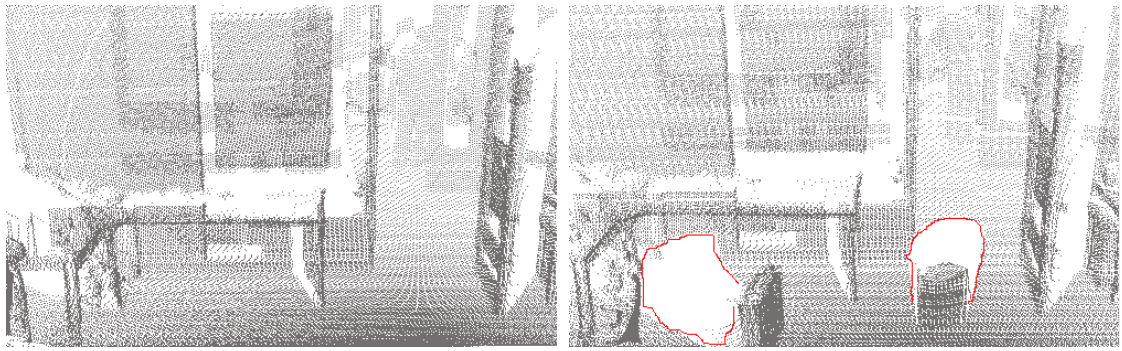

Abbildung 4.3: Beispiel für zwei zu vergleichende 3D-Scans. Der rechte Scan (S_n) enthält zwei Objekte, die im linken Scan (S_{n+1}) nicht vorhanden sind. Rot markiert sind die Schatten hinter den beiden Objekten

2. Die Normalen von $p \in \mathcal{P}_n$ und dem korrespondierendem Punkt $q \in \mathcal{P}_{n+1}$ müssen für einen Grenzwert t folgende Eigenschaft erfüllen:

$$\|Normale_p - Normale_q\| \geq t \tag{4.2}$$

Implementieren kann man 1), indem man für S_{n+1} einen kd-Baum aufbaut und dort für jeden Punkt $p \in \mathcal{P}_n$ eine Umkreissuche im kd-Baum von S_{n+1} mit dem entsprechenden Radius r ausführt. Ist die Ergebnisliste leer, fügt man p in $\text{Diff}_{n,n+1}$ ein. Gibt es Elemente in der Ergebnisliste, verwirft man p.
Um die Normalen für 2) zu berechnen, wird eine Umkreissuche für Punkt p in $\mathcal{P}_n$ durchgeführt und alle Punkte in diesem Umkreis als Nachbarschaft gespeichert. Für diese Nachbarschaft wird eine Ebene berechnet, wie in [23] beschrieben. Auf dieser Ebene wird die Normale berechnet, die den Winkel zur Sensorposition minimiert.

Mit der ersten Eigenschaft kann man nur vollständige Objektansichten berechnen, falls ein Objekt in zwei 3D-Scans weit genug auseinander steht. Beim Verschieben werden Objekte aber auch häufig leicht rotiert, so dass die Flächen des Objektes die Ausrichtung ändern. Solche Punkte können mit Hilfe von 2) berechnet werden. Durch die Sichtlinieneigenschaft von Laserscannern erhält man neben den nicht stationären Objekten auch jeweils den Schatten, den das jeweilige Objekt wirft. Solche Schatten sind beispielsweise im rechten Bild in Abbildung 4.3 rot markiert. Dort sind keine Punkte für den Boden vorhanden. Die hier fehlenden Punkte führen mit Eigenschaft 1) dazu, dass ein Teil des Bodens aus dem linken Bild in Abbildung 4.3 in die Puntkwolke $\text{Diff}_{n,n+1}$ hinzugefügt wird, da diese keinen NN besitzen in S_{n+1}. Da aber im Allgemeinen keine Aussage über Unterschiede hinter einem Hindernis getroffen werden können, fordern wir zusätzlich:

Verwerfe alle Punkte $p \in \text{Diff}_{n,n+1}$, die in S_{n+1} von der Sensorposition für S_{n+1} nicht sichtbar wären.

Analog wird $\text{Diff}_{n+1,n}$ aus den Punkten $q \in P_{n+1}$ und S_n berechnet. Die Sichtbarkeit von Punkten wird mit dem im Folgenden beschriebenen Raytracing Verfahren berechnet.

4.3.1 Berechnung der Sichtbarkeit eines Punktes

Im Allgemeinen gibt es zwei Verfahren, um die Sichtbarkeit eines Punktes p von einer Observationsposition q, in einer Szene zu bestimmen.

Z-Buffer Algorithmus

Eine effiziente Technik ist beispielsweise der Z-Buffer Algorithmus. Hier werden alle Punkte der Szene auf eine Ebene vor der Observationsposition projiziert. Diese Ebene ist in einzelne Zellen unterteilt. Der Abstand des betrachteten Punktes zu dieser Ebene wird in der Zelle gespeichert, in der die Projektion von p nach q auftrifft. Ist in der entsprechenden Zelle schon ein Wert gespeichert, wird dieser Überschrieben, falls der neue Wert kleiner ist. Damit sind alle Entfernungen der sichtbaren Punkte berechnet, sobald alle Punkte in Richtung q projiziert worden sind. Die Qualität dieser Berechnung hängt aber von der Auflösung der Projektionsebene ab. Ist die Auflösung zu grob gewählt, können sich benachbarte Punkte überdecken. Bei einer zu feinen Auflösung landen sich verdeckende Punkte eventuell nur in Nachbarzellen. Das Verfahren ist sehr ähnlich zur Berechnung von Tiefenbildern.

Raytracing

Analog zum Z-Buffer Algorithmus wird auch beim Raytracing eine Projektion zwischen den Punkt $p \in \mathcal{P}$ und einer Observationsposition q berechnet, nur in der entgegengesetzten Richtung. Hier wird ausgehend von q in Richtung eines Punktes p projiziert. Ein Punkt ist dann sichtbar, wenn im Umkreis um den Projektionsvektor $\mathbf{v}_{q,p}$ von q nach p kein Punkt liegt. Der zu betrachtende Umkreis hängt von der Größe der Punkte ab und diese von der Rasterung der Punkte. Bei naivem Vorgehen müsste man auf dem Projektionsstrahl im Abstand des Rasters entlang und für jeden erreichten Punkt eine Umkreissuche durchführen. Liegt ein Punkt in diesem Umkreis ist p von q aus nicht sichtbar. Wird p ohne erfolgreiche Umkreissuche erreicht, ist p von q aus sichtbar. Dies ist deutlich aufwändiger als der Z-Buffer Algorithmus, jedoch auch genauer.
Da die betrachteten Punktwolken in kd-Bäumen gespeichert sind, lässt sich Raytracing effizienter gestalten. Durch Ausnutzen der effizienten Nächste Nachbarn (NN) Berechnung, kann die Schrittweite auf dem Projektionsvektor $\mathbf{v}_{q,p}$ dynamisch bestimmt werden. Für jede Station auf $\mathbf{v}_{q,p}$ wird der NN berechnet und mit Schrittweite $s = dist_{NN} - \epsilon$ auf $\mathbf{v}_{q,p}$ in Richtung p gesprungen. Liegt der NN näher an

der aktuellen Position auf $\mathbf{v}_{q,p}$ als ein vom Raster abhängiger Mindestabstand, ist der Punkt p nicht zu sehen und es kann abgebrochen werden. Erreicht man auf beschrieben Wege p ist p auch sichtbar. Raytracing mit kd-Bäumen gilt für nicht animierte Szenen als eines der effizientesten Verfahren, siehe [20].

4.3.2 Gruppieren der Punkte

Für die Gruppierung von Punkten der Objektansichten soll als Eigenschaft die Distanz zwischen den Nachbarn $d_{ij} \leq t$ benutzt werden. Dies lässt sich mit Hilfe der Umkreissuche in einem kd-Baum effizient realisieren. Jedoch sind einige Besonderheiten zu beachten:

- **Abstand zu Nachbarsegmenten**. Objekte, die segmentiert werden sollen, müssen einen gewissen Abstand zu anderen Objekten haben. Boden, Wände und Decken verbinden alle Objekte die darauf oder daran stehen. Daher müssen diese eigentlich extra gefiltert werden. Durch die Beschränkung auf „nicht stationäre" Objekte wird das Segmentierungsproblem jedoch erheblich erleichtert.

- **Empfindlichkeit gegenüber Sensorrauschen**. Leicht verwischte Kanten können den Abstand zu benachbarten Segmenten verkleinern. Daher ist es sinnvoll vor dem Segmentieren Methoden zur Reduktion von verrauschten Messpunkten anzuwenden.

- **Abnehmende Datendichte bei zunehmender Entfernung**. Der zu durchsuchende Umkreis t muss in Abhängigkeit zur Entfernung des Sensors zu den beteiligten Punkten normalisiert werden.

Mit Hilfe von Region Growing gruppieren wir jeweils die Punkte aus $\text{Diff}_{n,n+1}$ und $\text{Diff}_{n+1,n}$ zu Punktgruppen. Dabei werden nur Punktgruppen mit einer bestimmten Mindestgröße in die Ergebnisliste aufgenommen. Abbildung 4.4 zeigt die berechneten, unterschiedlichen Punktgruppen für die beiden 3D-Scans aus Abbildung 4.3.

Kleinere Punktgruppen repräsentieren sehr häufig Sensorrauschen und ungenau berechnete Normalen an Kanten von Objekten. Die Mindestgröße für zu betrachtende Objekte hängt natürlich von der Sensorauflösung ab. Je höher die Sensorauflösung und um so genauer die Sensormessung, desto kleinere Objekte lassen sich extrahieren und damit lernen.
Desweiteren werden Punktgruppen, die nur eine Ebene repräsentieren verworfen. Um solche Punktgruppen zu erkennen, werden die Normalen berechnet. Innerhalb der Punktgruppe wird dann erneut Region Growing angewandt. Als Eigenschaft wird die Richtung der Normale verwendet. Enthält die berechnete Punktgruppe mehr als 90% der Elemente der originalen Punktgruppe, wird diese als Ebene angesehen und verworfen.

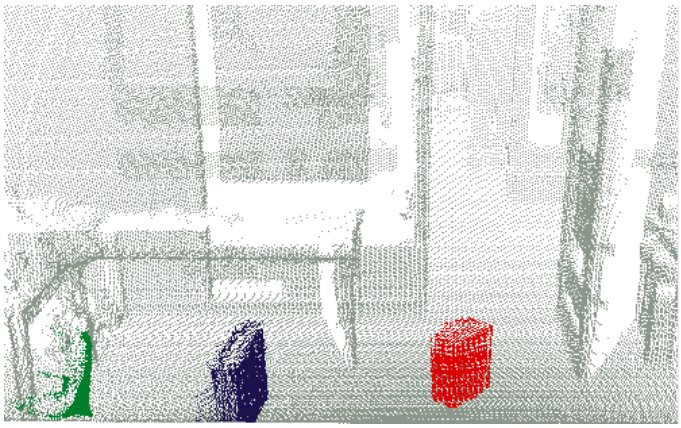

Abbildung 4.4: Punktgruppen für extrahierte „nicht stationäre" Objekte

Im Folgenden wird nun eine Technik zum Registrieren der extrahierten Punktgruppen betrachtet.

5 Globale Registrierung mit Tiefenbildern

Das Registrierungsproblem zwischen zwei Punktwolken $\mathcal{D}$ (Daten) und $\mathcal{M}$ (Modell) besteht darin, dass korrespondierende Strukturen aus $\mathcal{D}$ und $\mathcal{M}$ verschiedene Koordinaten haben. Dies kann durch eine fehlerhafte Lokalisierung oder durch einen veränderten Standort eines Objektes hervorgerufen werden. Um konsistente Karten oder Objektmodelle zu erhalten, sollen alle Punkte aus $\mathcal{D}$ in das gleiche Koordinatensystem wie $\mathcal{M}$ gebracht werden. Dies geschieht unter der Annahme, dass ein überlappender Bereich zwischen den beiden Punktwolken existiert, da es sonst keine sinnvollen Korrespondenzen zwischen $\mathcal{D}$ und $\mathcal{M}$ geben kann. Das Ziel ist es also, eine Transformation bestehend aus Rotationsmatrix R und Translationsvektor $\mathbf{t}$ zu berechnen, mit der $\mathcal{D}$ in das Koordinatensystem von $\mathcal{M}$ überführt werden kann. Abbildung 5.1 zeigt ein Beispiel für die Registrierung einer Puntkwolke in das Koordinatensystem einer anderen.
Verfahren für die globale Registrierung benutzen meistens Merkmale, um korrespondierende Strukturen in $\mathcal{D}$ und $\mathcal{M}$ zu identifizieren.

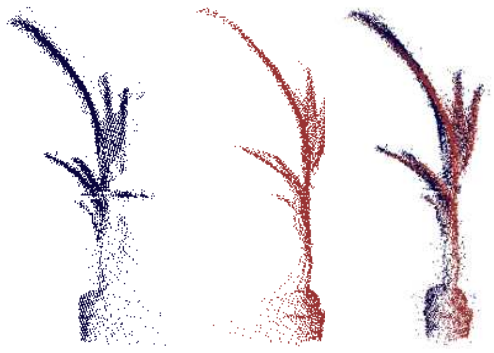

Abbildung 5.1: Zwei Punktwolken $\mathcal{D}$ und $\mathcal{M}$ sollen entsprechend ihrer Struktur aneinander ausgerichtet werden. Rechte Grafik zeigt wie $\mathcal{D}$ erfolgreich in das Koordinatensystem von $\mathcal{M}$ registriert wurde.

Abbildung 5.2: Auszüge einer Tiefenbilder Kollektion für ein initiales Modell. Das erste Tiefenbild zeigt die Ansicht von der Observationsposition. Für einen Punkt ist der 6×6 Merkmaldeskriptor mit abgebildet. Die folgenden drei Tiefenbilder zeigen das selbe Modell aus verschiedenen Beobachtungspositionen und verdeutlichen, dass es sich bei dem Modell nur um eine Teilansicht handelt. Das fünfte Bild zeigt die Modellpunktwolke aus der Observationsposition und die, in der Tiefenbilderkollektion gefundenen, interessanten Punkte in grün.

5.1 Merkmalsextraktion auf Tiefenbildern

Mit der Kombination aus Tiefenbildern (siehe Kapitel 3.4) und dem Harris Detektor (siehe Kapitel 3.5.2) sollen strukturell interessante Merkmale für Punktwolken berechnet werden. Ein Tiefenbild ist eine gute Repräsentation für einen Blickwinkel auf eine Punktwolke, jedoch nicht für die Sicht um ein Objekt herum. Aus diesem Grund werden für jedes Modell $\mathcal{M}^{(k)}$ t Tiefenbilder $\mathcal{DI}_t^{(k)}$ berechnet. Die $\mathcal{DI}_t^{(k)}$ werden in der Ebene des Sensors um das Modell mit einem Radius r und einer Winkelschrittweite von $\alpha = \frac{360°}{t}$ berechnet, wobei das erste Tiefenbild von der Sensorposition $p_i^{(k)}$ berechnet wird. Für die Sensorposition enthält das Tiefenbild die komplette Information für das Objekt aus der Observation. Da die Sensorpositionen aber beliebig um das Objekt verteilt sein können, ist es nicht zu vermeiden, dass die t Tiefenbilder für zwei Objekte zueinander um einen konstanten Winkel ϵ verschoben sind. Je dichter die Tiefenbilder um das Objekt verteilt werden, desto kleiner wird ϵ und durch die Invarianzeigenschaft des Harris-Detektors bezüglich Rotation hat dies für ein genügend großes t keinen großen Einfluss auf die Ergebnisse. Diesem Sachverhalt ist ein Experiment in Kapitel 7.4 gewidmet.

Abbildung 5.2 zeigt vier Tiefenbilder aus verschiedenen Beobachtungspositionen, dabei ist gut zu erkennen, dass es sich jeweils nur um eine Teilansicht handelt und nicht um ein vollständiges Modell.
Auf den Tiefenbildern werden nun in den interessanten Bereichen Merkmale ex-

trahiert. Als Merkmaldeskriptor dienen 6×6 Pixel große Ausschnitte aus dem Tiefenbild. Für einen Punkt im ersten Tiefenbild in Abbildung 5.2 ist ein solcher Merkmaldeskriptor abgebildet. Im Sinne der Problemstellung interessant sind solche Bereiche, die besondere strukturelle Informationen besitzen. Diese Bereiche liegen an den Objektkanten und werden mit Hilfe des Harris Detektors berechnet. Die berechneten Merkmale aus den Tiefenbildern werden im nächsten Schritt auf die entsprechenden Punkte in der Punktwolke abgebildet. In Abbildung 5.2 rechts abgebildet sind die grünen Merkmale auf einer dunkelblauen Punktwolke, die aus der gesamten Tiefenbilderkollektion berechnet wurden.
Sind die Merkmale für $\mathcal{D}$ und $\mathcal{M}$ berechnet, ist die interessante Frage, welche Merkmale aus $\mathcal{D}$ zu den Merkmalen aus $\mathcal{M}$ korrespondieren.

5.2 Berechnen von Merkmalskorrespondenzen

Es gibt verschiedene Techniken für das Berechnen von Korrespondenzen in zwei Merkmalsmengen. Um eine Transformation von zwei 3D Punktwolken $\mathcal{D}$, $\mathcal{M}$ zu berechnen sind mindestens drei korrespondierende Merkmale notwendig. Für die Berechnung der Transformation werden im Folgenden jedoch nur die Merkmalspunktwolken $\mathcal{D}_M$, $\mathcal{M}_M$ betrachtet, nicht die kompletten Punktwolken. Dabei werden die Kernideen von RANSAC, PROSAC und dem in dieser Arbeit verwendeten GOODSAC vorgestellt.

RANSAC [26] (Fischler 1981) steht für „Random Sample Consensus". Dabei handelt es sich um eine Methode, um aus einer Menge möglicher Korrespondenzen eine Transformation zu berechnen. Einer der markanten Vorteile dieses Ansatzes ist es, dass die Menge der Korrespondenzen auch stark mit Ausreißern belastet sein kann und RANSAC trotzdem gute Ergebnisse liefert. Sei m die Mindestanzahl an übereinstimmenden Punkten, die notwendig ist, um die Transformation zu berechnen. Dann wird zufällig eine Menge mit m Korrespondenzen ausgewählt und die Transformation berechnet. Mit dieser Transformation wird $\mathcal{D}_M$ transformiert und die Güte der Transformation berechnet. Die Güte hängt dabei von der Menge an korrespondierenden Merkmalspunkten aus $\mathcal{D}_M$ und $\mathcal{M}_M$ ab, die durch die Transformation aufeinander abgebildet wurden. Diese Menge wird auch Consensus set genannt. Ergebnisse mit einer Consensus set Größe über einem Grenzwert werden gespeichert. Die Idee ist, dass eine große Anzahl an aufeinander abgebildeten Merkmalen für eine gute Lösung spricht. Wird keine Abbruchbedingung erfüllt, beginnt eine neue Iteration und es werden wieder m Punkte zufällig ausgewählt. Mögliche Abbruchbedingungen sind:

- Die maximale Anzahl Iterationen ist erreicht.

- Mindestgröße des Consensus sets ist erreicht oder es sind n Ergebnisse mit bestimmter Consensus set Größe berechnet.

Eine Voraussetzung für RANSAC ist jedoch, dass mehr Punkte vorhanden sind, als zur Bestimmung der Transformation notwendig sind.

PROSAC [27] (Chum 2005) steht für Progressive Sample Consensus und erweitert das Konzept von RANSAC. In einigen Anwendungen kann eine Wahrscheinlichkeit für die Korrespondenz von Paaren aus den beiden Punktmengen gegeben werden. Beispielsweise kann die Ähnlichkeit der korrespondierenden Merkmaldeskriptoren zueinander als Maß für die Wahrscheinlichkeit genutzt werden. Die Idee von PROSAC ist es, dieses zusätzliche Wissen zu nutzen und die m Paare anhand der Wahrscheinlichkeitsverteilung des Vorwissens zu ziehen. Durch Nutzen des Vorwissens erhöht sich die Chance mit weniger Iterationen zu einem guten Ergebnis zu kommen.

GOODSAC [13] (Michaelsen 2006) steht für „Good Sample Consensus" und ist eine alternative Erweiterung für RANSAC. Dieser Ansatz verzichtet auf das randomisierte Auswählen von Merkmalen und wählt stattdessen gute Merkmale aus. Dabei orientiert sich die Auswahl zum einen an der Qualität der Merkmalskombinationen, zum anderen müssen die geometrischen Abhängigkeiten erfüllt sein, damit eine Transformation berechnet werden kann. Als Beispiel hierfür kann man sich drei korrespondierende Merkmale auf einer Geraden vorstellen. Aus dieser geometrischen Anordnung kann keine eindeutige Transformation berechnet werden, da die gleiche Information auch mit zwei Merkmalen beschrieben werden kann und dies für eine Ausrichtung in einem 3D Raum unzureichend ist.
Sind m Paare ausgewählt, wird analog zu RANSAC die Transformation berechnet und mit dieser dann die Consensus Menge. Durch den Verzicht auf den Randomisierungsschritt wird die Berechnung von GOODSAC deterministisch. Der Algorithmus arbeitet wie folgt:

1. Bestimme alle Korrespondenzen der Merkmale aus $\mathcal{D}_M$ und $\mathcal{M}_M$ als Arbeitselemente.
2. Sortiere die Arbeitselemente bezüglich einer Bewertungsfunktion α.
3. Nehme eine bestimmte Anzahl von Elementen vom „guten" Ende der sortierten Liste.
4. Bilde Korrespondenzpaare für die Arbeitselemente, die in $\mathcal{D}_M$ und $\mathcal{M}_M$ eine ähnliche geometrische Anordnung zueinander haben. Für jedes auf diese Weise gefundene Paar werden die Bewertungen aufsummiert und das Korrespondenzpaar in einer neuen Arbeitsliste gespeichert.

5. Falls es noch Arbeitselemente in der Liste gibt und keine Abbruchbedingung (siehe RANSAC) erfüllt ist, springe zu Punkt 2 für die neue Liste der Arbeitselemente.

Sind auf diese Art eine Menge von m Korrespondenzen ausgewählt, wird wieder analog zu RANSAC eine Transformation berechnet und die Güte der sich überdeckenden Merkmale zu der Güte der Korrespondenzen addiert. Auf diese Weise lässt sich für jede Transformation eine GOODSAC Bewertung berechnen. Ist ein eindeutiges Registrierungsergebnis gewünscht, wird die Transformation mit der besten Güte als Ergebnis ausgegeben.
In dieser Arbeit sollen die Ergebnisse der globalen Registrierung als Hypothesen für eine Objektzugehörigkeit dienen. Dies kann nicht alleine auf der Grundlage der GOODSAC Bewertungen entschieden werden, da nicht bekannt ist, ob die betrachteten Punktwolken tatsächlich eine Überlappung besitzen und ob diese zum selben Objekttyp gehören. Daher liefert der hier beschriebene globale Registrierungsansatz eine Ergebnisliste, um die Bewertung der einzelnen Transformationen in einem umfangreicheren Kontext mit Hilfe von Heuristiken zu berechnen.

6 Objekt Modelle lernen

6.1 Modelle und Problemrepräsentation

Ohne weiteres Vorwissen muss davon ausgegangen werden, dass jede Punktgruppe eine Teilansicht für ein individuelles Objekt darstellt. Aus diesem Grund wird für jede Punktgruppe ein Modell erstellt und die entsprechende Punktgruppe zentriert darin gespeichert. Ein Modell $\mathcal{M}^{(i)}$ besteht aus einer Anzahl Punktwolken $\mathcal{V}_i^{(k)}$, welche verschiedene Teilansichten enthalten, sowie die korrespondierenden Positionen $p_i^{(k)}$, von denen aus die jeweilige Teilansicht wahrgenommen wurde. Im Folgenden soll es darum gehen, aus dieser initialen Modelldatenbank, gleiche Instanzen eines Objekttyps zu identifizieren und die Teilansichten zu kombinieren. Dabei spielt es keine Rolle, ob es sich um Teilansichten des selben Objektes handelt oder nur um Teilansichten einer gleichen Objektinstanz.

6.2 Hypothesen berechnen

6.2.1 Transformationen berechnen

Unter der Annahme, dass beide Teile eines Modellpaares $\mathcal{M}^i$, $\mathcal{M}^j$ $(k \neq l)$ zum selben Objekttyp gehören, berechnen wir die besten Transformationen $T_{n,\mathcal{M}^i,\mathcal{M}^j}$ für eine Paarung, die die Punkte von $\mathcal{M}^j$ in das Koordinatensystem von $\mathcal{M}^i$ überführen. ICP eignet sich hierfür jedoch nicht, da die initiale Ausrichtung der beiden Punktwolken zueinander nicht zwangsläufig im Konvergenzradius um das globale Maximum ist. Außerdem ist das globale Maximum zwar die beste Lösung bezüglich der vorhandenen Informationen, aber dies kann bei Teilansichten unzureichend sein. Die Transformationen werden mit Hilfe der globalen Registrierung, wie in Kapitel 5 beschrieben, berechnet. Dabei werden für jedes Modell die besten GOODSAC Merkmalskorrespondenzen, mit den zugehörigen Transformationen als Hypothesen für die Zusammengehörigkeit von zwei Modellen betrachtet. Gesucht sind die n besten Hypothesen für die Zusammengehörigkeit von $\mathcal{M}^i$ und $\mathcal{M}^j$. Eine Hypothese $H_{n,\mathcal{M}^i,\mathcal{M}^j}$ besteht aus einer Transformation $T_{n,\mathcal{M}^i,\mathcal{M}^j}$ und einer Bewertung $s_{n,\mathcal{M}^i,\mathcal{M}^j}$. Abbildung 6.2 zeigt den vollständigen Hypothesen Graphen.
Auf jede Hypothese im Graphen wird ICP angewendet, so konfiguriert, dass die Transformation nur minimal verändert wird. Dann werden alle bis auf eine Hypo-

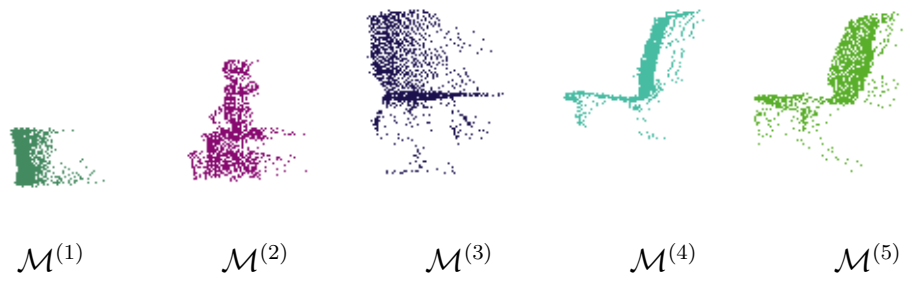

Abbildung 6.1: Beispiel einer initialen Modelldatenbank mit fünf Modellen

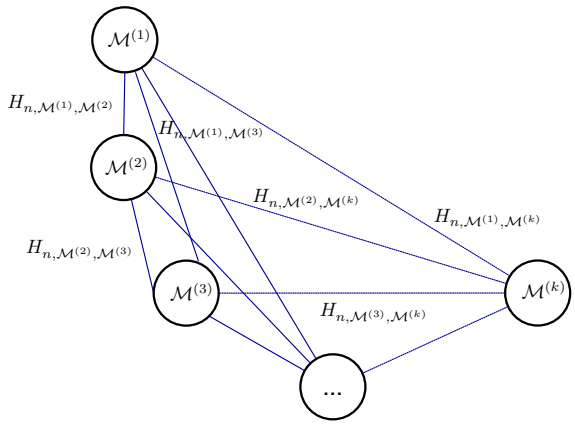

Abbildung 6.2: Hypothesen Graph für k Modelle, mit jeweils n Hypothesen für jedes Modellpaar

these, die zur gleichen Transformation konvergiert sind, aussortiert. Das Ergebnis mit der besten GOODSAC Bewertung wird behalten.
Um eine akzeptable Maximallaufzeit zu erhalten, werden nur die besten n Hypothesen für jedes Modell betrachtet. Die Ergebnisliste ist bereits nach der GOODSAC Qualität sortiert. So ist es relativ einfach möglich, die ersten n Ergebnisse in den Hypothesengraph einzufügen. Die Gefahr beim Beschneiden ist, dass zutreffende Hypothesen verloren gehen können. Ist n gut gewählt, wird die benötigte Rechenzeit auf die guten Hypothesen fokussiert, wie Experiment 7.4.1 belegt.
Die Wahl von n hängt von der Anzahl der Ergebnisse ab und damit von den gewählten Parametern für die Merkmalsanzahl und die Tiefenbilderanzahl. Ein guter Wert für die durchgeführten Experimente war $n = 200$.

6.2.2 Bewertungsfunktion

Die Bewertungsfunktion soll die Qualität von Hypothesen mit Hilfe von Heuristiken bestimmen. Eine hohe Bewertung soll dabei für die paarweise Konsistenz der beiden Modelle sprechen.

GOODSAC Qualität

Bereits beim Finden korrespondierender Merkmale mit GOODSAC wird eine Qualität für jedes Ergebnis berechnet. Je mehr Merkmale von Modell $\mathcal{M}^{(i)}$ auf Modell $\mathcal{M}^{(j)}$ mit der Transformation abbildbar sind, desto besser passen die betrachteten Teilstrukturen zueinander.
Die Anzahl der korrespondierenden Merkmale ist zum einen abhängig von der Überlappung der beiden Modelle. Je größer diese ist, desto mehr Korrespondenzen können gefunden werden. Zum anderen hängt sie von der Anzahl der gefundenen Merkmale auf den beiden Modellen ab. Je nach Objektstruktur kann die Anzahl der extrahierten Merkmale stark variieren. Aus den genannten Gründen fällt auch die Normierung der GOODSAC Qualität schwer. Trotzdem liefert die GOODSAC Qualität einen Hinweis auf die Konsistenz der Hypothese.

Überlappung

Eine weitere Heuristik ist die Überlappung von Modell $\mathcal{M}^{(i)}$ und Modell $\mathcal{M}^{(j)}$. Je größer die Überlappung der beiden Modelle, um so besser passen diese zusammen. Berechnen lässt sich die Überlappung, indem man für jeden Punkt aus Modell $\mathcal{M}^{(i)}$ eine Umkreissuche im kd-Baum von Modell $\mathcal{M}^{(j)}$ mit einem Radius r durchführt. Dabei werden alle Punkte mit nicht leerem Umkreis gezählt und so der überlappende Anteil berechnet. Analog werden die Punkte aus $\mathcal{M}^{(j)}$ mit nicht leerem Umkreis gezählt und auch hier der überlappende Anteil berechnet. Das Ergebnis ist dann die minimale berechnete Überlappung. Dabei ist der zu betrachtende Umkreis abhängig von der Datendichte des Modells. Bei einer Überlappung von nahezu 100 % aller Punkte reicht diese Heuristik zum Bestimmen kompatibler Modelle aus, doch wird in diesem Fall leider kaum zusätzliche Struktur gelernt.

Sichtlinienkompatibilität

Die wichtigste Heuristik basiert auf der Sichtlinieneigenschaft des Sensors. Zwischen Sensor und Punkten auf der Objektoberfläche liegen keine weiteren Punkte, da diese sonst mit dem Laser-Range-Finder gemessen worden wären. Dies muss auch für die Punkte eines kombinierten Modells $\mathcal{M}^{(1+2)}$ gelten, das von den Positionen $\mathbf{p^{(1)}}$ und $\mathbf{p^{(2)}}$ beobachtet wurde. Alle Punkte, die aus $\mathcal{M}^{(1)}$ stammen, dürfen von $\mathbf{p^{(1)}}$ aus nicht von Punkten aus $\mathcal{M}^{(2)}$ verdeckt werden. Analoges gilt für die Punkte aus $\mathcal{M}^{(2)}$ von $\mathbf{p^{(2)}}$ aus gesehen. Abbildung 6.3 illustriert diese Idee.

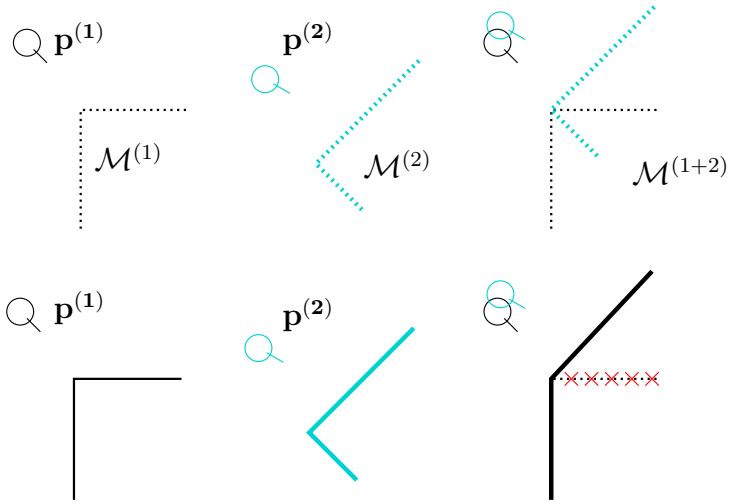

Abbildung 6.3: Beispielskizze für die Idee der Sichtlinieneigenschaft mit einer schlechten Hypothese. In der ersten Reihe sind die einzelnen Modelle $M^{(1)}$, $M^{(2)}$ und das kombinierte Modell $M^{(1+2)}$ abgebildet. In der unteren Reihe sind die entsprechenden Tiefenbilder abgebildet, jeweils aus den Observationspositionen $p^{(1)}$, $p^{(2)}$. Für das Tiefenbild des kombinierten Modells ist der rot markierte Bereich von $p^{(1)}$ aus nicht sichtbar, was für eine Inkompatibilität der Hypothese spricht.

$\mathcal{M}^{(1+2)}$ muss demnach auf einem Tiefenbild von $\mathbf{p^{(1)}}$ ungefähr so aussehen wie $\mathcal{M}^{(1)}$ und von $\mathbf{p^{(2)}}$ aus gesehen, muss $\mathcal{M}^{(1+2)}$ ungefähr so aussehen wie $\mathcal{M}^{(2)}$. Abbildung 6.4 zeigt dies am Beispiel von zwei Punktwolken eines Stuhltyps. Ohne Sensorrauschen wäre die Ähnlichkeit der Tiefenbilder von $\mathcal{M}^{(1+2)}$ zu denen von $\mathcal{M}^{(1)}$ und $\mathcal{M}^{(2)}$ noch größer.

Da nur initiale Modelle aus einer Objektansicht bestehen, erweitern wir den Ansatz für Modelle mit beliebig vielen Ansichten. Ein kombiniertes Modell aus $\mathcal{M}^{(i)}$ und $\mathcal{M}^{(j)}$ mit der Transformation $T_{n,\mathcal{M}^i,\mathcal{M}^j}$ besitzt $m+n$ Punktwolken aus $\mathcal{V}_m^{(i)}$ und $\mathcal{V}_n^{(j)}$, mit den korrespondierenden Observationspositionen $p_m^{(i)}$ und $p_n^{(j)}$.
Sei $\mathcal{P}_{i,m}^k$ der i-te Punkt aus $\mathcal{V}_m^{(i)}$ und $\mathcal{V}_m^{(i)}$ enthalte $q_{\mathcal{V}_m^{(i)}}$ Punkte. Weiterhin sei $\mathcal{I}$ die Indikatorfunktion. Dann hat $\mathcal{V}_r^{(i+j)}$ mit Gleichung 6.1 y_r verdeckte Punkte.

$$y_r = \sum_{i=1}^{q_{\mathcal{V}_m^{(i+j)}}} \mathcal{I}_{\mathcal{P}_{i,m}^{i+j} verdeckt} \tag{6.1}$$

Dann berechnen wir die Heuristik h für $\mathcal{M}^{(i+j)}$ wie folgt:

$$h(\mathcal{M}^{(i+j)}) = 1 - \frac{\sum_{i=1}^{m+n} y_i}{\sum_{i=1}^{m+n} q_{\mathcal{V}_i^{(i+j)}}} \tag{6.2}$$

Abbildung 6.4: Beispiel für Sichtlinieneigenschaft mit Punktwolken und den korrespondierenden Tiefenbildern. Die erste Reihe zeigt die Punktwolken für die Modelle $\mathcal{M}^{(1)}$ und $\mathcal{M}^{(2)}$ aus ihren Observationspositionen. Die zweite Reihe zeigt die dazugehörigen Tiefenbilder aus jeweils der selben Perspektive. In der unteren Reihe sind links die kombinierte Punktwolke aus einem alternativen Blickwinkel und daneben die Tiefenbilder für $\mathcal{M}^{(1+2)}$ aus den Blickpositionen $\mathbf{p^{(1)}}$ und $\mathbf{p^{(2)}}$

Damit lässt sich der Anteil an verdeckten Punkten berechnen und jeder verdeckte Punkt soll die Bewertung der Heuristik entsprechend verkleinern. Für die Berechnung der Sichtbarkeitseigenschaft von Punkten in Gleichung 6.1 ergeben sich zwei Möglichkeiten. Zum einen kann diese Heuristik mit Tiefenbildern berechnet werden. Zum anderen eignet sich auch der in Kapitel 4.3.1 besprochene Raytracer. Der Vorteil von Tiefenbildern wie in Abbildung 6.4 ist, dass die Kontur der Objekte gut repräsentiert ist. Der Raytracer hingegen liefert ein sehr präzises Ergebnis auf Punktebene, da direkt auf der Punktwolke gearbeitet wird und nicht auf einer 2D Projektion. Aus diesem Grund wird die Sichtlinieneigenschaft auf beiden Repräsentationen berechnet.

Gewichtete Kombination der Heuristiken

Die Bewertung für eine Transformation wird aus der gewichteten Summe der einzelnen Heuristiken berechnet. Die vorgestellten Heuristiken sind h_1 die GOODSAC

Qualität, h_2 die Überlappung, h_3 und h_4 die Sichtlinienkompatibilität einmal mit Tiefenbildern und mit dem Raytracer. Mit den ω_i als Gewichten, ergibt sich so folgende Bewertungsfunktion:

$$\begin{aligned} s_{n,\mathcal{M}^{(i)},\mathcal{M}^{(j)}} &= \omega_1 \cdot h_1(\mathcal{M}^{(i+j)}) + \omega_2 \cdot h_2(\mathcal{M}^{(i+j)}) \\ &+\omega_3 \cdot h_3(\mathcal{M}^{(i+j)}) + \omega_4 \cdot h_4(\mathcal{M}^{(i+j)}) \end{aligned} \tag{6.3}$$

$$\sum_{i=1}^{n} \omega_i = 1 \tag{6.4}$$

In der Implementierung wurden folgende Gewichte benutzt:

$$\omega_1 = 0.1,\ \omega_2 = 0.3,\ \omega_3 = 0.2, \omega_4 = 0.4$$

Damit lassen sich nun die Bewertungen für alle Hypothesen $H_{n,\mathcal{M}^i,\mathcal{M}^j}$, aller Modellpaarungen berechnen. Dies ist der rechenintensivste Schritt des Ansatzes, da alle Hypothesen für den vollständigen Graphen berechnet werden müssen.

6.3 Ähnlichkeitsmatrix

Die Ähnlichkeitsmatrix ist eine Repräsentation für die besten Hypothesen, der jeweiligen Modellpaarungen i, j aus dem Hypothesengraph. Mit Hilfe dieser Matrix werden die Modelle algorithmisch bezüglich ihrer Objektzugehörigkeit gruppiert. Zur Erinnerung, eine Hypothese besteht aus einer Transformation T und einer Bewertung s. Die n-te Hypothese ist also

$$H_{n,\mathcal{M}^i,\mathcal{M}^j} = \langle T_{n,\mathcal{M}^i,\mathcal{M}^j}, s_{n,\mathcal{M}^i,\mathcal{M}^j} \rangle. \tag{6.5}$$

Die beste Bewertung ist dann

$$s_{max,\mathcal{M}^i,\mathcal{M}^j} = \text{argmax}(s_{n,\mathcal{M}^i,\mathcal{M}^j}) \tag{6.6}$$

und die beste Hypothese

$$H_{max,\mathcal{M}^i,\mathcal{M}^j} = \langle T_{max,\mathcal{M}^i,\mathcal{M}^j}, s_{max,\mathcal{M}^i,\mathcal{M}^j} \rangle. \tag{6.7}$$

Die besten Hypothesen jeder Modellpaarung sind in der Ähnlichkeitsmatrix A repräsentiert. Dabei sind die Einträge wie folgt definiert:

$$A_{ij} = s_{max,\mathcal{M}^i,\mathcal{M}^j}. \tag{6.8}$$

Abbildung 6.5 zeigt ein Beispiel für eine Ähnlichkeitsmatrix. Mit Hilfe der Ähnlichkeitsmatrix kann die tatsächliche Anzahl von Objekttypen in der Modelldatenbank bestimmt werden und welche Modellpaare eines Objekttyps zusammengeführt werden können. Die Diagonale einer Ähnlichkeitsmatrix besteht per Definition nur aus weißen Feldern, da keine Ähnlichkeitsvergleiche mit einem Modell $\mathcal{M}^{(i)}$ zu sich selbst berechnet werden.

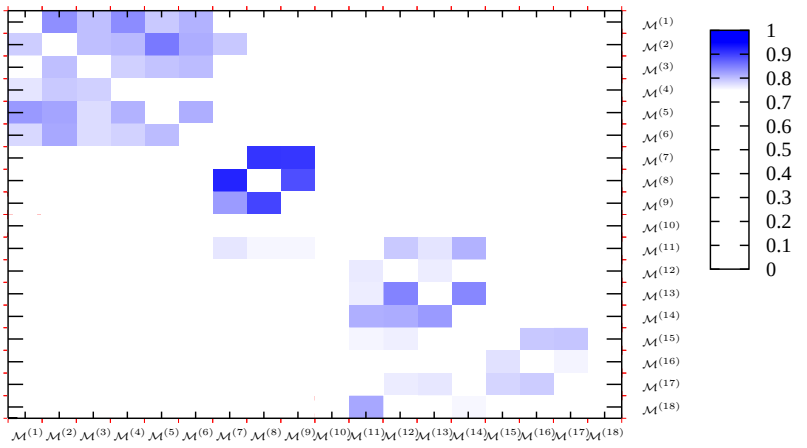

Abbildung 6.5: Beispiel einer Ähnlichkeitsmatrix für eine Modelldatenbank mit 18 Einträgen. Dabei ist in jeder Zelle die Bewertung der besten Hypothese $s_{max,\mathcal{M}^i,\mathcal{M}^j}$ eingetragen. Die Objekte sind bezüglich ihrer Zusammengehörigkeit in Reihe angeordnet und nur die Bewertungen $> 0{,}76$ sind sichtbar. Mit dieser Ansicht lassen sich die Modelle gut Gruppen zuweisen, die jeweils einen Objekttyp repräsentieren.

6.3.1 Ähnlichkeitsmatrix ist nicht symmetrisch

Intuitiv müsste eine Ähnlichkeitsmatrix symmetrisch sein, da die Ähnlichkeit von $\mathcal{M}^{(i)}$ zu $\mathcal{M}^{(j)}$ exakt der Ähnlichkeit $\mathcal{M}^{(j)}$ zu $\mathcal{M}^{(i)}$ entsprechen sollte. Dies ist aber nicht zwingend der Fall, beispielsweise kann ein Modellvergleich $\mathcal{M}^{(i)}$, $\mathcal{M}^{(j)}$ eine zusätzliche Hypothese mit besserer Bewertung liefern, als der inverse Vergleich $\mathcal{M}^{(j)}$, $\mathcal{M}^{(i)}$. Dies hängt damit zusammen, dass die Berechnung der globalen Registrierung von $\mathcal{M}^{(i)}$ zu $\mathcal{M}^{(j)}$ nur die Merkmale von einem Tiefenbild aus der Observationsposition für Modell k, gegen die Merkmale von $t^{(l)}$ Tiefenbildern für Modell l betrachtet. Zusammen mit der Tatsache, dass die $t^{(k)}$ Tiefenbilder zu den korrespondierenden Tiefenbildern $t^{(l)}$ durch die verschiedenen Observationspositionen $p^{(k)}, p^{(l)}$ einen zueinander verschobenen Beobachtungswinkel haben, werden je nach Betrachungsrichtung zwischen $\mathcal{M}^{(i)}$ und $\mathcal{M}^{(j)}$ verschiedene Merkmale verglichen. Daher ist es möglich, dass im Vergleich $\mathcal{M}^{(j)}$ zu $\mathcal{M}^{(i)}$ eine bessere Hypothese gefunden wird, als in der Umkehrrichtung. Durch Erhöhung der Tiefenbilderanzahl t kann die Symmetrie der Ähnlichkeitsmatrix verbessert werden, da die Positionen der korrespondierenden Tiefenbilder dichter zusammen liegen. Da dies aber mit zusätzlichem Rechenaufwand verbunden ist und die Ergebnisse für die gelernten Modelle vergleichbar sind, haben wir darauf verzichtet. Außerdem handelt es sich um bereits vorhandene Information, die nur redundant berechnet würde. Ohne zusätzlichen Rechenaufwand könnte man die beste Hypothese zwischen zwei Modellen auswählen und die Inverse Transformation für die andere Vergleichsrichtung einfügen. Darauf wird aber im Folgenden verzichtet, da gerade falsche Hypothesen

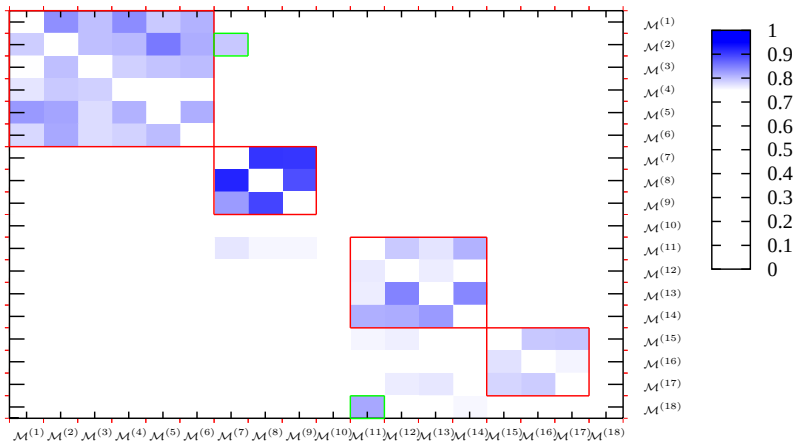

Abbildung 6.6: Ähnlichkeitsmatrix in der die zusammengehörigen Modellgruppen durch rote Rechtecke gekennzeichnet sind. Ausreißer mit hoher Bewertung sind in grün eingezeichnet.

mit hoher Bewertung, sehr häufig nur in einer Vergleichsrichtung gefunden werden. Damit ist eine bessere Abgrenzung der Objektgruppen zu Ausreißern möglich.

6.4 Zusammenführen von Modellen

Gesucht ist eine Einteilung der Modelle in Gruppen, so dass alle Modelle, die zu einem Objekttyp gehören, auch in der gleichen Gruppe sind. Da die Modelle in Abbildung 6.5 entsprechend ihrer Objektzugehörigkeit sortiert sind, fällt eine visuelle Einteilung durch einen Menschen nicht schwer. Abbildung 6.6 zeigt die tatsächlichen, zusammengehörenden Modelle mit Hilfe von roten Rechtecken. Intuitiv könnte man zusammengehörige Modelle mit einem Greedyansatz zusammenführen, also Iterativ die Modelle mit der besten Hypothesenbewertung zusammenführen. Da es aber gut bewertete Hypothesen zwischen einzelnen Modellen verschiedener Objekttypen geben kann (siehe grün markierte Zellen in Abbildung 6.6) eignet sich der globale Greedyansatz nur bedingt.
Die Ausreißer in der Ähnlichkeitsmatrix führen zu inkonsistenten Modellen. Da die Ähnlichkeitsmatrix ein Maß für die paarweise Konsistenz ist, gilt es eine Gruppe von Modellen mit maximaler paarweiser Konsistenz zu finden. Ausreißer kennzeichnen sich dadurch, dass es in einer Gruppe mehr Modelle gibt, die gegen einen Ausreißer sprechen, als dafür. Aus diesem Grund wird die Ähnlichkeitsmatrix mit dem Single-Cluster Spectral Graph Partitioning (SCGP) Algorithmus von Olson *et al.* [14] partitioniert. Ziel ist es, die beste Partition aus der Ähnlichkeitsmatrix zu berechnen, unter Ausschluss von Ausreißern.

6.4.1 Iteratives Zusammenführen von Modellen

Aus der berechneten Ähnlichkeitsmatrix soll nun mit Hilfe von SCGP der dominante Cluster berechnet werden. Die Modelle, die in diesem dominanten Cluster liegen, sollen iterativ zusammengeführt werden. Ist der berechnete Cluster vollständig zusammengeführt, wird ein weiterer Cluster berechnet und iterativ zusammengeführt. Dies wird weitergeführt bis eine der Abbruchbedingungen erfüllt wird. Die resultierende Modelldatenbank repräsentiert die aus den Teilansichten gelernten Modelle. Für die Anwendung von SCGP wird eine symmetrische Adjazenzmatrix A als Eingabe benötigt. Die Ähnlichkeitsmatrix $A_{\mathcal{M}}$ eignet sich als Eingabe für A. Da die Ähnlichkeitsmatrix nur nahezu symmetrisch ist, erzeugen wir die Symmetrie durch :

$$A_{i,j} = A_{\mathcal{M}\,i,j} \cdot A_{\mathcal{M}\,j,i} \tag{6.9}$$

Die Eigenwerte für SCGP werden mit dem *Rayleigh Quotienten Iteration* Algorithmus berechnet. Als Ergebnis erhalten wir einen Indikatorvektor $\mathbf{v}_i(t)$ und berechnen damit die zum Cluster gehörenden Modelle. Innerhalb dieses Clusters werden Modelle mit einem Greedy Ansatz zusammengeführt. Ein Zusammenführschritt besteht dabei aus folgenden Teilschritten:

- Bestimme die Hypothese $H_{max,\mathcal{M}^i,\mathcal{M}^j}$ für den Eintrag mit der besten Bewertung in dem berechneten Cluster, so dass $\text{argmax}(A_{ij}) = s_{max,\mathcal{M}^i,\mathcal{M}^j}$ mit $i, j \in \mathbf{v}(t)$.

- Transformiere Punktwolken $\mathcal{V}_k^{(j)}$ und Observationspositionen $p_k^{(j)}$ für Modell $\mathcal{M}^{(j)}$ in Koordinatensystem von $\mathcal{M}^{(i)}$

$$p_l^{(j')} = p_l^{(j)} \cdot T_{max,\mathcal{M}^i,\mathcal{M}^j} \tag{6.10}$$
$$\mathcal{V}_l^{(j')} = \mathcal{V}_l^{(j)} \cdot T_{max,\mathcal{M}^i,\mathcal{M}^j} \tag{6.11}$$

- Einfügen von $p_l^{(j')}$ und $\mathcal{V}_l^{(j')}$ in $\mathcal{M}^{(i')}$.

$$p_{i+j}^{(i')} = p_k^{(i)} \cup p_l^{(j')} \tag{6.12}$$
$$\mathcal{V}_{i+j}^{(i')} = \mathcal{V}_k^{(i)} \cup \mathcal{V}_l^{(j')} \tag{6.13}$$

- Neuberechnung der Tiefenbilderkollektion $\mathcal{D}_t^{(i')}$

- Ersetze Modell $\mathcal{M}^{(i)}$ durch $\mathcal{M}^{(i')}$ und entferne Modell $\mathcal{M}^{(j)}$ aus Modelldatenbank.

- Berechne Hypothesen für $\mathcal{M}^{(i')}$ neu.

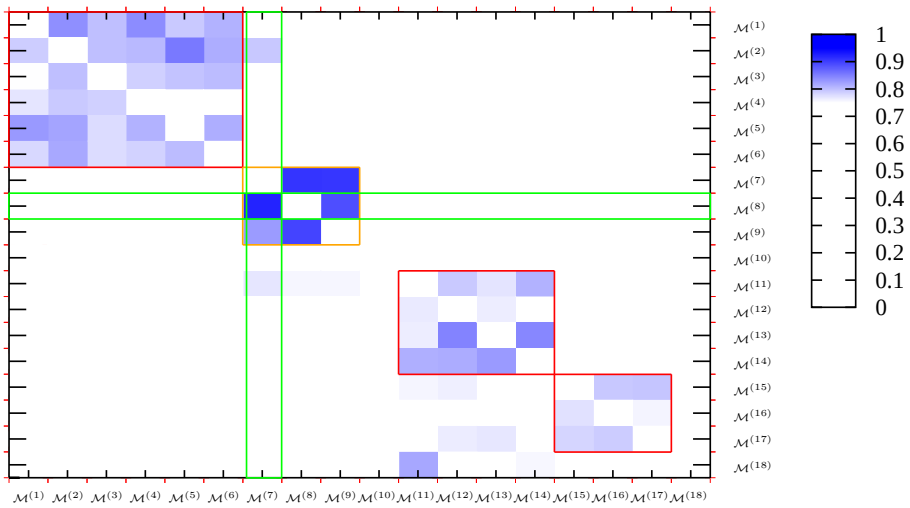

Abbildung 6.7: Ähnlichkeitsmatrix in der die dominante Modellgruppe durch ein orangenes Rechteck gekennzeichnet ist. Reihe und Spalte der besten Bewertung sind grün markiert. Die beteiligten Modelle sind $\mathcal{M}^{(8)}$ und $\mathcal{M}^{(7)}$.

Auf das Beispiel in Abbildung 6.7 übertragen, ist der mit SCGP berechnete Cluster in orange eingezeichnet und die Zelle mit der höchsten Bewertung durch grüne Balken gekennzeichnet. Die beteiligten Modelle sind $\mathcal{M}^{(8)}$, $\mathcal{M}^{(7)}$. Die Punktwolken $\mathcal{V}_k^{(7)}$ und Observationspositionen $p_k^{(7)}$ von $\mathcal{M}^{(7)}$ werden nun mit der Transformation $T_{max,\mathcal{M}^8,\mathcal{M}^7}$ in das Koordinatensystem von $\mathcal{M}^{(8)}$ gebracht und zu $\mathcal{M}^{(8)}$ hinzugefügt. Danach wird das Modell $\mathcal{M}^{(7)}$ gelöscht und die resultierende Matrix wird in Abbildung 6.8 gezeigt. In der nächsten Iteration werden $\mathcal{M}^{(7:8)}$ und $\mathcal{M}^{(9)}$ zusammengeführt.
Auf diese Art werden iterativ alle Modelle eines Clusters zusammengeführt. Danach wird der nächste Cluster berechnet und die Modelle dieses Clusters auch zusammengeführt, siehe Abbildung 6.9, bis nur noch Bewertungen für Hypothesen in der Ähnlichkeitsmatrix enthalten sind, die unter einem bestimmten Grenzwert liegen.
Das Neuberechnen der Hypothesen für ein zusammengeführtes Modell sorgt dafür, dass eventuell durch die kombinierte Information neue Hypothesen gefunden werden. Da der Hypothesengraph aus Effizienzgründen beschnitten wird, sind nach dem Zusammenführen auch wieder mehr „Plätze" für Hypothesen bezüglich anderer Modelle frei.

6.4.2 Konsistenz von Modellen

In jedem Iterationsschritt auf der Ähnlichkeitsmatrix werden Modelle kombiniert und damit auch kleine Inkonsistenzen hinzugefügt. Die addierten Inkonsistenzen in einem Modell, mindern auch die Bewertung der Sichtlinienheuristik im Bezug auf

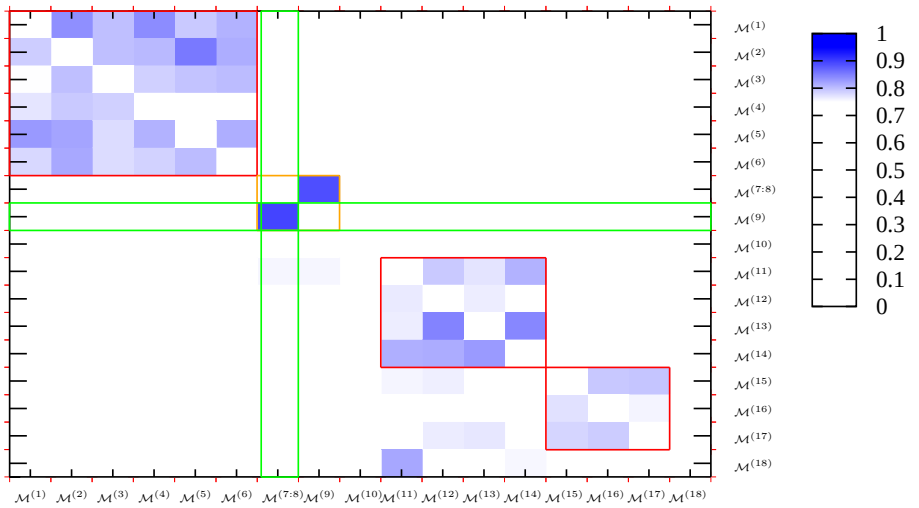

Abbildung 6.8: Ähnlichkeitsmatrix nach Zusammenführen von $\mathcal{M}^{(8)}$ und $\mathcal{M}^{(7)}$. Die Reihe und Spalte für $\mathcal{M}^{(7)}$ sind nicht mehr in der Matrix enthalten, daher sind in dem Modellcluster nur noch 4 Felder. Als nächstes werden die beiden verbleibenden Modelle mit der besten Hypothese (grünes Rechteck) zusammengeführt.

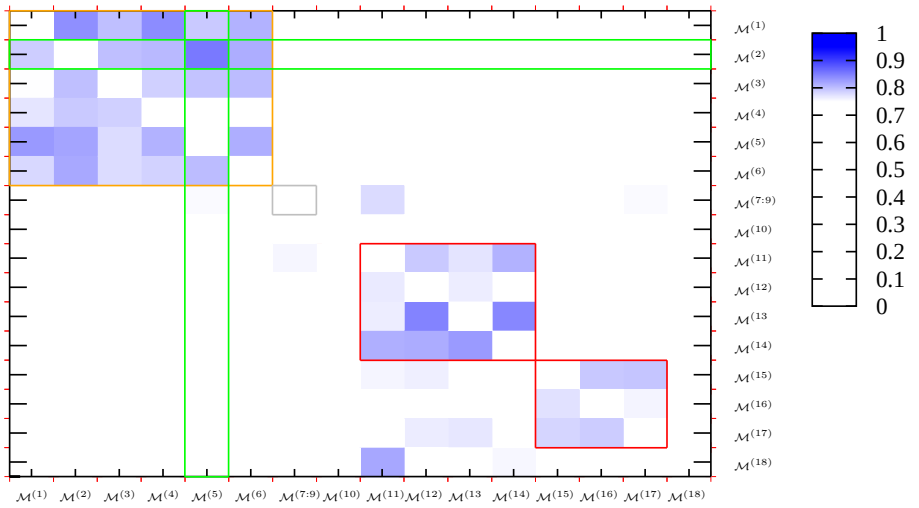

Abbildung 6.9: Ähnlichkeitsmatrix nach Zusammenführen von $\mathcal{M}^{(7:8)}$ und $\mathcal{M}^{(9)}$. Damit ist dieser Modellcluster (grau) vollständig zusammengeführt. Als nächstes wird ein neuer Cluster (orange) berechnet, in diesem Fall für die Pflanzen, und weiter iterativ zusammengeführt.

andere Modelle. Das kann dazu führen, dass ein Modell ein Inkonsistenzstadium erreicht, in dem keine weitere Teilansicht dazu gelernt werden kann. Aus diesem Grund ist es zwingend erforderlich die Modellkonsistenz nach jedem Lernschritt zu optimieren.
Inkonsistenzen können durch kleine Anordnungsfehler der Teilansichten zueinander entstehen oder einfach durch Fehlmessungen und Sensorrauschen. Je mehr überlappende Modellansichten vorhanden sind, desto genauer ist eine konsistente Anordnung möglich. Um Anordnungsfehler zu verringern, werden die Punktwolken der Teilansichten paarweise mit Hilfe von ICP aneinander ausgerichtet.
Die Reduktion von Fehlmessungen ist hingegen schwieriger und kann nur in Bereichen des Modells erfolgen, in denen die kombinierte Information von mehreren Teilansichten die Identifikation von Fehlmessungen ermöglicht. Die Fehlmessungen lassen sich einteilen in Messungen, die vor oder hinter der eigentlichen Oberfläche liegen.
Fehlmessungen, die vor der Objektoberfläche liegen, verdecken Oberflächenpunkte einer anderen Teilansicht und haben keine oder wenig Überlappung mit einer anderen Teilansicht. Die Berechnung dieser Eigenschaften wurde bereits im Abschnitt der Bewertungsfunktion 6.2.2 erläutert.
Die Konsistenz ω_P eines Punktes P wird definiert als:

$$\#H = \text{Anzahl durch P verdeckter Punkte} \tag{6.14}$$

$$\#O = \text{Anzahl überlappender Punkte für P} \tag{6.15}$$

$$\omega_P = \begin{cases} \#H & \text{, falls es keine überlappenden Punkte für P gibt.} \\ \frac{\#H}{\#O} & \text{, sonst.} \end{cases} \tag{6.16}$$

Alle Punkte für die $\omega \geq 1$ werden aus der kombinierten Punktwolke entfernt, jedoch nicht aus den Punktwolken der Teilansichten $\mathcal{P}_M^i$. Das iterative Löschen von Punktens könnte sonst zu schlechten Ergebnissen führen, in denen kaum noch Punkte enthalten sind. Auf diese Art müssen inkonsistente Punkte in jeder Iteration zwar wieder aus dem kombinierten Modell gelöscht werden, aber dafür kann die Konsistenzinformation zu einem späteren Zeitpunkt noch dazu führen, dass bestimmte Punkte doch noch in das Modell übernommen werden.
Enthält ein Modell weniger als 50% konsistente Punkte einer Teilansicht, wird diese Teilansicht aus dem Modell entfernt und wieder als eigenständiges Modell in die Modelldatenbank eingefügt. Dieser Fall tritt bei einer Offline Anwendung des Algorithmuses eigentlich nicht auf, da alle Teilansichten bekannt sind und inkonsistente Teilansichten nicht in eine Graphpartition eingefügt werden.

7 Experimente

In diesem Kapitel werden einige Experimente mit dem implementierten System vorgestellt, um die Robustheit des Ansatzes zu testen. Dabei werden auch Experimente mit Objekten, die eine ähnliche Struktur besitzen, sowie mit symmetrischen Objekten aufgeführt.

7.1 Scanpaar

Im ersten Experiment werden wie in Kapitel 4.3 beschrieben, die Unterschiede für zwei Scans berechnet und mit Region Growing gruppiert. Abbildung 7.1 zeigt die einzelnen Objektansichten. Als Objekte für dieses Experiment dienen drei Holzstühle, zwei Mülleimer und ein Bürostuhl. Bilder der verwendeten Objekttypen sind in Abbildung 7.3 zu sehen. Die beiden Mülleimer sind Instanzen eines punktsymmetrischen Objektes. Die Holz- und Bürostühle sind ähnliche Objekte, deren Sitzflächen die gleiche Struktur haben. Obwohl die beiden Stuhltypen eine ähnlich geformte Sitzfläche haben, sehen sie aus vielen Betrachtungsrichtungen unterschiedlich aus. Hier müssen mehrere Teilansichten des Holzstuhles zusammengeführt werden, damit die Inkonsistenz zum Bürostuhl erkennbar wird.
Abbildung 7.2 zeigt die initiale Modelldatenbank, die korrespondierende initiale Ähnlichkeitsmatrix ist in Abbildung 7.4 zu sehen. Die zusammen gehörenden Modelle sind die Mülleimer $\{\mathcal{M}^{(1)},\mathcal{M}^{(2)}\}$ und die Stühle $\{\mathcal{M}^{(3)}, \mathcal{M}^{(4)}, \mathcal{M}^{(5)}\}$. Modelle $\{\mathcal{M}^{(1)}$ und $\mathcal{M}^{(2)}\}$ werden in der ersten Iteration zusammengeführt. Die folgenden Iterationen fügen dann $\{\mathcal{M}^{(4)}, \mathcal{M}^{(3)}\}$ und $\{\mathcal{M}^{(5)},\mathcal{M}^{(4)}\}$ zusammen, siehe Abbildung 7.5.
Während des Zusammenführens der Modelle in der Ähnlichkeitsmatrix, wird für das jeweils „neu" gelernte Modell nur die Zeile neu berechnet und nicht die vollständige Ähnlichkeitsmatrix, um Rechenzeit zu sparen. Daher bleiben die hohen Bewertungen der Ausreißer in den Spalten, für nicht neu berechnete Modelle erhalten. Durch die gewählte Methode, mit der die Ähnlichkeitsmatrix in eine symmetrische Matrix umgewandelt wird, reduzieren sich solche Bewertungen aber. Ein Beispiel hierfür ist in Abbildung 7.5 die finale Ähnlichkeitsmatrix. Dort ist in der Zelle $\mathcal{M}^{(6)},\mathcal{M}^{(5)}$ immer noch eine hohe Bewertung, da die Zeile von $\mathcal{M}^{(6)}$ in keinem Iterationsschritt neu berechnet wurde. Durch die Multiplikation von Zelle $\mathcal{M}^{(6)},\mathcal{M}^{(5)}$ mit der neu berechneten Zelle $\mathcal{M}^{(5)},\mathcal{M}^{(6)}$ sinkt die Bewertung.
Interessant an den Ergebnissen ist das resultierende Modell $\mathcal{M}^{(1)}$, da es aus zwei

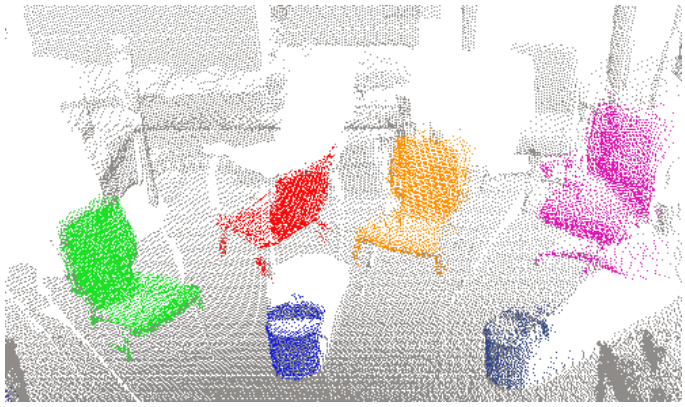

Abbildung 7.1: Eingefärbt sind die extrahierten, nicht stationäre Modelle aus zwei 3D-Scans. Als Modelle wurden zwei Instanzen des Objekttyps Mülleimer, drei Instanzen des Objekttyps Stuhl und eine Instanz des Objekttyps Bürostuhl gefunden.

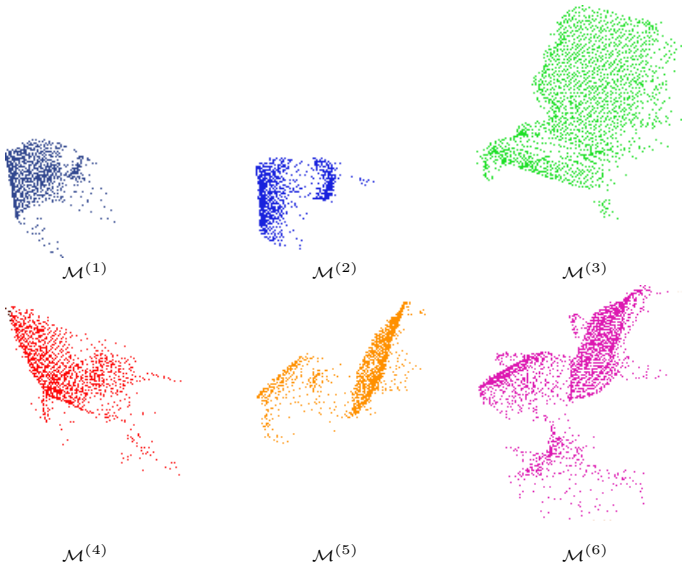

Abbildung 7.2: Modelldatenbank aus den in Abb.7.1 extrahierten Punktgruppen. Der Teilansichtscharakter der initialen Modelle ist in dieser Darstellung gut zu erkennen.

Abbildung 7.3: Bilder der Objektinstanzen, die in diesem Experiment verwendet wurden. Im Ersten Bild ist der Mülleimer, im Zweiten Bild der Holzstuhl und im Dritten Bild ist der Bürostuhl zu sehen.

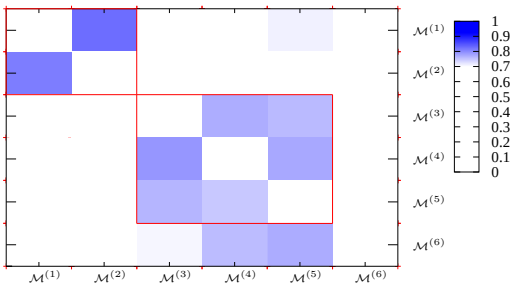

Abbildung 7.4: Initiale Ähnlichkeitsmatrix für die extrahierten Modelle. Die zu Zeilen und Spalten gehörenden Modelle $\mathcal{M}^{(x)}$ sind in Abb. 7.2 ersichtlich. Zusammengehörige Modelle sind $\mathcal{M}^{(1)}$,$\mathcal{M}^{(2)}$ und $\mathcal{M}^{(3)}$,$\mathcal{M}^{(4)}$,$\mathcal{M}^{(5)}$

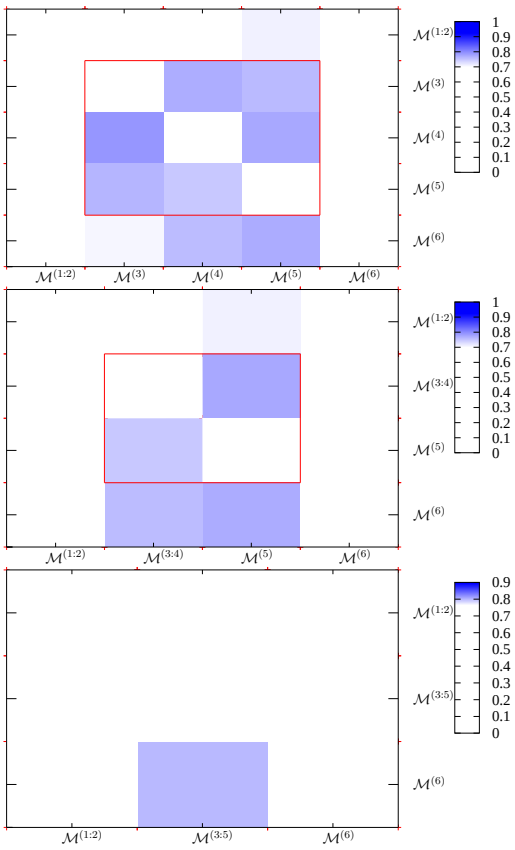

Abbildung 7.5: Ähnlichkeitsmatrizen der Iterationen zwei, drei und der finalen Iteration vier. In diesen Schritten wird der letzte Cluster (rotes Rechteck), bestehend aus den Modellen $\mathcal{M}^{(3)}$, $\mathcal{M}^{(4)}$ und $\mathcal{M}^{(5)}$, zusammengeführt. In der letzten Ähnlichkeitsmatrix ist kein konsistenter Cluster mehr zu erkennen. Nur noch eine Hypothese für $\mathcal{M}^{(6)}$ und $\mathcal{M}^{(5)}$, die aber nicht durch die neu berechnete Hypothese $\mathcal{M}^{(5)}$, $\mathcal{M}^{(6)}$ bestätigt wird.

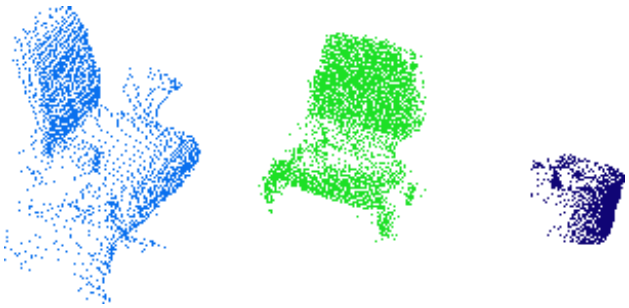

Abbildung 7.6: Ergebnis des Experiments nach vier Iterationen: aus den Teilansichten gelernte Modelle nicht stationärer Objekte. Der Bürostuhl besteht nur aus der einen verfügbaren Teilansicht, der Holzstuhl besteht aus 3 Teilansichten und der Mülleimer aus zwei Ansichten.

symmetrischen Teilansichten besteht. Das Problem mit symmetrischen Objekten ist, dass sie aus jeder Beobachtungsposition gleich aussehen und eine Zuordnung von zwei Teilansichten daher immer eine maximale Überlappung besitzt. Da es keine strukturellen Informationen gibt, die eine eindeutige Positionierung einer Teilansicht ermöglicht, verhält sich der Algorithmus so, als würde er immer die gleiche Teilansicht des Objektes sehen. Abgesehen von der Symmetrieproblematik ist das Ergebnis allerdings als sehr gut anzusehen, da die beiden Stuhltypen nicht zu einem inkonsistenten Modell vereinigt worden sind.

7.2 Erweiterte Modelldatenbank

Da die Anzahl der nicht stationären Objekte, die in einem 3D-Scanpaar angeordnet werden können, limitiert ist, werden in diesem Experiment die Modelle aus einer Vielzahl von 3D-Scans kombiniert. Auf eine Darstellung der einzelnen Scans, mit den extrahierten Modellen wird hier aus Platzgründen verzichtet. Die vollständige Modelldatenbank ist in Abbildung 7.7 zu sehen. Die Objektinstanzen der Modelle sind in den Abbildungen 7.3 und 7.8 aufgeführt. Damit die Abbildungen für das Experiment übersichtlicher erscheinen, wurden die Modelle bezüglich ihrer Objektzugehörigkeit sortiert. Speziell die Abbildung der Ähnlichkeitsmatrix ist dadurch optisch aussagekräftiger. Die Modelldatenbank besteht aus sechs Teilansichten einer Pflanze $\{\mathcal{M}^{(1)}, \mathcal{M}^{(2)}, \mathcal{M}^{(3)}, \mathcal{M}^{(4)}, \mathcal{M}^{(5)}\mathcal{M}^{(6)}\}$, drei Teilansichten des Objekttyps Mülleimer $\{\mathcal{M}^{(7)}, \mathcal{M}^{(8)}\mathcal{M}^{(9)}\}$, einer Teilansicht eines Pioneer Roboters $\{\mathcal{M}^{(10)}\}$, vier Teilansichten des Objekttyps Holzstuhl $\{\mathcal{M}^{(11)}, \mathcal{M}^{(12)}, \mathcal{M}^{(13)}, \mathcal{M}^{(14)}\}$, drei Teilansichten des Objekttyps Monitor und einer Teilansicht für einen Bürostuhl.
Die Modelle für die Pflanze stammen von sechs verschiedenen Scans der selben Pflanze, da Pflanzen eine eher einmalige Struktur besitzen. Die Modelle der Mülleimer, Holzstühle und der Monitore stammen dagegen von verschiedenen Instanzen des jeweiligen Objekttyps. Der Roboter und der Bürostuhl sind als mögliche Kandidaten für das Zusammenführen falscher Modelle gedacht.
Die initiale Ähnlichkeitsmatrix wird in Abbildung 7.9 gezeigt. Die Gruppe von Modellen, die zu einem Objekttyp gehören, sind mit roten Rechtecken gekennzeichnet. Idealerweise sollte der Algorithmus nur jeweils die Modelle in den roten Rechtecken zusammenführen. Interessant an dieser Ähnlichkeitsmatrix ist auch, dass die Bewertungen in den roten Rechtecken je nach Objekttyp unterschiedlich deutlich sind. Da die Bewertung für eine Hypothese von der Überlappung und von der Anzahl übereinstimmender Merkmale (GOODSAC Bewertung) abhängt, ist anzunehmen, dass für Objekttypen mit schwacher Struktur mehr Teilansichten mit stärkerer Überlappung notwendig sind, um ein vollständiges Modell zu lernen. Aus diesem Grund sind die Bewertungen für die Pflanze deutlich ausgeprägter als die des Monitors, da der Monitor weniger eindeutige Struktur aufweist als die Pflanze.
Abbildung 7.10 zeigt die finale Ähnlichkeitsmatrix nach 11 Iterationen und Abbil-

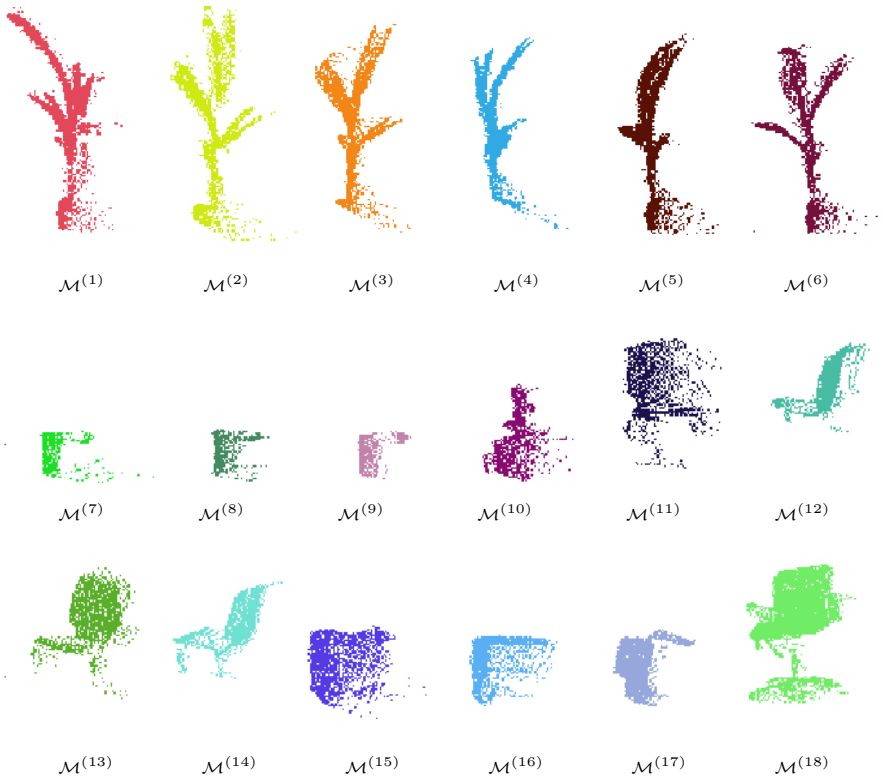

Abbildung 7.7: Initiale Modelldatenbank bestehend aus sechs Modelle einer Pflanze, drei Modellen des Objekttyps Mülleimer, ein Modell eines Pioneer Roboters, vier Modellen des Objekttyps Holzstuhl, drei Modellen des Objekttyps Monitor und ein Modell eines Bürostuhls.

Abbildung 7.8: Bilder der zusätzlichen Objektinstanzen, die neben den Objekten aus Abbildung 7.3, in diesem Experiment verwendet wurden. Im Ersten Bild ist die Pflanze, im Zweiten ist der Monitor und im Dritten Bild ist der Pioneer Roboter zu sehen

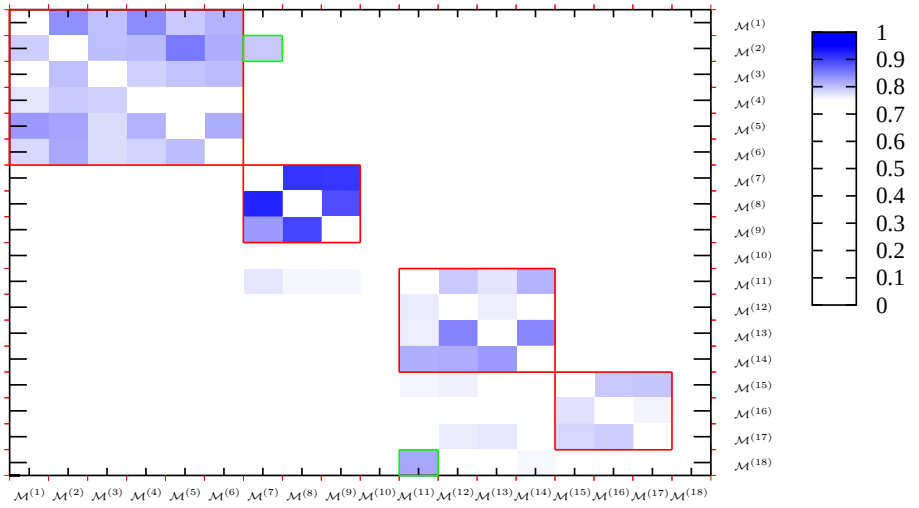

Abbildung 7.9: Initiale Ähnlichkeitsmatrix mit 18 Modellen. Rote Rechtecke kennzeichnen die zum selben Objekttyp gehörenden Modelle.

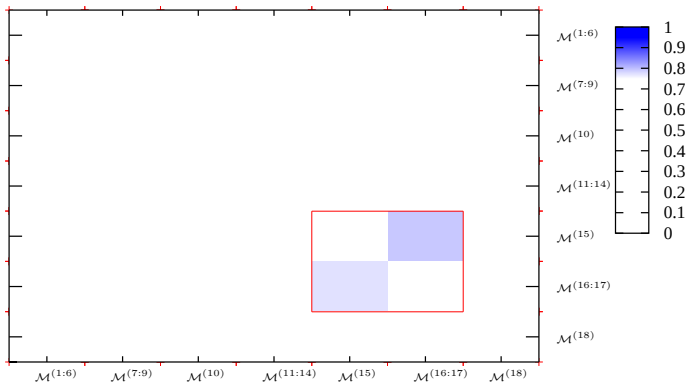

Abbildung 7.10: Finale Ähnlichkeitsmatrix nach 11 Iterationen. Es existiert noch die Möglichkeit $\mathcal{M}^{(15)}$ und $\mathcal{M}^{(16:17)}$ zu kombinieren, aber die Bewertungen für diese Kombination sind zu niedrig.

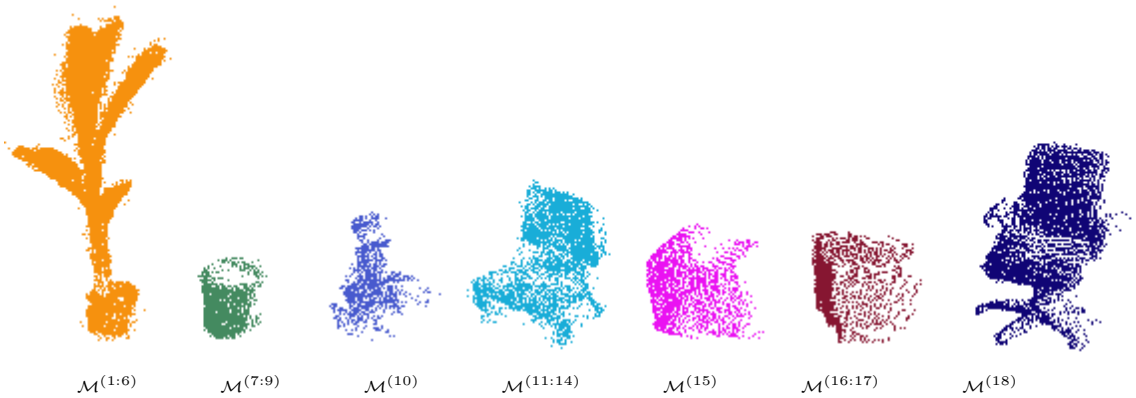

Abbildung 7.11: Die resultierende Modelldatenbank zeigt ein Pflanzenmodell bestehend aus sechs kombinierten Teilansichten, ein Mülleimermodell bestehend aus drei Teilansichten, das Modell für den Pioneer Roboter, ein Stuhlmodell bestehend aus vier kombinierten Teilansichten, zwei Monitormodelle für drei Teilansichten und das Modell für den Bürostuhl

dung 7.2 zeigt die dazugehörende Modelldatenbank. Für die sechs Teilansichten der Pflanze ist ein konsistentes Modell $\mathcal{M}^{(1:6)}$ gelernt worden, genauso wie für die drei Teilansichten der Mülleimer $\mathcal{M}^{(7:9)}$ und die vier Teilansichten der Stühle $\mathcal{M}^{(11:14)}$. Für den Monitor wurden zwei der drei Teilansichten kombiniert. Die Ähnlichkeitsmatrix in Abb. 7.10 enthält noch einen Cluster für $\mathcal{M}^{(15)}$ und $\mathcal{M}^{(16:17)}$. Die Bewertungen diese Clusters sind jedoch zu schwach, um mit Sicherheit sagen zu können, dass das Resultat ein konsistentes Modell ergeben würde. In diesem Fall wäre es zwar denkbar das kombinierte Modell zu lernen, jedoch sollten im Allgemeinen keine Modelle mit einer so geringen Bewertung zusammengeführt werden.
Die Ergebnisse dieses Experiments lassen darauf schließen, dass der Ansatz gut in Büroumgebungen angewandt werden kann.

7.3 Statistik über fehlerhafte Modellzuordnungen

Eine Interessante Fragestellung ist, ob zusätzliche Informationen bzw. Teilansichten das Ergebnis für verschiedene Objekttypen verbessern. Intuitiv müsste das Ergebnis besser werden, je mehr Teilansichten für jeden Objekttyp vorhanden sind, da inkonsistente Ausreißer in der Ähnlichkeitsmatrix weniger ins Gewicht fallen. Um diesen Sachverhalt zu Untersuchen, wurden sechs verschiedene Modelldatenbanken zusammengestellt und für verschiedene Objekttypen die Anzahl an verfügbaren Teilansichten variiert. Als Objekte wurden die in den bereits vorgestellten Experimenten gesammelten Modelle Pflanze, Stuhl, Monitor und Mülleimer ausgewählt. Die erste Modelldatenbank bestand aus zwei Modellen für jeden Objekttyp und die

letzte aus sieben Modellen je Objekttyp. Als fehlerhaft gelerntes Modell wurden folgende Sachverhalte gezählt:

1. Modell enthält eine **falsche Teilansicht**, die zu einem anderen Objekttyp gehört.

2. **Nicht gelernte Teilansicht**. Hierzu zählt jede Teilansicht die zu einem Objekttyp gehört, aber nicht in das maximal konsistente Modell dieses Objekttyps integriert wurde.

3. Modell das einem anderen Modell des selben Objekttyps zugeordnet wird, aber innerhalb des Objekts an **falscher Position** eingefügt wird.

Der Anteil an richtig zugeordneten Teilansichten für einen Objekttyp wurde auf folgende Art berechnet:

$$\text{Anteil} = \frac{\text{Anzahl Fehler}}{\text{Anzahl Teilansichten}} \tag{7.1}$$

Abbildung 7.12 zeigt die Ergebnisse dieses Experiments. Interessant an den Ergebnissen ist, dass es keinen Fall gab, in dem eine Teilansicht einem falschen Objekttyp zugeordnet wurde. Auch das Einfügen eines Modells an falscher Position in ein anderes Modell des gleichen Objekttyps, trat nur für den Objekttyp Monitor auf. Dabei handelt es sich um eine Teilansicht die durchgängig nicht richtig positioniert wurde. Dies hängt wahrscheinlich mit einer geringen Überlappung besagter Teilansicht zu den verfügbaren Nachbaransichten zusammen und der beinahe würfelförmigen Objektstruktur des Monitors. Ist nur ein Teil eines Würfels bekannt, neigt die Registrierung dazu eine neue Würfelseite eher in den bekannten Bereich zu registrieren. Es ist durchaus wahrscheinlich, dass diese Teilansicht, mit zusätzlichen überlappenden Teilansichten, richtig positioniert werden könnte.
Abbildung 7.12 lässt auch darauf schließen, dass die Anzahl der notwendigen Teilansichten, um ein konsistentes Modell zu bauen, stark von der Objektstruktur abhängt. Die Modelle des symmetrischen Objekttyps Mülleimer wurden, unabhängig von der Anzahl der Teilansichten, immer zu 100% richtig zugeordnet. Wie schon angedeutet stellen symmetrische Objekte eine Besonderheit, dar sie aus jeder Observationsrichtung gleich aussehen. Der Unterschied zwischen der Pflanze, dem Stuhl und dem Monitor liegt in der Dichte der strukturellen Information. Während bei der Pflanze wenig ebene Oberfläche und eingeschlossenes Volumen vorhanden ist, verfügt der Monitor sowohl über viel ebene Oberfläche, als auch über ein großes eingeschlossenes Volumen. Bei Objekten mit weniger struktureller Information benötigt der vorgestellte Ansatz mehr Teilansichten des Objektes, um ein konsistentes Modell zu lernen.

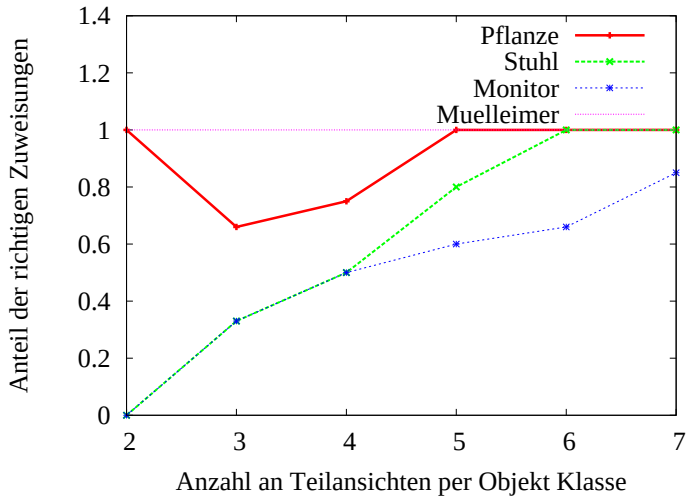

Abbildung 7.12: Verhalten des Algorithmus als Funktion der Anzahl an Teilansichten pro Objekttyp. Jede Kurve repräsentiert den Anteil an richtig zugeordneten Teilansichten für ein spezifisches Objekt, die in ein konsistentes Modell zusammengeführt worden sind.

7.4 Abhängigkeit von der Tiefenbilderanzahl

In diesem Experiment geht es um den Einfluss der Tiefenbilderanzahl auf die Ähnlichkeitsmatrix. Wie viele Tiefenbilder sind nötig, um gute Ergebnisse mit der Globalen Registrierung zu erhalten? Je weniger Tiefenbilder genutzt werden, desto größer sind die Abstände der Positionen rund um das Objekt, aus denen die Merkmale berechnet werden. Dabei wird die initiale Modelldatenbank aus Experiment 7.2 als Eingabe benutzt. Für diese wird jeweils die initiale Ähnlichkeitsmatrix mit variierender Tiefenbilderanzahl berechnet. Die Anzahl der Tiefenbilder wird Schrittweise von 1 bis 21 erhöht.
Abbildungen 7.13 und 7.14 zeigen mehrere resultierenden Ähnlichkeitsmatrizen dieses Experiments. Wenig überraschend sind die Hypothesen für den symmetrischen Mülleimer auch schon mit einem Tiefenbild optimal, da der Mülleimer aus jeder Richtung gleich aussieht. Interessant an Abbildung 7.14 ist, dass die Hypothesen kaum noch besser werden im Vergleich zwischen 12 und 20 Tiefenbildern. Die Anzahl der inkonsistenten Hypothesen verringert sich jedoch. Dies könnte damit zusammen hängen, dass der Hypothesengraph ab 200 Hypothesen beschnitten wird.
Die Ergebnisse lassen darauf schließen, dass 11 bis 12 Tiefenbilder für die Hypothesengenerierung ausreichend sind. In Anhang B.1 sind alle 22 Ähnlichkeitsmatrizen aufgeführt.

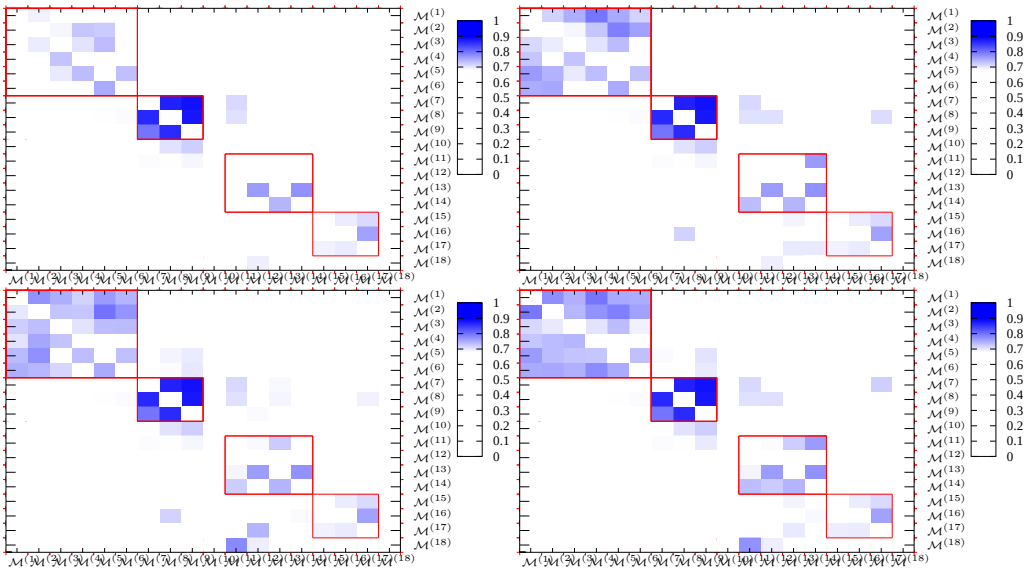

Abbildung 7.13: Ähnlichkeitsmatrizen für eine Modelldatenbank mit variierender Tiefenbilderanzahl. Die Erste Matrix steht für ein Tiefenbild pro Modell, aus denen die Merkmale für die globale Registrierung berechnet werden. In den folgenden Matrizen wird die Anzahl der Tiefenbilder jeweils um eines erhöht. Bei der Berechnung dieser Matrizen wurde der Hypothesengraph bei 200 Hypothesen pro Objekt beschnitten.

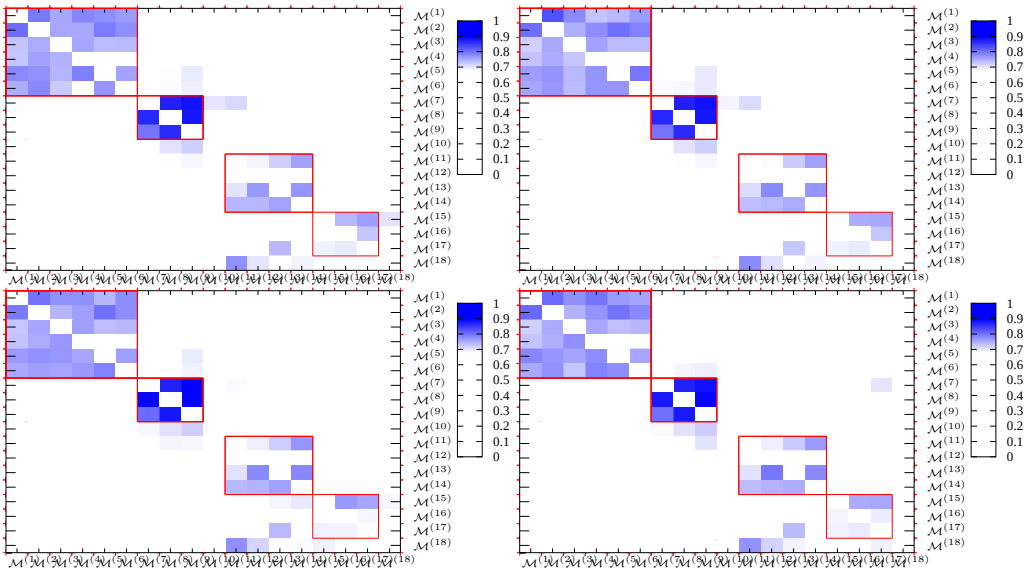

Abbildung 7.14: Ähnlichkeitsmatrizen für 11, 12, 20 und 21 Tiefenbilder. Die Information für die Modellgruppen nimmt mit zunehmender Tiefenbilderanzahl kaum noch zu. Die sichtbaren Hypothesen außerhalb der mit roten Rechtecken gekennzeichneten Modellzugehörigkeiten nehmen jedoch ab.

7.4.1 Unbeschnittener Hypothesengraph

Wie in Kapitel 6.2.1 beschrieben, wird die Anzahl an Hypothesen für ein Modell beschränkt. Um die Auswirkungen dieser Einschränkung zu klären, wurde das Experiment ohne die Beschränkung auf n Hypothesen wiederholt. Abbildung 7.15 zeigt die Ergebnisse für 1,2,11 und 12 Tiefenbilder, ohne Beschränkung der Hypothesenanzahl.
Die ersten beiden Ähnlichkeitsmatrizen in Abbildung 7.15 zeigen kaum Unterschiede zu den entsprechenden Ähnlichkeitsmatrizen für 1 und 2 Tiefenbilder mit beschnittenem Hypothesengraph in Abbildung 7.13. Dies hängt damit zusammen, dass hier die Hypothesengraphbeschränkung noch keine Wirkung zeigt. Es werden weniger als 200 Hypothesen pro Modell gefunden. Für 12 Tiefenbilder ist der Unterschied jedoch deutlich zu sehen, es gibt wesentlich mehr Hypothesen außerhalb der roten Rechtecke, welche die Tatsächlichen Modellzugehörigkeiten kennzeichnen. Beim Beschneiden des Hypothesengraphen, werden also eher schlechte Ergebnisse verworfen. Dies hängt damit zusammen, dass die Transformationen aus der globalen Registrierung bereits nach der GOODSAC-Qualität geordnet sind. Neben der Ergebnisqualität ist auch die unterschiedliche Rechenzeit zwischen den beiden Experimentalbedingungen interessant. Hiermit befassen wir uns im Folgenden.

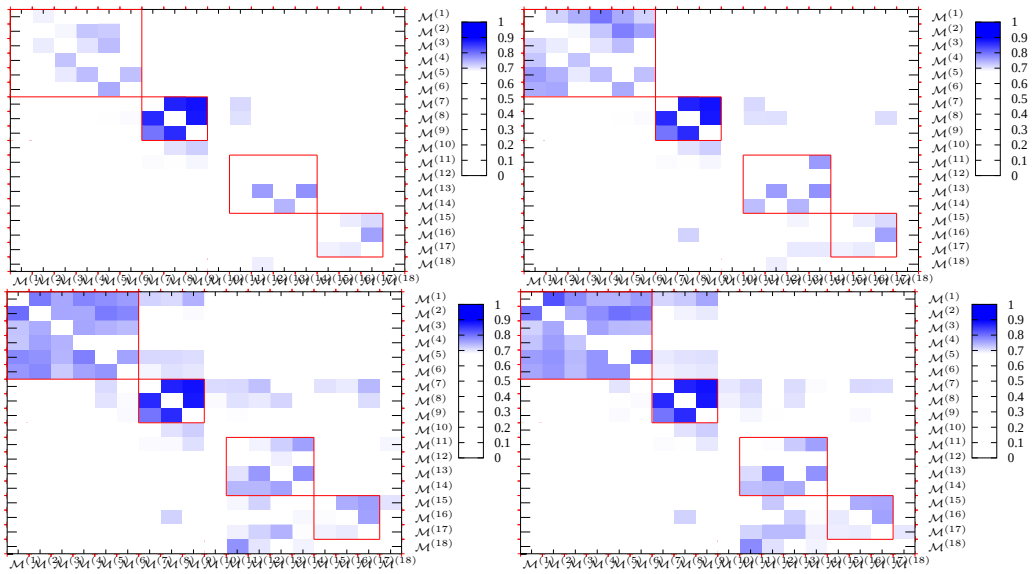

Abbildung 7.15: Ähnlichkeitsmatrizen für 1, 2, 11 und 12 Tiefenbilder ohne Beschneidung des Hypothesengraphen. Die ersten beiden Ähnlichkeitsmatrizen zeigen keinen großen Unterschied zu den korrespondierenden Matrizen in 7.13. Die Ähnlichkeitsmatrizen für 11 und 12 Tiefenbilder zeigen mehr Hypothesen außerhalb der tatsächlichen Modellzugehörigkeiten als in die korrespondierenden Matrizen in 7.14

7.4.2 Leistungsanalyse

Als Hardware kam ein Doppelkern Prozessor mit 1,7 Ghz zum Einsatz, ausgestattet mit 2 GByte RAM. Dabei ist zu beachten, dass die Software nur einen Prozessorkern auslastet. Die Rechenzeiten für dieses Experiment sind in Abbildung 7.16 zu sehen. Die unterschiedliche Entwicklung der Rechenzeit ist gut zu erkennen, für den beschnittenen Hypothesengraph (in rot) und den unbeschnittenen (in grün). Mehr Tiefenbilder bedeuten auch mehr Merkmale für die GOODSAC Berechnung, daher steigt die Rechenzeit auch für den beschnittenen Hypothesengraph leicht an. Im Gegensatz dazu steigt die Rechenzeit für den unbeschnittenen Hypothesengraph nahezu linear, mit der Tiefenbilderanzahl. Dabei werden die Ergebnisse wie in Kapitel 7.4.1 gezeigt nicht besser.

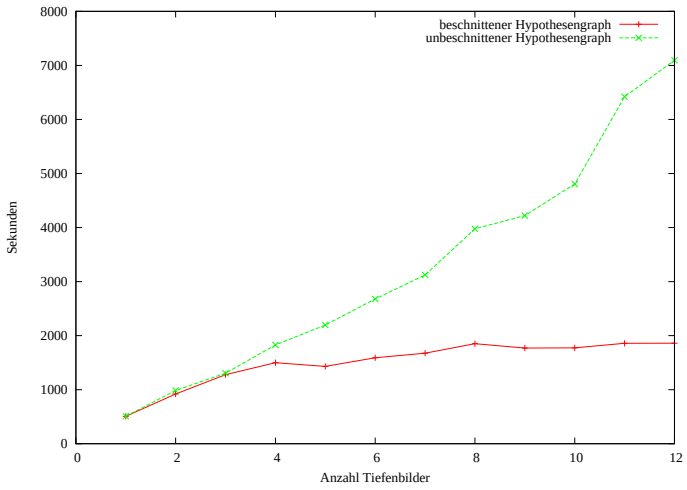

Abbildung 7.16: Berechnungszeit für die vollständige initiale Modelldatenbank, für variierende Tiefenbilderanzahl. Ein Durchgang wurde mit beschnittenem Hypothesengraph (rot) und einer mit unbeschnittenem Hypothesengraph durchgeführt.

8 Zusammenfassung

In dieser Arbeit wurde ein Ansatz zum Lernen nicht stationärer Objekte auf volumetrischen Daten vorgestellt. Dabei wurde gezeigt, wie Objektansichten aus den Sensordaten eines Laser-Range-Finders extrahiert werden können.
Ausgehend von diesen Teilansichten, wurde eine Modelldatenbank aufgebaut, mit dem Ziel zusammengehörende Teilansichten zu identifizieren und diese iterativ zusammenzuführen. Es wurde ein Ansatz für die globale Registrierung, basierend auf Tiefenbildern vorgestellt, der auch bei sehr geringer Überlappung gute Ergebnisse liefert. Mit dessen Hilfe wurde unter der Annahme, dass die betrachteten Modelle zum gleichen Objekttyp gehören, eine Liste an möglichen Transformationen berechnet. Diese Transformationen wurden mit Heuristiken bewertet und in einen Hypothesengraphen eingefügt. Die besten Hypothesen aus diesem Graph werden in einer Ähnlichkeitsmatrix dargestellt. Um die zusammengehörenden Teilansichten zu kombinieren, wurde die Ähnlichkeitsmatrix mit Hilfe eines Graphpartitionierungsansatzes aufgeteilt und die einzelnen Partitionen wurden iterativ zusammengeführt. Im Anschluss daran wurde die Optimierung möglichst konsistenter Objekte diskutiert und eine Reihe von Experimenten vorgestellt, welche die Performanz des Ansatzes belegen. Dabei wurde der Ansatz mit verschiedenen Szenarien getestet und die Abhängigkeit der Anzahl von Teilansichten für die Konvergenz eines Objektmodells gezeigt. Zusätzlich wurde die Abhängigkeit der Tiefenbilderanzahl auf die Ähnlichkeitsmatrix getestet.

8.1 Mögliche Problemquellen

Die Ergebnisse dieser Experimente sind vielversprechend, jedoch gibt es einige Szenarien, in denen der Ansatz nur bedingt gute Ergebnisse erzielen kann. Symmetrische Objekte stellen eine besondere Herausforderung dar. Im Allgemeinen ist es nicht ohne zusätzliche Informationen möglich ein vollständiges Modell zu bauen, da die verschiedenen Positionen für Teilansichten durch eine globale Registrierung nicht eindeutig zu lösen sind. Für die Objekterkennung ist diese Einschränkung jedoch irrelevant. Da die Teilansichten auf ein symmetrisches Modell sehr ähnlich aussehen, reichen auch die Merkmale einer Teilansicht, um ein Objekt in einem 3D Scan zu finden.
Da der vorgestellte Ansatz strukturelle Informationen braucht, liefert er nur schlechte Ergebnisse bei sehr ebenen Strukturen. Eine kleine Fläche lässt sich mit 100%

Überlappung auf eine größere Ebene legen, ohne dass neue Information gelernt wird. Dieses Problem ist ohne zusätzliche Information (beispielsweise die Farbe aus Kamerabildern) in den Punktwolken nicht zu lösen. Wobei man natürlich solche Zusammenführungen einfach lassen könnte, da sie keine neue Information liefern.

Eine weitere Schwierigkeit dieses Ansatzes ist es, dass Objekte keine beliebig ähnliche Struktur haben dürfen. Gibt es Instanzen eines Objektes, die sich in nur einem kleinen Detail unterscheiden, würde dieses Detail wohl als Fehler gefiltert werden. Ein Beispiel für so eine abweichende Instanz wäre ein Stuhl, den es auch mit Armauflagen gibt. Es wäre denkbar, leicht abweichende Objektinstanzen in einem Modell zu organisieren. Ein solches Modell könnte mehrere Prototypen enthalten. Auf diese Weise könnten Objekte flexibler modelliert werden.
Verformbare Objekte liefern schlechte Ergebnisse mit diesem Ansatz. Selbst einfache Bürostühle haben einen Freiheitsgrad, in dem der obere Teil zum Fuß verdreht werden kann. Das strukturelle Lernen kann nur einen der beiden Teile erfolgreich registrieren. Der andere Teil ist damit inkonsistent registriert.

8.2 Ausblick

Es gibt eine ganze Reihe an interessanten Fragestellungen, bei denen sich eine weitere Betrachtung lohnt. Diese lassen sich in zwei Kategorien einteilen, zum einen das direkte Weiterführen des Ansatzes und zum anderen die Kombination mit bestehenden Techniken und sich daraus neu ergebender Möglichkeiten.
Ein Erhöhen der Performanz des Ansatzes ohne einen Verlust an Genauigkeit ist wünschenswert. Dies könnte zum einen durch weitere Optimierung auf dem Hypothesengraph geschehen und zum anderen durch effizientere Berechnungen der Heuristiken. Mit zusätzlichen Heuristiken könnte auch die Diskretisierung von guten und schlechten Hypothesen weiter verbessert werden.
Das Lernen vollständiger Modelle für symmetrische Objekte sollte mit Heuristiken möglich sein. Ein symmetrisches Objekt ist daran zu erkennen, dass eine hinreichend große Anzahl an Teilansichten vorhanden ist, die die gleiche Struktur enthält. Ist weiterhin keine Teilansicht vorhanden, die eine andere Struktur enthält, könnte die Teilansichten mit einer alternativen Registrierung um den Objektmittelpunkt angeordnet werden, so dass die Überlappung zwischen den Teilansichten höchstens bis zu einem bestimmten Grenzwert minimiert wird.
Der in dieser Arbeit vorgestellte Ansatz könnte für andere Objekttypen als nicht stationäre Objekte erweitert werden. Statische Objekte lassen sich mit einem erweiterten Segmentierungsalgorithmus extrahieren. Die Wahrnehmung von dynamischen Objekten hängt primär von der Geschwindigkeit ab, mit dem eine Sensormessung erfolgt. Ist das Messintervall des Sensors klein genug, lässt sich ein dynamisches Objekt identifizieren und auch segmentieren. Für die genannten Objekttypen liegt die

Schwierigkeit also eher in der Segmentierung. Um verformbare Objekte mit wenigen Freiheitsgraden zu lernen, also beispielsweise Bürostühle mit einer drehbaren Sitzfläche, muss die Registrierungstechnik erweitert werden. Es gibt bereits Arbeiten, die sich mit der Registrierung von Objektteilen für verformbare Objekte beschäftigen, siehe [2]. Mit zusätzlichen Freiheitsgraden wird das Problem, konsistente Objektmodelle zu lernen natürlich noch schwieriger.
Durch eine Kombination des heuristischen Modelllernens, mit einer auf dem Roboter laufenden Objekterkennung, sollte es möglich sein, Modelle rekursiv zu verfeinern. Wird ein Modell in einer Szene gefunden, kann die Observation in die Modellliste eingefügt werden. Inkonsistente Teilansichten müssen verworfen werden, bevor sie eine Objekterkennung unmöglich machen. Das vorgestellte Verfahren könnte offline die initialen Modelle berechnen und, um einen rekursiven Updateschritt erweitert, so online die Modelle verfeinern.
Spannend wäre auch eine Kombination aus Modelllernen, Objekterkennung und einem Simultaneous Localization and Mapping (SLAM) Ansatz. Damit könnte eine 3D Karte der Umgebung, auch in Bereichen die bisher nur teilweise exploriert worden sind, mit den erkannten Objekten vervollständigt werden. Und die in der Karte erkannte Teilansicht könnte wieder für die Verfeinerung des Modells genutzt werden.

A Verwendete Hardware

A.1 Mobiler Roboter

A.1.1 Roboter Basis

Maße (B × H × T)	850mm × 625mm × 430mm
Antrieb	Differential
Gewicht	4,5 kg
Nutzlast	100 kg
Max. Operationszeit	2,5 h
Aufladezeit	5 h
Max. Steigung	15 %
Max. Geschwindigkeit	6 km / h

Abbildung A.1: Die verwendete Roboterbasis, ein PowerBot von ActivRobots

A.1.2 Laser-Scanner

Parameter	Wert
Reichweite	max. 80m
Scanwinkel	max. 180°
Messauflösung	10 mm
Mögliche Winkelauflösungen	0,25° / 0,5° / 1°
Versorgungsspannung	24 VDC ± 15 %
Leistungsaufnahme	ca. 20 W
Maße (B × H × T)	155mm × 185mm × 156 mm
Gewicht	4,5 kg

Abbildung A.2: SICK LMS 291

A.1.3 Servo Schwenk-Neige-Einheit

	Pan	Tilt
Maximale Winkelgeschwindigkeit [°/sec]	149	248
Auflösung [Winkelsec/Inc]	4	5
Lageerfassung [Inc/°]	894	672
Wiederholgenauigkeit der Positionierung [°]	± 0,02	± 0,02
Spannung [V]	24 ± 1	
Max. Strom [A]	15	
Masse [kg]	3,4	

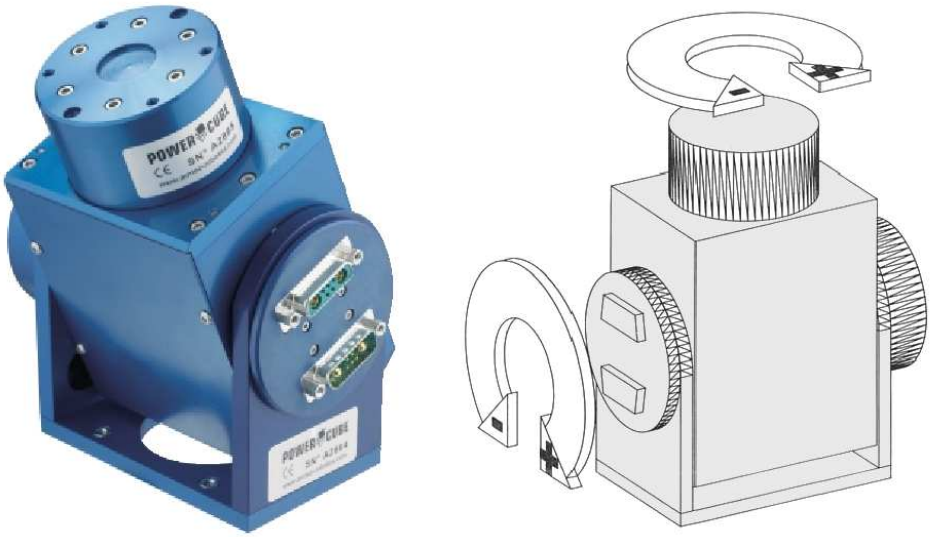

Abbildung A.3: Amtec PowerCube PW090

B Experimentaldaten

B.1 Vollständiger Datensatz zu Experiment 7.4

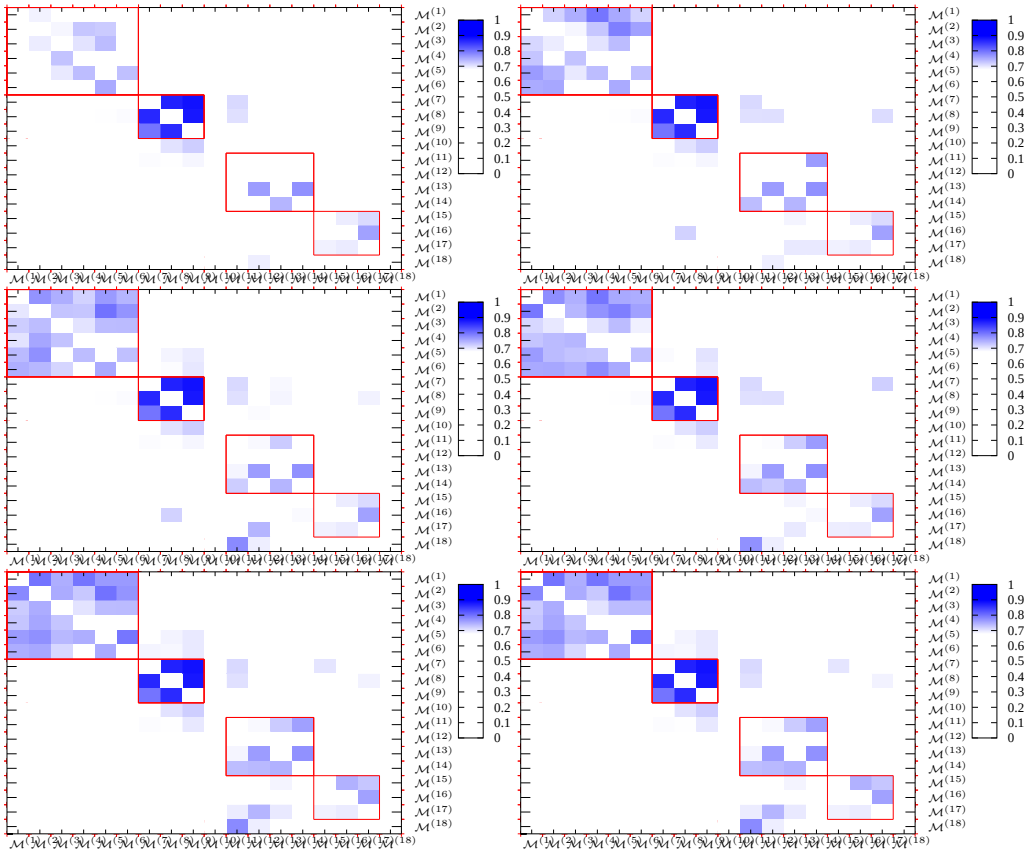

Abbildung B.1: Ähnlichkeitsmatrizen für eine Modelldatenbank mit varrierender Tiefenbilderanzahl. Tiefenbilderanzahl 1 - 6

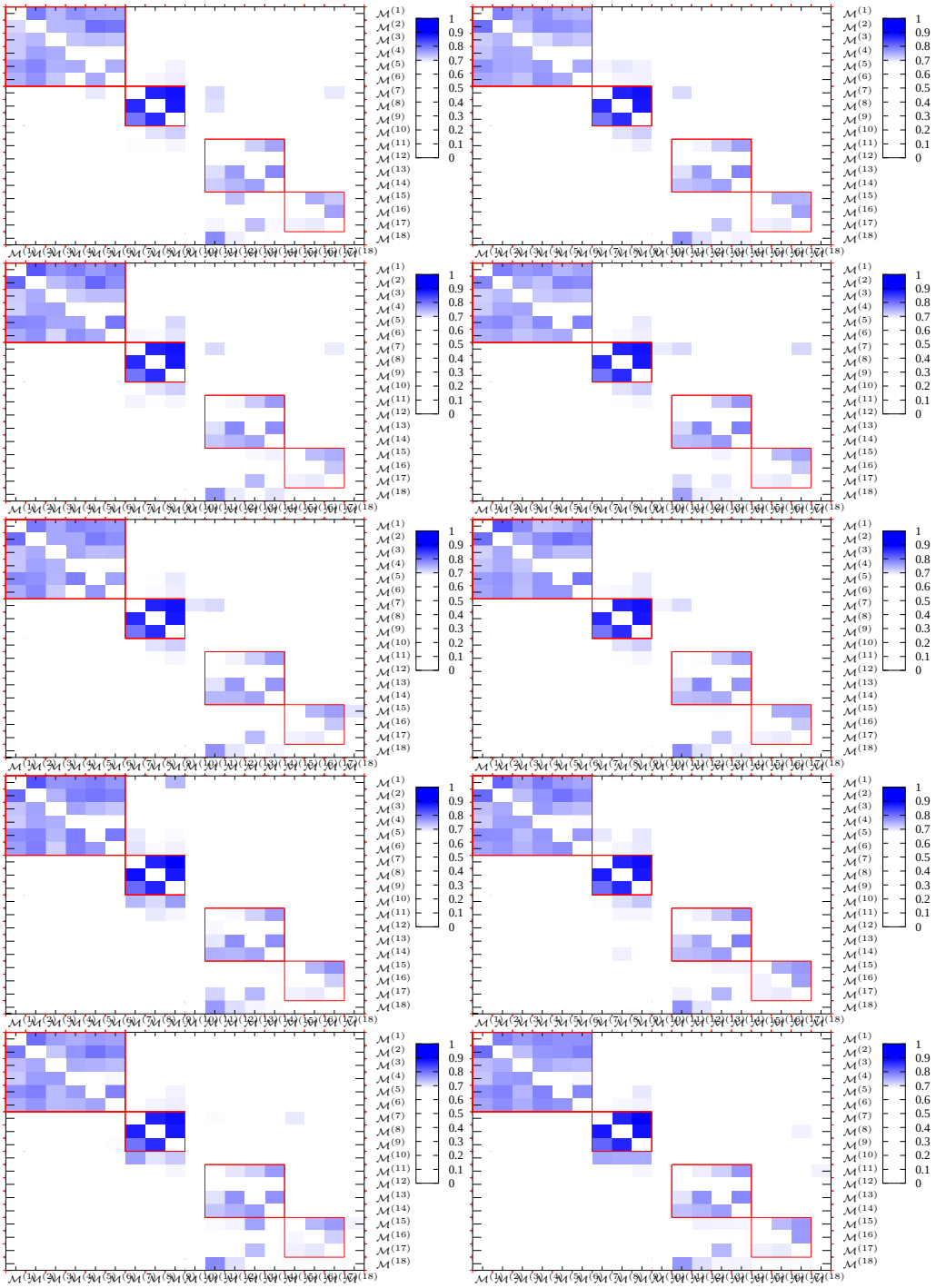

Abbildung B.2: Ähnlichkeitsmatrizen für eine Modelldatenbank mit varrierender Tiefenbilderanzahl. Tiefenbilderanzahl 7 - 16

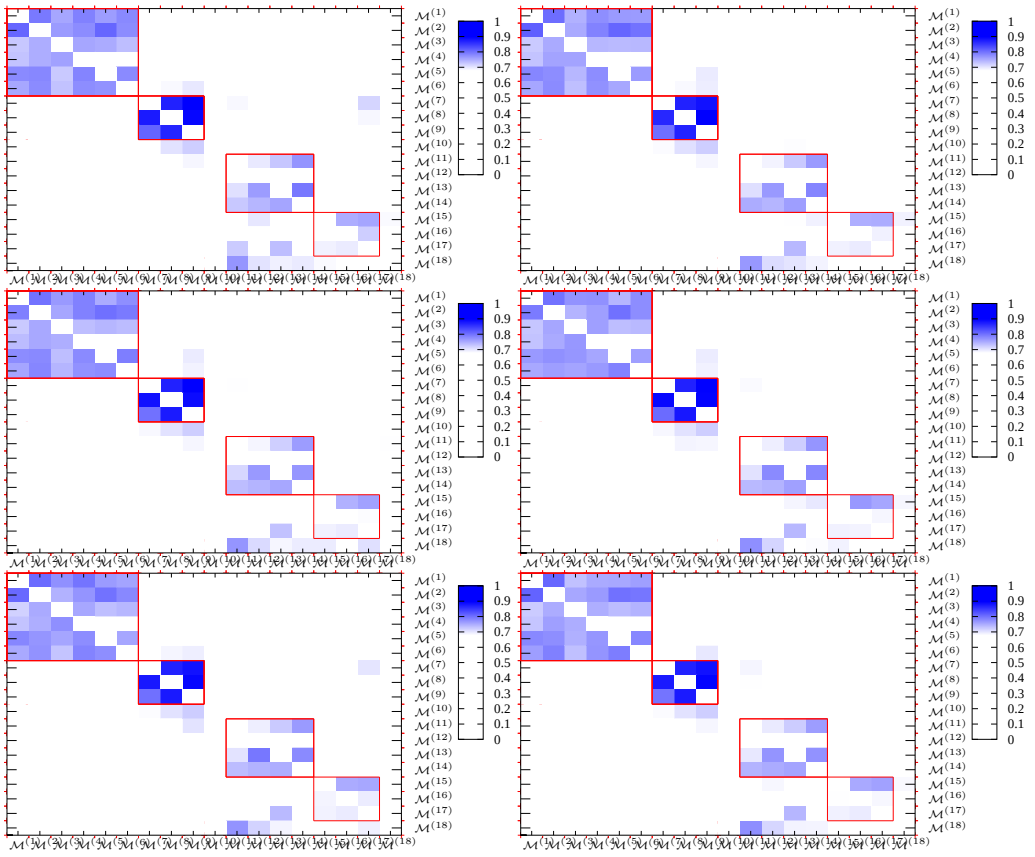

Abbildung B.3: Ähnlichkeitsmatrizen für eine Modelldatenbank mit varrierender Tiefenbilderanzahl. Tiefenbilderanzahl 17 - 22

Literaturverzeichnis

[1] D. Anguelov, R. Biswas, D. Koller, B. Limketkai, S. Sanner, and S. Thrun. Learning hierarchical object maps of non-stationary environments with mobile robots. *Proc. of the Conf. on Uncertainty in Artificial Intelligence (UAI)*, pages 10–17, 2002.

[2] D. Anguelov, P. Srinivasan, H.-C. Pang, D. Koller, S. Thrun, and J. Davis. The correlated correspondence algorithm for unsupervised registration of nonrigid surfaces. In L. Saul, Y. Weiss, and L. Bottou, editors, Proceedings of Conference on Neural Information Processing Systems (NIPS), Cambridge, MA. MIT Press. 2004

[3] J. D. Foley, A. Van Dam, K Feiner, J.F. Hughes, and Phillips R.L. *Introduction to Computer Graphics.* Addison-Wesley, 1993.

[4] Natasha Gelfand, Niloy J. Mitra, Leonidas J. Guibas, and Helmut Pottmann. Robust global registration. In *SGP '05: Proceedings of the third Eurographics symposium on Geometry processing*, page 197, Aire-la-Ville, Switzerland, Switzerland, 2005. Eurographics Association.

[5] A. Makadia, A. Patterson, und K. Daniilidis. Fully Automatic Registration of 3D Point Clouds. In *IEEE Conference on Computer Vision and Pattern Recognition*, Seiten 1297–1304, 2006.

[6] Wikipedia - Corner detection. http://en.wikipedia.org/wiki/Corner_detection, Abruf 20.10.08.

[7] Moravec, Hans: Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover. In *Tech. report CMU-RI-TR-80-03, Robotics Institute, Carnegie Mellon University and doctoral dissertation, Stanford University.* 1980, Kapitel 5

[8] C. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings of The Fourth Alvey Vision Conference*, pages 147–151, 1988.

[9] K. Berthold P. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America. A*, 4(4):629–642, Apr 1987.

[10] B. Steder Techniken für bildbasiertes SLAM unter Verwendung von Lagesensoren Diplomarbeit, 2007.

[11] Andrew E. Johnson and Martial Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 21(5):433–449, 1999.

[12] A. S. Mian, M. Bennamoun, and R. A. Owens. 3d recognition and segmentation of objects in cluttered scenes. In *WACV-MOTION '05: Proceedings of the Seventh IEEE Workshops on Application of Computer Vision (WACV/MOTION'05) - Volume 1*, pages 8–13, Washington, DC, USA, 2005. IEEE Computer Society.

[13] E. Michaelsen, W. von Hansen, M. Kirchhof, J. Meidow, and U. Stilla. Estimating the essential matrix: Goodsac versus ransac. 2006.

[14] E. Olson, M. Walter, S. Teller, and J. Leonard. Single-Cluster Spectral Graph Partitioning for Robotics Applications. *Robotics: Science and Systems, RSS*, 5.

[15] Salvador Ruiz-Correa, Linda G. Shapiro, and Marina Meila. A new signature-based method for efficient 3-d object recognition. In *CVPR (1)*, pages 769–776, 2001.

[16] Stefan Stiene, Kai Lingemann, Andreas Nuchter, and Joachim Hertzberg. Contour-based object detection in range images. In *3DPVT '06: Proceedings of the Third International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT'06)*, pages 168–175, Washington, DC, USA, 2006. IEEE Computer Society.

[17] R. Triebel and W. Burgard. Recovering the shape of objects in 3d point clouds with partial occlusions. In *Proc. of the 6th International Conference on Field and Service Robotics (FSR)*, 2007.

[18] R. Triebel, R. Schmidt, O. Martinez Mozos, and W. Burgard. Instance-based amn classification for improved object recognition in 2D and 3D laser raneg data. In *Proc. of the Int. Conf. on Artificial Intelligence (IJCAI)*, 2007.

[19] Bentley, J.L. Multidimensional binary search trees used for associative searching. In *Communications of the ACM*, Seiten 509–517, 1975.

[20] Wald, I. and Havran, V. On building fast kd-Trees for Ray Tracing, and on doing that in O (N log N) Interactive Ray Tracing 2006, IEEE Symposium on, Seiten 61–69, 2006

[21] Wikipedia - kD-Tree http://en.wikipedia.org/wiki/Kd-tree, Abruf 20.10.08.

[22] Besl, PJ and McKay, ND A Method for Registrationof 3D shapes, In *IEEE transactions on pattern analysis and machine intelligence*, 1992.

[23] R. Triebel. Three-dimensional Perception for Mobile Robots *Dissertation*, 2007.

[24] Umeyama S. Least-square estimation of transformation parameters between two point patterns, In IEEE PAMI,13:376-380, 1991.

[25] Adams, R. and Bischof, L. Seeded Region Growing PAMI, Volume 16, Seiten 641-647, 1994.

[26] Fischler, Martin A. ; Bolles, Robert C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. In: Commun. ACM 24 (1981), Nr. 6, Seiten 381–395

[27] Chum, O. and Matas, J. Matching with PROSAC - Progressive Sample Consensus. In: Proc. of Conference on Computer Vision and Pattern Recognition (CVPR) Bd. 1, IEEE Computer Society, 2005, Seiten 220–226

[28] Wikipedia - Perron-Frobenius Theorem http://en.wikipedia.org/wiki/Perron-Frobenius_theorem, Abruf 5.11.08.