UNIVERSITÄT
KOBLENZ · LANDAU
Fachbereich 4: Informatik

Terrainklassifikation für ein autonomes mobiles System

Studienarbeit
im Studiengang Informatik

vorgelegt von

Denis Dillenberger

Betreuer: Dipl.-Inform. Johannes Pellenz, Institut für Computervisualistik, Fachbereich Informatik, Universität Koblenz-Landau
Erstgutachter: Dipl.-Inform. Johannes Pellenz, Institut für Computervisualistik, Fachbereich Informatik, Universität Koblenz-Landau
Zweitgutachter: Prof. Dr.-Ing. Dietrich Paulus, Institut für Computervisualistik, Fachbereich Informatik, Universität Koblenz-Landau

Koblenz, im Juli 2009

Kurzfassung

Das sichere Befahren von komplexen und unstruktierten Umgebungen durch autonome Roboter ist seit den Anfängen der Robotik ein Problem und bis heute eine Herausforderung geblieben. In dieser Studienarbeit werden drei Verfahren basierend auf 3-D-Laserscans, Höhenvarianz, der Principle Component Analysis (PCA) und Tiefenbildverarbeitung vorgestellt, die es Robotern ermöglichen, das sie umgebende Terrain zu klassifizieren und die Befahrbarkeit zu bewerten, sodass eine sichere Navigation auch in Bereichen möglich wird, die mit reinen 2-D-Laserscannern nicht sicher befahren werden können.

Hierzu werden 3-D-Laserscans mit einem 2-D-Laserscanner erstellt, der auf einer Roll-Tilt-Einheit basierend auf Servos montiert ist, und gleichzeitig auch zur Kartierung und Navigation eingesetzt wird. Die einzeln aufgenommenen 2-D-Scans werden dann anhand des Bewegungsmodells der Roll-Tilt-Einheit in ein gemeinsames 3-D-Koordinatensystem transformiert und mit für die 3-D-Punktwolken Verarbeitung üblichen Datenstrukturen (Gittern, etc.) und den o.g. Methoden klassifiziert.

Die Verwendung von Servos zur Bewegung des 2-D-Scanners erfordert außerdem eine Kalibrierung und Genauigkeitsbetrachtung derselben, um zuverlässige Ergebnisse zu erzielen und Aussagen über die Qualität der 3-D-Scans treffen zu können.

Als Ergebnis liegen drei Implementierungen vor, welche evolutionär entstanden sind. Das beschriebene Höhenvarianz-Verfahren wurde im Laufe dieser Studienarbeit von einem Principle Component Analysis basierten Verfahren, das bessere Ergebnisse insbesondere bei schrägen Untergründen und geringer Punktdichte bringt, abgelöst. Die Verfahren arbeiten beide zuverlässig, sind jedoch natürlich stark von der Genauigkeit der zur Erstellung der Scans verwendeten Hardware abhängig, die oft für Fehlklassifikationen verantwortlich war. Die zum Schluss entwickelte Tiefenbildverarbeitung zielt darauf ab, Abgründe zu erkennen und tut dies bei entsprechender Erkennbarkeit des Abgrunds im Tiefenbild auch zuverlässig.

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Die Vereinbarung der Arbeitsgruppe für Studien- und Abschlussarbeiten habe ich gelesen und anerkannt, insbesondere die Regelung des Nutzungsrechts.

Mit der Einstellung dieser Arbeit in die Bibliothek bin ich einverstanden.	ja ☒	nein ☐
Der Veröffentlichung dieser Arbeit im Internet stimme ich zu.	ja ☒	nein ☐

Koblenz, den 31. Juli 2009

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Einleitung

Das sichere Befahren von komplexen und unstruktierten Umgebungen durch autonome Roboter ist seit den Anfängen der Robotik ein Problem und bis heute eine Herausforderung geblieben. In dieser Studienarbeit werden drei Verfahren basierend auf 3-D-Laserscans, Höhenvarianz, der Principle Component Analysis (PCA) und Tiefenbildverarbeitung vorgestellt, die es Robotern ermöglichen das sie umgebende Terrain zu klassifizieren und die Befahrbarkeit zu bewerten, sodass eine sichere Navigation auch in Bereichen möglich wird, die mit reinen 2-D-Laserscannern nicht sicher befahren werden können.

1.1 Ziel und Kontext der Arbeit

Ziel der Arbeit ist die Erstellung beliebiger Arten von 3-D-Scans auf Pan-Tilt-Roll-Einheiten, sowie die Transformation der Einzelscans in ein gemeinsames 3-D-Koordinatensystem und die Untersuchung von 3-D-Scans auf befahrbare Flächen. Diese Verfahrung sollen als Erweiterung der autonomen Kartierungs- und Navigationsfunktionen der mobilen Software und Platform *„Robbie“* dienen.

Die Arbeit wurde im Kontext des RoboCup[1] in den Ligen RoboCup Rescue und RoboCup@Home geschrieben. Während der Entstehung dieser Arbeit wurden die Verfahren in mehreren Wettbewerben erprobt.

1.2 Aufbau der Arbeit

In Kapitel 2 werden die bisherigen Aktivitäten der Universität im Bereich der Robotik, effiziente Datenstrukturen zur Analyse und Verarbeitung von großen

[1] `http://www.robocup.org/`

Abbildung 1.1: RoboCup Rescue Weltmeisterschaft (Graz, 2009): Liga Foto

3-D-Datenmengen, sowie verschiedene Ansätze zur Terrainklassifikation vorgestellt.

In Kapitel 3 werden die eigenen Ansätze inklusive der verwendeten Hardware zur Erstellung von 3-D-Scans beschrieben. Es werden drei Ansätze zur Klassifikation von Terrain beschrieben. Ein Verfahren basierend auf der Höhenvarianz einer Punktmenge, ein zweites auf Basis der Principle Component Analysis sowie zum Schluß eine Methode zur Detektion von Unregelmäßigkeiten im Tiefenbild, welche auf der Detektion von Kanten aus der Bildverarbeitung basiert.

In Kapitel 4 werden die Ergebnisse der aus Kapitel 3 implementierten Verfahren und durchgeführten Experimente vorgestellt.

In Kapitel 5 werden die Ergebnisse dieser Studienarbeit noch einmal zusammengefasst, sowie ein Ausblick gegeben, der Vorschläge für die evtl. Fortführung dieser Arbeit macht.

1.3 Danksagung

An dieser Stelle möchte ich mich bei einigen Personen und Institutionen bedanken, welche mich bei der Erstellung dieser Studienarbeit unterstützt und betreut haben:

Meinem Betreuer Johannes Pellenz für die Geduld und Zeit, die er bei der Erstellung dieser Arbeit mitbringen musste. Frank Neuhaus für sein umfangreiches C++- und Mathematik-Wissen. Christian Fuchs für die kompentente Beratung in Bildverarbeitungsfragen. Sowie, last, but not least, der Robocup Federation, welche den RoboCup veranstaltet, der für mich eine interessante Erfahrung und Bereicherung im Zuge meines Studiums dargestellt hat.

Kapitel 2

Stand der Wissenschaft

Dieses Kapitel beschreibt zum einen die eigenen Vorarbeiten der Arbeitsgruppe Aktives Sehen an der Universität Koblenz-Landau, sowie die gängige Technik für die Speicherung und effiziente Verarbeitung von 3-D-Daten. Außerdem werden einige veröffentlichte Verfahren zur Terrainklassifikation am Ende des Kapitels kurz erklärt.

2.1 Robotik-Projekte an der Universität Koblenz-Landau

Robbie ist das autonome mobile System der *Arbeitsgruppe Aktives Sehen* (AGAS) an der Universität Koblenz-Landau und befindet sich in der 12. Generation seiner Entwicklung. In den vergangenen 5 Jahren wurde der Roboter und vor allem die dazugehörige Software stetig weiterentwickelt. Die Software basiert auf einer in der Praxis ausgiebig erprobten und getesteten Architektur, welche sich hervorragend zur Entwicklung von Robotik-Anwendungen eignet. Der modulare Aufbau der gesamten Software ermöglicht die Wiederverwendung der Kernkomponenten und vieler Module in unterschiedlichsten Robotik-Projekten und Kontexten. In den Jahren 2007 und 2008 gelang es den Teams um *Robbie 8* und *Robbie X* den Weltmeistertitel in der Kategorie „Autonomie“ in der RoboCup Rescue-Liga in Atlanta (USA) und Suzhou (China) zu gewinnen. Im Jahr 2009 wurde das Team um *Robbie 12* als erstes Team mit einem autonomen Roboter Gesamtsieger bei den RoboCup German Open in Hannover. Robbie verfügt über Module, die es dem Roboter ermöglichen Karten zu erstellen und innerhalb von diesen autonom zu navigieren. Diese sind jedoch bislang auf 2-D-Navigation in Höhe des Laserscanners beschränkt, d.h. Hindernisse, welche sich nicht auf Höhe des Laserscanners befinden, werden nicht erkannt und sind potentielle Fallstricke für den Roboter.

2.2 Bisherige Arbeiten mit Laserscannern

In [Del07] wurden 3-D-Laserscans mittels einer Rotationsplatform basierend auf Servos für den *Hokuyo URG-04LX* erstellt. Der Laserscanner wurde mittels eines einzelnen Servos um 180° gedreht und somit eine Punktwolke der Umgebung erzeugt. Weiterhin wird in [Brö07] beschrieben, wie bestimmte Punkte in einem von einer Kamera aufgenommenen Bild mit Hilfe eines 2-D-Laserscanners und eines Servos angepeilt werden können. Der Laserscanner von Robbie ist außerdem mittels eines Accelerometers in der Lage seine Position mit Hilfe von zwei Servos (Roll und Tilt) auf Schrägen zu justieren, was für die Kartierung in unebenem Gelände notwendig ist. Diese Servos können zusätzlich für 3-D-Scans eingesetzt werden, was im folgenden Kapitel beschrieben wird

Abbildung 2.1: Robbie im Wettbewerb (RoboCup German Open 2008, Hannover Messe); Rechts ein Stepfield, welches von einer Terrain-Klassifikation erkannt werden soll

2.3 Datenstrukturen für die Verarbeitung von 3-D-Laserdaten

Zur effizienten und schnellen Verarbeitung großer Mengen von 3-D-Laserdaten ist die Nutzung geeigneter Datenstrukturen unerläßlich. Da 3-D-Daten in fast allen Fällen in nicht organisierter Form vorliegen, besteht die Schwierigkeit meist darin eine beliebige Operation auf einen bestimmten Bereich im 3-D-Raum möglichst effizient anzuwenden. Dieses Problem tritt z. B. bei allen Formen der Nachbarschaftsanalyse eines Punktes auf. Zur Lösung dieser Probleme sind die im Folgenden aufgeführten Datenstrukturen geeignet.

2.3.1 Gitter

Gitter (engl. grids) können auf 3-D-Daten in Form von 2-D- und 3-D-Gittern angewendet werden. Hierzu werden die Punkte in ein 2- oder 3-dimensionales Gitter einsortiert, was den Zugriff auf einen bestimmten Bereich im 3-D-Raum von der Komplexitätsklasse $\mathcal{O}(n)$ auf $\mathcal{O}(1)$ reduziert.

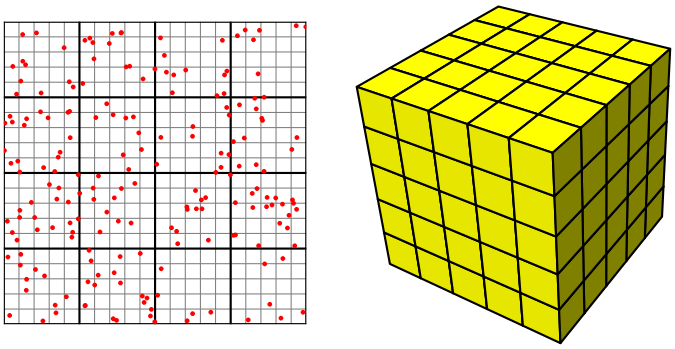


Abbildung 2.2: Links: Ein 2-D-Gitter mit einsortierten Punkten; Rechts: Ein 3-D-Gitter

2.3.2 Suchbäume

kD-Bäume

k-dimensionale Bäume (engl. kD-Trees) sind binäre Bäume, welche sich für die Suche im mehrdimensionalen Raum eignen. Hierbei wird der mehrdimensionale Raum durch das Einziehen von Ebenen oder Hyperebenen solange unterteilt bis alle Elemente, die im Baum gespeichert sind, durch diese Ebenen eindeutig voneinander abgegrenzt sind.

Oktal-Bäume

Oktal-Bäume (engl. Oct-Trees) ähneln einem 3-dimensionalen Gitter, allerdings erfolgt die Unterteilung rekursiv: Ein Würfel wird gleichmäßig in 8 Würfel zerteilt, welche wiederum in 8 Würfel zerteilt werden usw. Hierdurch entsteht eine hierarchische Baumstruktur, bei der jede Verzweigung des Baums 0 bis 8 Kinder besitzt.

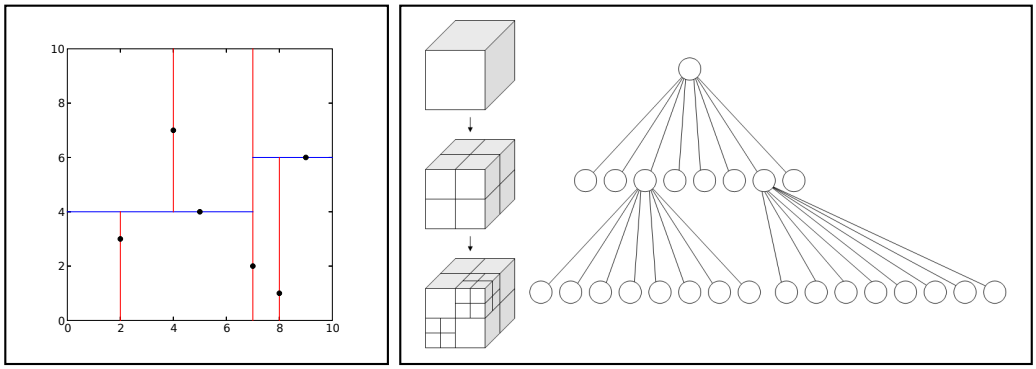


Abbildung 2.3: Links: Unterteilung eines 2-dimensionalen Raums durch einen kD-Baum mit k=2; Rechts: Oktalbaum: Teilung des Raums durch den Baum, sowie der Baum selbst. (Quelle: Wikimedia Commons)

2.3.3 Voxelisierung

Die Voxelisierung von 3-D-Daten – also die Diskretisierung einer Koordinate – wie in [LVH07] beschrieben (3-D-Gitter mit sehr hoher Auflösung und maximal einem Punkt pro Gitterzelle) ermöglicht weitere Optimierungen sowie die Wiederverwendung von Berechnungsergebnissen bei der Iteration über alle Punkte. Allerdings ergibt sich ein eventueller Verlust der Genauigkeit, wenn die Voxel-Auflösung nicht hoch genug gewählt wird.

2.4 Terrain-Klassifikation

Basierend auf den 3-D-Daten beurteilt die Terrain-Klassifikation, welche Bereiche für das mobile System befahrbar sind und welche nicht.

2.4.1 Verfahren

Höhendifferenz-basierte Verfahren

Sebastian Thrun beschreibt in [TMA+06a] eine sehr einfache Form der Klassifizierung des Terrains für das von ihm entwickelte autonome Fahrzeug „Stanley“, welches 2005 die DARPA Grand Challenge gewann. Thrun betrachtet hierbei den absoluten Höhenunterschied der Punkte in einer Gitterzelle und entscheidet daraus, ob das Gelände befahrbar ist oder nicht. Nachgelagert ist ein probabilistisches Verfahren, welches durch fehlerhaftes Scanmatching entstandene Fehler ausgleicht.

Principal Component Analysis

Aktuelle wissenschaftliche Veröffentlichungen (siehe z.B. [LVHH06, ATC+05]) nutzen die Hauptkomponentenanalyse (engl. Principal Component Analysis, PCA), welche in [Pea01] und [Hot33] beschrieben ist, zur Extraktion von Merkmalen in Form linearer Strukturen und planarer Flächen sowie 3-D-Punktwolken ohne dominante Richtung. Dieses grundsätzliche Merkmalsextraktionsverfahren wird durch den Einsatz verschiedener mathematischer Modelle zur Ausnutzung von Nachbarschaftsinformationen (Markov-Netze in [ATC+05, MVH08] sowie Gaussian Mixture Model in [LVHH06]) ergänzt.

Ausnutzung der Laseranordnung

In [MBB+08] wird die Ausnutzung der Laseranordnung zur Hinderniserkennung beschrieben. Hierbei werden von den 64 Lasern eines *Velodyne HDL-64E S2* benachbart liegende Laserstrahlen betrachtet, welche auf flachen Gelände innerhalb einer Umdrehung des Lasers einen Kreis formen. In [MBB+08] werden die Abstände zweier Laser zum Boden und der Radius, der durch sie auf den Boden projizierten Kreise betrachtet und hieraus Schlüsse über evtl. vorhandene Hindernisse und das Terrain gezogen. Das Prinzip der Ausnutzung der Laseranordnung ist grundsätzlich auf andere Schwenkvorrichtungen, wenn auch in anderer Form, übertragbar.

Kapitel 3

Eigener Ansatz

3.1 3-D-Laserscannen mittels Pan-Tilt-Roll-Einheiten

Ein 2-D-Laserscanner kann mit denen in Abb. 3.1 und 3.2 auf der nächsten Seite dargestellten Konstruktionen zur Erstellung von 3D Scans verwendet werden. Dazu werden mehrere 2-D-Scans mit unterschiedlichen Winkelstellungen der Pan-Tilt-Roll-Einheiten aufgenommen und in ein gemeinsames 3-D-Koordinatensystem transformiert. Als Vorteil erweist sich der Umstand, dass die in 3.1 auf der nächsten Seite dargestellte Konstruktion bereits für den Lageausgleich des Sensors in unebenem Gelände verwendet wird und die Schwenkvorrichtung für beide Aufgabenstellungen genutzt werden kann.

3.2 Umrechnung in kartesische Koordinaten

Für die Umrechnung der Rohdaten aus dem Laserscanner, müssen diese zuerst in kartesische 2-D-Koordinaten und dann mittels Transformationmatrizen (anhand eines Bewegungsmodells) in echte 3-D-Laserkoordinaten umgerechnet werden. Diese Umrechnung wird im folgenden Abschnitt beschrieben.

3.2.1 Generische Transformations-Matrizen

Für die folgenden Erklärungen seien die folgenden generischen Matrizen definiert:

$$\boldsymbol{R}_{\mathrm{X}}(\gamma) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\gamma) & -\sin(\gamma) & 0 \\ 0 & \sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad \text{(Rotationsmatrix um die X-Achse)}$$

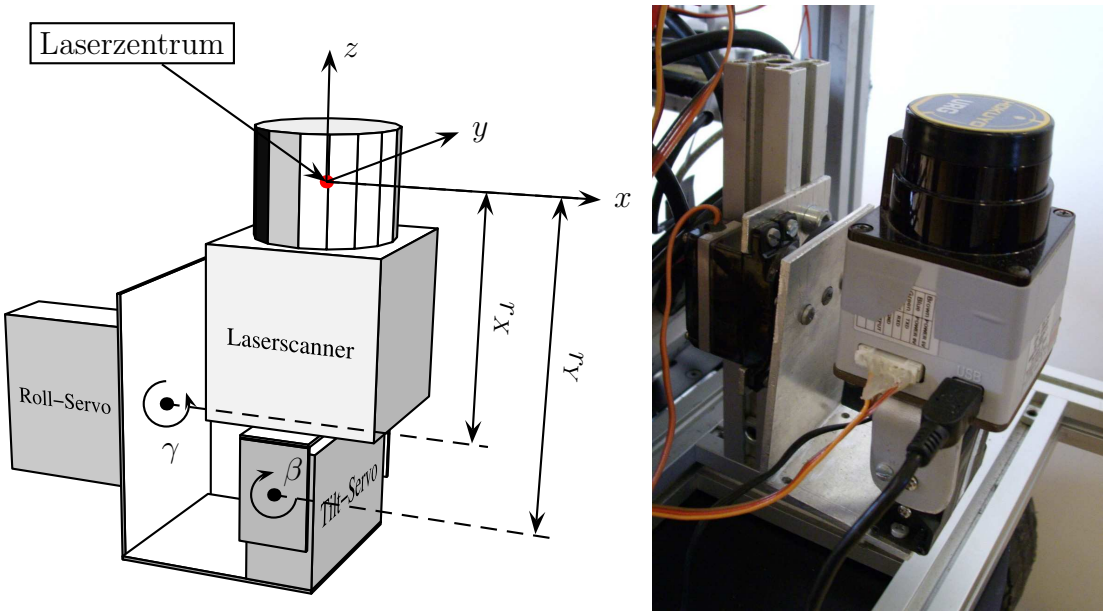


Abbildung 3.1: Aufbau eines 3-D-Laserscanners mittels einer Roll-Tilt-Einheit basierend auf Servos: Links: Schematischer Aufbau, Rechts: Foto des realen Aufbaus

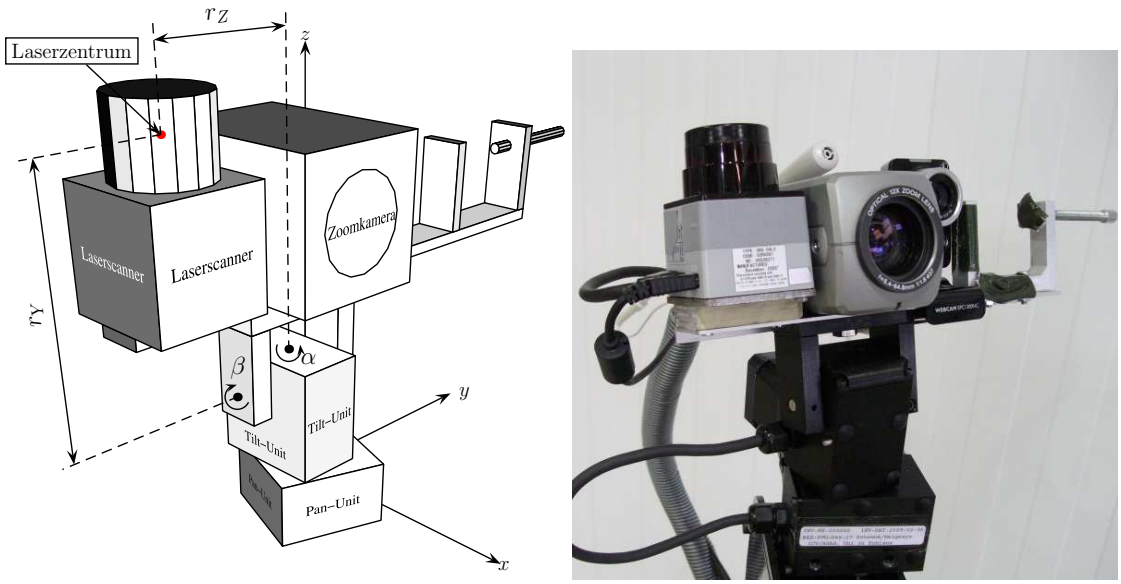


Abbildung 3.2: Aufbau eines 3-D-Laserscanners auf einer Pan-Tilt-Einheit: Links: Schematischer Aufbau, Rechts: Foto des realen Aufbaus

$$\boldsymbol{R}_{\mathrm{Y}}(\beta) = \begin{pmatrix} \cos(\beta) & 0 & \sin(\beta) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad \text{(Rotationsmatrix um die Y-Achse)}$$

$$\boldsymbol{R}_{\mathrm{Z}}(\alpha) = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) & 0 & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad \text{(Rotationsmatrix um die Z-Achse)}$$

$$\boldsymbol{T} \begin{pmatrix} t_{\mathrm{x}} \\ t_{\mathrm{y}} \\ t_{\mathrm{z}} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & t_{\mathrm{x}} \\ 0 & 1 & 0 & t_{\mathrm{y}} \\ 0 & 0 & 1 & t_{\mathrm{z}} \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad \text{(Generische Translationsmatrix)}$$

3.2.2 Umrechnung der Laserscanner Distanzen

Die Daten von Laserscannern liegen im Regelfall in Form von Polarkoordinaten vor. Diese Polarkoordinaten müssen zunächst in kartesische 2-D-Koordinaten (bezogen auf die Scanebene) umgerechnet werden um mit ihnen mittels der in Abschnitt 3.2.1 definierten Rotations- und Translations-Matrizen rechnen zu können.

Der Punkt $\boldsymbol{p}_{\mathrm{L}}$ im Koordinatensystem des Lasers (ohne Berücksichtigung jeglicher Drehungen des gesamten Scanners) berechnet sich als

$$\boldsymbol{p}_{\mathrm{L}} = \begin{pmatrix} d \cdot \cos(\phi) \\ d \cdot \sin(\phi) \\ 0 \\ 1 \end{pmatrix} \tag{3.1}$$

mit d die gemessene Distanz des Laserscanners und ϕ die Richtung der Messung in der Scannerebene.

3.2.3 Berechnung der 3D-Punktwolke

Die 3-D-Punktwolke bzw. ein 3-D-Punkt berechnet sich aus dem kartesischen Koordinaten im 2-D-Koordinatensystem des Lasers $\boldsymbol{p}_{\mathrm{L}}$ und den darauf angewandten im folgenden definierten Umkehr-Matrizen $\boldsymbol{S}_{\mathrm{rRoll}}$, $\boldsymbol{S}_{\mathrm{rTilt}}$ und $\boldsymbol{S}_{\mathrm{rPan}}$, welche die Bewegungen in den drei Achsen umkehren. Die Matrizen sind abhängig von den

Rotationsradien $r_\mathrm{X}, r_\mathrm{Y}$ und r_Z, sowie dem zum Zeitpunkt der Aufnahme eines Laserscans eingestellten Winkel. Die Winkel-Bezeichnungen und Radien sind in Abb. 3.1 und 3.2 auf Seite 20 eingezeichnet.

$$\boldsymbol{S}_\mathrm{rRoll}(\beta, r_\mathrm{X}) = \boldsymbol{T}\begin{pmatrix}0\\0\\-r_\mathrm{X}\end{pmatrix} \cdot \boldsymbol{R}_\mathrm{X}(-\beta) \cdot \boldsymbol{T}\begin{pmatrix}0\\0\\+r_\mathrm{X}\end{pmatrix} \tag{3.2}$$

$$\boldsymbol{S}_\mathrm{rTilt}(\gamma, r_\mathrm{Y}) = \boldsymbol{T}\begin{pmatrix}0\\0\\-r_\mathrm{Y}\end{pmatrix} \cdot \boldsymbol{R}_\mathrm{Y}(-\gamma) \cdot \boldsymbol{T}\begin{pmatrix}0\\0\\+r_\mathrm{Y}\end{pmatrix} \tag{3.3}$$

$$\boldsymbol{S}_\mathrm{rPan}(\alpha, r_\mathrm{Z}) = \boldsymbol{T}\begin{pmatrix}0\\+r_\mathrm{Z}\\0\end{pmatrix} \cdot \boldsymbol{R}_\mathrm{Z}(-\alpha) \cdot \boldsymbol{T}\begin{pmatrix}0\\-r_\mathrm{Z}\\0\end{pmatrix} \tag{3.4}$$

Die Translationsmatrizen in den vorhergehenden Formeln sind notwendig, da sich der Laserscanner um eine andere Achse bewegt als das Zentrum des Laserstrahls. Die Bewegung des Laserscanners selbst muss mittels dieser Matrizen kompensiert werden, um korrekte Messergebnisse zu erhalten. Da die Rotationsmatrizen im Fall $\alpha = \beta = \gamma = 0$ der Einheitsmatrix entsprechen, müssen sie jeweils von zueinander inversen Translationsmatrizen umschlossen sein, da im Falle $\alpha = \beta = \gamma = 0$ folgendes gelten muss: $\boldsymbol{p}_\mathrm{L} = \boldsymbol{p}_\mathrm{L3D}$.

Durch Multiplikation der Umkehr-Matrizen in der Reihenfolge des Bewegungsmodells, d.h. Umkehren der Bewegungen des Laserscanners ergibt sich dann aus dem in Unterabschnitt 3.2.2 auf der vorherigen Seite definierten Punkt $\boldsymbol{p}_\mathrm{L}$, der hierzu gehörige Punkt $\boldsymbol{p}_\mathrm{L3D}$ im 3-D-Raum mit dem Laserzentrum als Ursprung des Koordinatensystems:

$\boldsymbol{p}_\mathrm{L3D} = \boldsymbol{S}_\mathrm{rRoll} \cdot \boldsymbol{S}_\mathrm{rTilt} \cdot \boldsymbol{p}_\mathrm{L}$ (Umkehrung der Bewegung aus Abb. 3.1 auf Seite 20)

$\boldsymbol{p}_\mathrm{L3D} = \boldsymbol{S}_\mathrm{rPan} \cdot \boldsymbol{S}_\mathrm{rTilt} \cdot \boldsymbol{p}_\mathrm{L}$ (Umkehrung der Bewegung aus Abb. 3.2 auf Seite 20)

3.3 Kalibrierung von Servos mittels Laserscanner

3.3.1 Servo-Steuerung

Servos werden durch Anlegen eines Pulsweiten modulierten Signals (PWM-Signal) gesteuert. Ein PWM-Signal entspricht einem elektrischen Rechtecksignal, bei welchem das Verhältnis oder die Zeitspanne der High-/Low-Phase die Steuerungs-Information enthält. Bauartbedingt verhalten sich jedoch alle Servos, selbst solche

des gleichen Typs, bei identischem PWM-Signal unterschiedlich, d.h. sie stellen sich auf unterschiedliche Winkel ein.

Um mit einem 2-D-Laserscanner, welcher durch Servos geschwenkt wird, hinreichend genaue 3-D-Laserscans durchführen zu können, ist es von entscheidender Wichtigkeit, den Stellwinkel eines Servos bei allen anlegbaren PWM-Signalen zu kennen, um eine möglichst exakte Transformation der 2-D-Scans in ein gemeinsames 3-D-Koordinatensystem zu gewährleisten. Dies lässt sich mit Hilfe einer automatischen Kalibrierung der Servo-Winkel in akzeptabler Zeit realisieren. Das Ergebnis ist eine Zuordnung zwischen gültigen PWM-Signalen und den daraus resultierenden Stellwinkeln.

Diese Zuordnung lässt sich in aller Regel als eine einfache lineare Funktion darstellen, wobei α in der folgenden Formel den Stell-Winkel symbolisiert.

$$p(\alpha) = m\alpha + b \tag{3.5}$$

Da es sich um eine lineare Funktion handelt muss nicht zu jedem PWM-Signal der Winkel ermittelt werden, sondern es genügen mehrere Stichproben um m und b hinreichend genau zu bestimmen.

3.3.2 Kalibrierung

Die im Folgendem beschriebene Kalibrierungsmethode setzt voraus, dass vor oder seitlich (je nach zu kalibrierender Achse) des Laserscanners genügend *ebener und freier* Boden vorhanden ist. Zur eigentlichen Kalibrierung werden mehrere gültige PWM-Signale an den zu kalibrierenden Servo angelegt und nach einer festzulegenden Zeitspanne, nach welcher sich der Servo auf das Signal eingestellt hat, eine oder evtl. mehrere Messungen (die dann zur Erhöhung der Genauigkeit gemittelt werden) mit dem Laserscanner vorgenommen. Aus den Daten eines Rundumscans des Laserscanners, also

- $d_i, i = 1...n$ die gemessenen Abstände des Laserscanners
- ϕ_i die Richtungen der Messungen innerhalb der Scanebene

wird dann der gemittelte Abstand mehrerer Laserstrahlen zum Boden bestimmt:

$$d_{\text{ground}} = \frac{1}{2k+1} \sum_{i=i_0-k}^{i_0+k} \cos(\phi_i)\ d_i \tag{3.6}$$

mit $k \in \mathbb{N}$ und $\phi_{i_0} \approx 0°$ (Tilt-Servo) oder $\phi_{i_0} \approx \pm 90°$ (Roll-Servo)

Die Einschränkung von i auf 2k-Werte um den Winkel ϕ_{i_0} ist notwendig, da Messungen des Laserscanners, welche nicht auf den Boden gerichtet sind, nicht

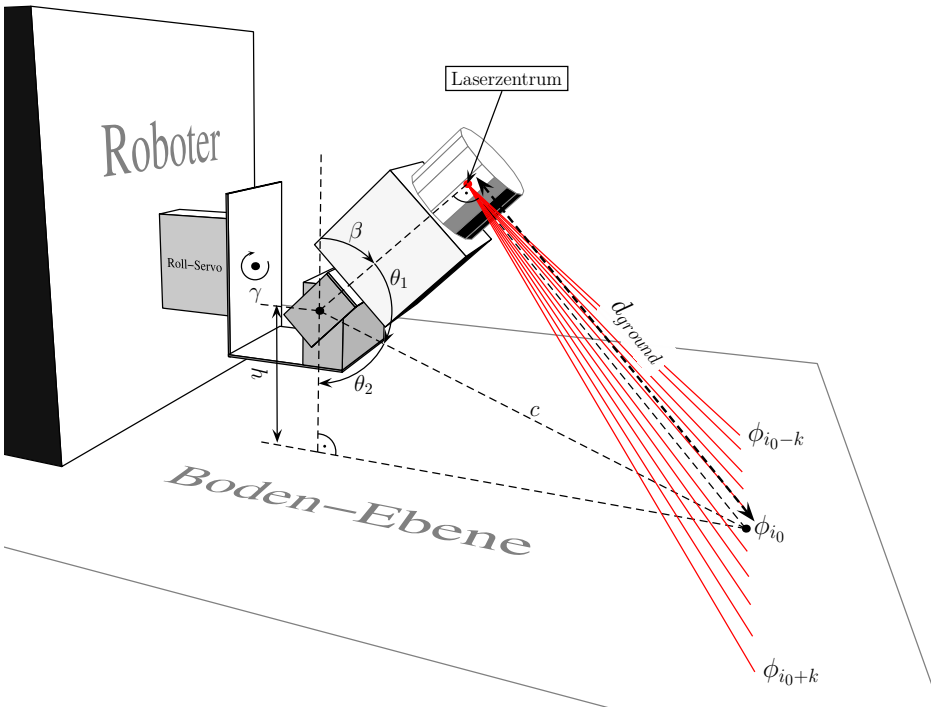


Abbildung 3.3: Skiziierte Darstellung des Kalibrierungsverfahrens für den „Tilt-Servo“

berücksichtigt werden dürfen. Außerdem lässt sich hiermit der für die Kalibrierung notwendige ebene und freie Boden einschränken. Für $k = 0$ wird z.B. nur ein Laserstrahl zur Kalibrierung verwendet, für $k = 1$ drei Laserstrahlen usw. (siehe auch Abb. 3.3 zur Erklärung).

Aus d_{ground} lässt sich jetzt, da der Aufbau des Modells (h = Abstand zwischen Drehachse des Servos und Boden, r_{Y} = Abstand des Laserursprungs zur Drehachse) bekannt ist, der Stellwinkel des Servos berechnen. Es ergeben sich aus Abb. 3.3 und Abb. 3.1 auf Seite 20 die folgenden Formeln zur Berechnung des Stellwinkels mit dem in Formel 3.6 auf der vorherigen Seite ermitteltenden Wert d_{ground} als Eingabe:

$$c = \sqrt{d_{\text{ground}}^2 + r_{\text{Y}}^2} \tag{3.7}$$

$$\beta = 180^\circ - \theta_1 - \theta_2 \tag{3.8}$$

$$= 180^\circ - \operatorname{acos}\left(\frac{r_{\text{Y}}}{c}\right) - \operatorname{acos}\left(\frac{h}{c}\right) \tag{3.9}$$

Die Kalibrierung des Winkels γ erfolgt analog, mit dem einzigen Unterschied, dass nicht der Abstand zum Boden nach vorne bestimmt wird, sondern der seitliche Abstand zum Boden.

Abbildung 3.4: Ein Stepfield bei der Robocup Rescue Weltmeisterschaft in Graz (2009)

3.4 Terrainklassifikation in RoboCup Rescue

RoboCup Rescue ist ein Szenario, in welchem Roboter in einer abstrahierten Umgebung, welche ein eingestürztes Gebäude simulieren soll, verschüttete Menschen orten sollen. Die Schwierigkeit besteht nicht nur in der Lokalisation, Kartierung des Gebäudes und Erkennung der Menschen, sondern auch in der Erkennung von für den Roboter befahrbaren und nicht befahrbaren Gebieten, was die Thematik des folgenden Abschnitts ist. In RoboCup Rescue müssen hauptsächlich Stepfields (Abb. 3.4) und Abgründe (Abb. 3.6 auf Seite 31) erkannt werden, da sie mit der verwendeten Plattform nicht befahren werden können. Die folgenden Verfahren basieren alle darauf, dass 3-D-Scans mit der im vorhergehenden Kapitel beschriebenen Vorrichtung aus Servos angefertigt werden und zwar speziell durch Kippen/Nicken des „Tilt-Servo“ um den Winkel β.

3.4.1 Klassifikation mittels Höhenvarianz

Ein einfaches Maß für die Befahrbarkeit einer planaren Fläche ist die Bildung der Varianz auf die Höhe bezogen. Flache Flächen haben eine niedrige Varianz, sehr unebene, raue Flächen oder gar Wände eine hohe. Die Varianz lässt sich mit Hilfe des Verschiebungssatzes effizient ohne vorherige Mittelwertbildung berechnen. Für

eine Menge von n Punkten $\boldsymbol{p_i} \in \mathbb{R}^3$ sei die Höhenvarianz V_H dieser Punktmenge wie folgt definiert:

$$\boldsymbol{p_i} = \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} \tag{3.10}$$

$$V_\mathrm{H}(\boldsymbol{p_i}) = \frac{1}{n-1} \left(\sum_{i=1}^{n} z_i^2 - \frac{1}{n} \left(\sum_{i=1}^{n} z_i \right)^2 \right) \tag{3.11}$$

Ablauf der Klassifikation

Der Ablauf der Verarbeitung und Klassifikation eines 3-D-Laserscans mittels Höhenvarianz ist wie folgt:

1. Vorverarbeitung des Tiefenbilds mit einem Medianfilter
2. Umrechnung der Laserscannerdistanzen in 3-D-Laserkoordinaten
3. Einsortieren der Punkte in ein Gitter
4. Berechnen des absoluten Höhenunterschieds und der Varianz pro Gitterzelle
5. Klassifikation mittels Varianz und Höhenunterschied
6. Nachfilterung des Ergebnisses mit einem Erosionsfilter
7. Eintragung der unbefahrbaren Stellen in die Navigationskarte

Vorverarbeitung des Tiefenbilds mit einem Medianfilter

Um einzelne Messfehler, Ausreißer und Scans durch schmale Spalten im Boden, wie sie in RoboCup Rescue oft auftreten, zu elimieren, wird noch vor der Umrechnung in 3-D-Laserkoordinaten das durch mehrere Scans entstandene Tiefenbild einer Median-Filterung unterzogen. Ein 3×3-Medianfilter erzielt normalerweise schon die erwünschte Wirkung.

Einsortieren der Punkte in ein Gitter

Nach der Vorfilterung und der Umrechnung in 3-D-Laserkoordinaten werden die Punkte in ein 2-dimensionales Gitter eingefügt, welches auf der XY-Ebene liegt und die Orientierung der Koordinatenachsen, die gleiche wie in Abb. 3.1 auf Seite 20 ist. Für RoboCup Rescue, sowie die Navigation im Indoor-Bereich hat sich eine Zellengröße von $10 \times 10\,\mathrm{cm}$ als sinnvoll erwiesen.

Berechnen der Höhendifferenz und Varianz pro Gitterzelle

Für jede Gitterzelle wird die Höhenvarianz $V_{\mathrm{H}}(\boldsymbol{p_i})$ und die maximale Höhendifferenz $D_{\mathrm{H}}(\boldsymbol{p_i})$, der n Punkte einer Gitterzelle berechnet. Die maximale Höhendifferenz ist dabei wie folgt definiert:

$$\boldsymbol{p_i} = \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} \tag{3.12}$$

$$D_{\mathrm{H}}(\boldsymbol{p_i}) = \max(|z_i - z_j|) \text{ für } i, j \in \{1 \dots n\} \tag{3.13}$$

Um den Einfluss von Ausreißern, welche von der Median-Filterung nicht elimiert wurden, zu verringern, wird die Varianz nur auf einem Teil der Punkte einer Gitterzelle berechnet. Hierzu werden die Punkte der Höhe nach sortiert und jeweils ein bestimmter Prozentsatz der höchsten und niedrigsten Punkte für die Varianzbildung nicht berücksichtigt.

Klassifikation

Überschreiten sowohl die maximale Höhendifferenz der Gitterzelle als auch die Varianz einen festgelegten Schwellwert, so wird die Gitterzelle vorläufig als nicht befahrbar markiert.

Nachfilterung mit einem Erosionsfilter

Um die Empfindlichkeit des Klassifikators weiter zu senken wird auf das Resultat der Klassifikation zum Schluss noch ein Erosionsfilter angewandt, welcher isolierte Hinderniszellen eliminiert. Befindet sich also in der 8'er Nachbarschaft einer Hinderniszelle keine weitere als nicht befahrbar markierte Zelle, so wird die Zelle nachträglich als befahrbar markiert.

Eintragung der unbefahrbaren Stellen in die Karte

Die Mittelpunkte der als nicht befahrbar markierten Gitterzellen werden in der Navigationskarte des Roboters als unbefahrbar markiert. Hierbei sollte die Position und die Anzahl der Punkte in einer Gitterzelle mit berücksichtigt werden. Zu weit entfernte Gitterzellen sollten nicht als unbefahrbar markiert werden, da hier die Genauigkeit des Laserscanners und der Schwenkvorrichtung massiv abnimmt. Ebenso sollten keine Zellen als unbefahrbar markiert werden, in denen sich nicht eine Mindestanzahl von Punkten befindet.

3.4.2 Principle Component Analysis basiert

Ein häufig in der Literatur beschriebenes Verfahren (siehe z. B. [LVHH06, ATC+05]) folgt der Idee alle Punkte des Laserscans in ein Gitter einzusortieren und im Anschluss eine *Principle Component Analysis* (PCA) auf den Kovarianzmatrizen der in den einzelnen Gitterzellen enthaltenen Punkte durchzuführen. Die 3×3 Kovarianzmatrix $\mathbf{C}$ einer Menge von n Punkten $\boldsymbol{x}_i \in \mathbb{R}^3$ ergibt sich mit Hilfe des Verschiebungssatzes in einem einzigem Durchlauf der Daten aus:

$$\mathbf{C} = \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{x}_i \cdot \boldsymbol{x}_i^{\mathrm{T}} - \frac{1}{n^2} \sum_{i=1}^{n} \boldsymbol{x}_i \cdot \sum_{i=1}^{n} \boldsymbol{x}_i^{\mathrm{T}} \tag{3.14}$$

$$= \begin{pmatrix} \mathrm{Cov}(X,X) & \mathrm{Cov}(Y,X) & \mathrm{Cov}(Z,X) \\ \mathrm{Cov}(X,Y) & \mathrm{Cov}(Y,Y) & \mathrm{Cov}(Z,Y) \\ \mathrm{Cov}(X,Z) & \mathrm{Cov}(Y,Z) & \mathrm{Cov}(Z,Z) \end{pmatrix} \tag{3.15}$$

Die Eigenvektoren $\boldsymbol{e}_i$ mit den zugehörigen Eigenwerten λ_i erhält man aufgrund der Symmetrie von $\mathbf{C}$ durch einen Spezialfall der Eigendekomposition:

$$\mathbf{C} = \begin{bmatrix} \boldsymbol{e}_0 & \boldsymbol{e}_1 & \boldsymbol{e}_2 \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} \lambda_0 & 0 & 0 \\ 0 & \lambda_1 & 0 \\ 0 & 0 & \lambda_2 \end{bmatrix} \begin{bmatrix} \boldsymbol{e}_0 & \boldsymbol{e}_1 & \boldsymbol{e}_2 \end{bmatrix} \quad \text{mit} \quad \lambda_0 \geq \lambda_1 \geq \lambda_2 \tag{3.16}$$

Eine Analyse der Eigenwerte und -vektoren der Kovarianzmatrix gibt Aufschluss über die Struktur der in einer Gitterzelle enthaltenen Punktwolke. Die Eigenvektoren, welche orthogonal zueinander sind, entsprechen den Hauptachsen der Punktwolke, geben also die Richtungen an, in denen die Varianz der Punktwolke am größten, zweitgrößten und drittgrößten ist. Die Eigenwerte entsprechen hierbei den Varianzen in Richtung der Eigenvektoren. Aufgrund der Eigenwerte und Vektoren lassen sich also Aussagen über die Form der Punktwolke treffen, wie in Abb. 3.5 auf der nächsten Seite dargestellt. Die Unebenheit u einer Gitterzelle wird definiert als der Eigenwert, dessen dazugehöriger Eigenvektor am ehesten nach oben zeigt. Also

$$u = \lambda_k \quad \text{mit} \quad k = \underset{i \in \{0,1,2\}}{\mathrm{argmax}} \left| \boldsymbol{e}_i^{\mathrm{T}} \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^{\mathrm{T}} \right| \tag{3.17}$$

Liegt der Wert u einer Zelle über einem gewissen Grenzwert, so wird die Zelle komplett als Hindernis markiert. Liegt u unter diesem Grenzwert, wird angenommen, dass die Punkte in der entsprechenden Zelle mehr oder weniger planar angeordnet sind. Die am besten approximierende Ebene der Punkte definiert sich dann durch den als Normale fungierenden Eigenvektor $\boldsymbol{e}_k$ sowie den Schwerpunkt der Punkte. Zusätzlich wird auch noch der Winkel δ mit in die Bewertung mit

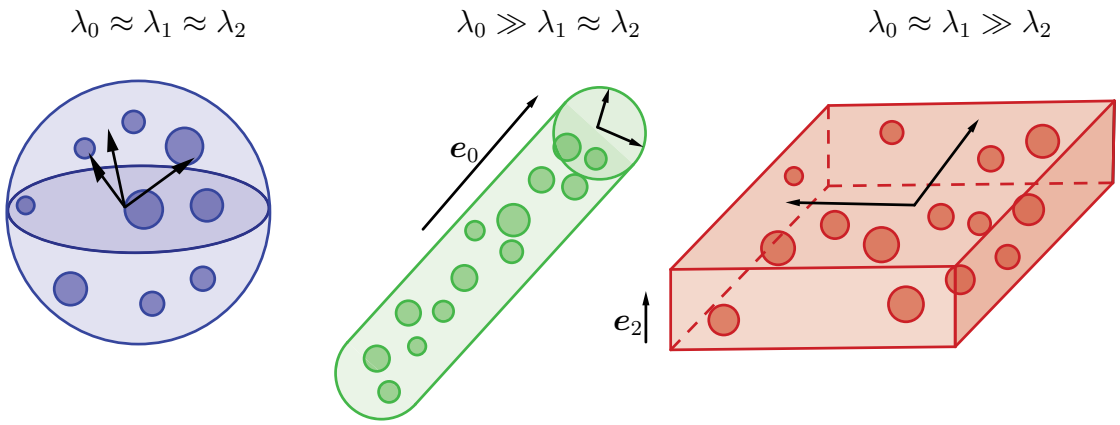


Abbildung 3.5: Visualisierung der Eigenwerte und -vektoren verschiedener Punktwolken

einbezogen. Dieser darf einen bestimmt Schwellwert nicht überschreiten, da die potentiell befahrbare Fläche ansonsten zu steil wäre.

$$\delta = \operatorname{acos}\left(\frac{|\boldsymbol{e}_k^{\mathrm{T}} \begin{bmatrix}0 & 0 & 1\end{bmatrix}^{\mathrm{T}}|}{|\boldsymbol{e}_k^{\mathrm{T}}|}\right) \tag{3.18}$$

Als problematisch erweisen sich weit entfernte Regionen, die zwischen zwei Laserscans liegen. Hier treffen keine Laserpunkte auf – und somit gibt es auch keine Punktwolke die mit Hilfe der PCA analysiert werden kann. Idealerweise würde man in diesen Bereichen die Größe der Zellen erhöhen. Dies ist aber nicht unproblematisch: Befindet sich nur in einem kleinen Teil einer Gitterzelle ein Hindernis, so muss immer die ganze Zelle, in der sich ein solches Hindernis befindet als nicht befahrbar markiert werden. Somit werden zu große Bereiche als nicht befahrbar markiert.

Um die Vorteile kleiner und großer Zellengröße zu kombinieren wird ein *hierarchisches Verfahren mit adaptiver Zellengröße* verwendet, welches unter anderem auch in [NDPP09] beschrieben ist. Kern des Verfahrens ist eine rekursive Funktion welche auf einem größeren Bereich – einer Makrozelle – des bestehenden Gitters operiert. Die Makrozelle umfasst dabei mehrere normale Zellen. In jedem Schritt wird die PCA aller Punkte berechnet, die in der betrachteten Makrozelle liegen. Wie im vorher beschriebenen Verfahren wird die Unebenheit u der Makrozelle untersucht. Liegt u unter einem Grenzwert, werden alle in der Makrozelle liegenden Zellen als planar und zu einer gemeinsamen Ebene gehörig betrachtet. Wie vorher wird eine Ebene durch diese Punkte gelegt. Liegt u über dem Grenzwert, so wird die Makrozelle mittig quer zur längsten Achse geteilt und die Funktion ruft sich zweimal mit diesen neuen, disjunkten Makrozellen rekursiv auf.

Pseudocode:

```
function recursivePCA(area)
    compute u from the points in 'area';
    if (u<threshold)
        mark all cells in 'area' as drivable;
        assign the computed plane to the cells;
    else
        if (size(area)==1x1)
            mark cell in 'area' as obstacle;
        else
            split 'area' into 2 parts;
            recursivePCA(subarea_0);
            recursivePCA(subarea_1);
        end
    end
end
```

Ergebnis dieser Aufspaltungsstrategie ist eine feinere Unterteilung der Gitterzellen in der Nähe von Hindernissen und ein Zusammenfassen von Zellen in großen, relativ flachen Bereichen. In der Nähe von weit entfernten Hindernissen, wo die Unterteilung feingranularer wird und gleichzeitig nur eine relativ dünne Abdeckung mit Laserstrahlen vorhanden ist, kann es allerdings passieren, dass eine Zelle wieder so klein wird, dass gar keine Punkte mehr hinein fallen. Zu diesen Bereichen kann dann wieder keine Aussage zur Befahrbarkeit getroffen werden.

Das Zusammenfassen großer, weit entfernter Bereiche geschieht teils unter der impliziten Annahme dass sich zwischen zwei übereinanderliegenden Laserringen kein tiefes negatives Hindernis befindet. Diese Annahme ist natürlich im Allgemeinen nicht korrekt. Da der Laserscanner allerdings aus großer Entfernung ohnehin kaum *in* Löcher hineinblicken kann, und somit das Loch eigentlich gar nicht als Loch wahrnehmen kann, ist die Klassifikation als negatives Hindernis aus großer Distanz sowieso äußerst schwierig. Nähert man sich dem Loch jedoch, kann der Laser immer weiter in das Loch hineinblicken und schließlich würde es auch von dem vorgestellten Verfahren als (negatives) Hindernis erkannt. Dies trifft jedoch nur auf Löcher mit rundum existierendem und für den Laserscanner erkennbarem Rand zu. Abgründe können derzeit von der PCA in der hier dargestellten Form nicht als Hindernis erkannt werden. Allerdings werden Abgründe aufgrund der fehlenden Lasermess-Punkte am Abgrund auch nicht als befahrbar erkannt.

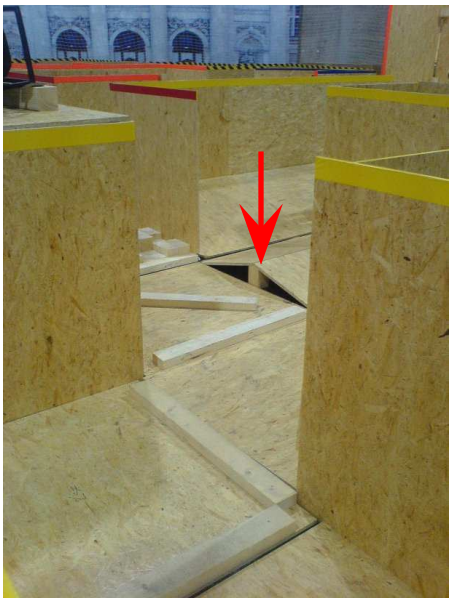


Abbildung 3.6: Ein Abgrund/eine Klippe im zu explorierenden Labyrinth (Graz, 2009)

3.4.3 Kantendetektion im Tiefenbild

Ein völlig andere Methode für die Klassifikation von Terrain, welche nicht auf der Verarbeitung der 3-D-Punktwolke basiert, ist die Analyse des Tiefenbildes mit Mitteln der Bildverarbeitung (vgl. [Jäh05]). Dass die Analyse des Tiefenbilds möglich ist und dass ein solches sinnvolles überhaupt existiert, ist der Art, wie die Scans angefertigt werden, geschuldet. Die Art der Erstellung wurde in Abschnitt 3.4 auf Seite 25 beschrieben.

Die Analyse des Tiefenbildes eignet sich besonders für die Erkennung von negativen Hindernissen, also Löchern, Abgründen wie z.B. in Abb. 3.6 zu sehen. Diese sind aufgrund der Verdeckung des Abgrunds (aus der Sicht des Roboters) und der sehr ungleichen Punktdichte im Scan nur sehr schwer in der 3-D-Punktwolke zu erkennen, jedoch bei entsprechender Stellung des Roboters recht deutlich im Tiefenbild.
Das Tiefenbild wird dazu wie folgt verarbeitet:

Median-Filterung

Zuerst wird das Bild mit einem 3×1-Medianfilter horizontal gefiltert um einzelne Fehlmessungen und Ausreißer zu elimieren. Da der Filter kantenerhaltend ist wird die nachfolgende Kantendetektion nicht beeinflusst.

Sobel-Filterung

Das Bild wird mit Hilfe des Sobel-Operators G_x in X-Richtung abgeleitet. Auf die Y-Richtung wird aufgrund der Ungenauigkeiten des Servos und daraus evtl. entstehender falscher Kanten verzichtet.

$$G_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} \qquad \text{(Horizontaler Sobel-Operator)}$$

Befindet sich ein Messfehler in der Maske des Sobeloperators, d.h. ein Distanzwert $\leq 20\,\text{mm}$, so wird das Ergebnis der Sobel-Operation nachträglich auf 0 gesetzt um falsche Kanten zu eliminieren. Distanzwerte $\leq 20\,\text{mm}$ entsprechen den Fehlercodes des verwendeten Laserscanners.

Erneute/Zweite Ableitung

Da fließende Grauwertübergänge in dem Tiefenbild, welches verarbeitet wird, die Regel sind, diese allerdings nicht von Relevanz sind, wird das Bild mit Hilfe der symmetrischen Differenz erneut abgeleitet, um diese gleichmäßigen Übergänge zu elimieren, sodass nur markante Änderungen im Grauwertverlauf einer Bildzeile erhalten bleiben.

$$\begin{bmatrix} +1 & 0 & -1 \end{bmatrix} \qquad \text{(Symmetrische Differenz)}$$

Die falschen Kanten aus der Sobel-Filterung müssen ebenfalls berücksichtigt werden. Im Falle, dass eine solche nachträglich elimierte Kante sich in der Maske befindet, so wird das Ergebnis der symmetrischen Differenz ebenfalls nachträglich wieder auf 0 gesetzt.

Binarisierung, Kanten ausdünnen und Erosion

Das Bild wird mittels eines einzustellenden Schwellwerts binarisiert und die durch die Ableitungen gespreizten Kanten werden auf maximal 1 Pixel Breite in horizontaler Richtung ausgedünnt. Einzelne isolierte Pixel werden gelöscht.

Markierung der Kante und Eintragung in die Karte

Nach der oben beschriebenen Verarbeitung des Tiefenbilds werden jeweils die Pixelpaare in horizontaler Richtung, welche eine Kante bilden, im Originalbild markiert und die entsprechenden 3-D-Positionen berechnet. Der höhere Punkt eines Kanten-Pixelpaares wird mit seiner X- und Y-Koordinate in die Karte als Hindernis eingetragen.

Der Grund dafür, dass der höhere Punkt der Kante als Hindernis eingetragen wird, ist der folgende: Erstens ist es intuitiv bei einem Abgrund den letzten Punkt

vor dem Abgrund als Hindernis zu markieren. Zweitens kommt es im Tiefenbild vor, dass es in einer Bildzeile Kantenübergänge von Wänden zu Böden gibt, wobei man in diesem Fall nicht den Bodenpunkt als Hindernis markieren möchte, sondern den Wandpunkt.

Wie bei den anderen Verfahren auch ist es sinnvoll den Bereich, in welchem man Punkte als unbefahrbar markiert, einzuschränken. Da der Laserscanner ein Sichtfeld von 240° besitzt erhält man auch Kanten und Punkte hinter dem Laserscanner und Roboter. Weil die zugehörigen Laserstrahlen in diesem Fall aber nach oben zeigen ist es nicht sinnvoll diese zu verwenden.

Kapitel 4

Experimente und Ergebnisse

4.1 3-D-Laserscannen mittels Pan-Tilt-Roll-Einheiten

Mit der in Abb. 3.1 auf Seite 20 dargestellten Vorrichtung lassen sich nun – je nach Anwendungsfall – durch beliebiges Bewegen der Servos unterschiedlichste Arten von 3-D-Scans durchführen. Gleichzeitig kann die Vorrichtung zum Lageausgleich benutzt werden. Drei verschiedene Scans, die durch verschiedene Bewegungen der Servos erstellt wurden, sind in den Abbildungen 4.1 und 4.2 zu sehen. Bei genauerer Betrachtung kann man die verschiedenartige Ausbreitung der Laserpunkte erkennen.

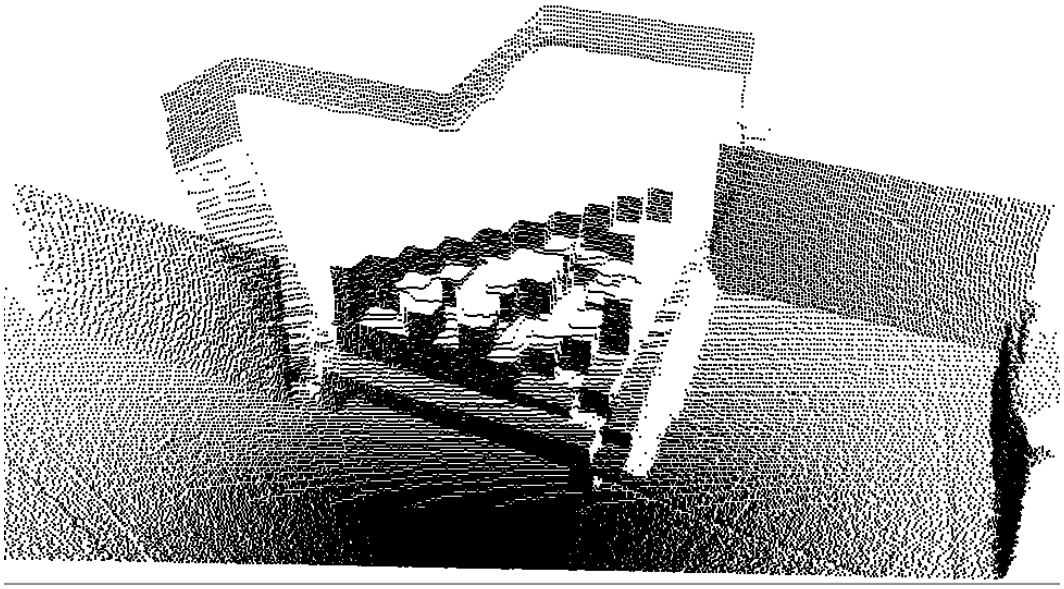
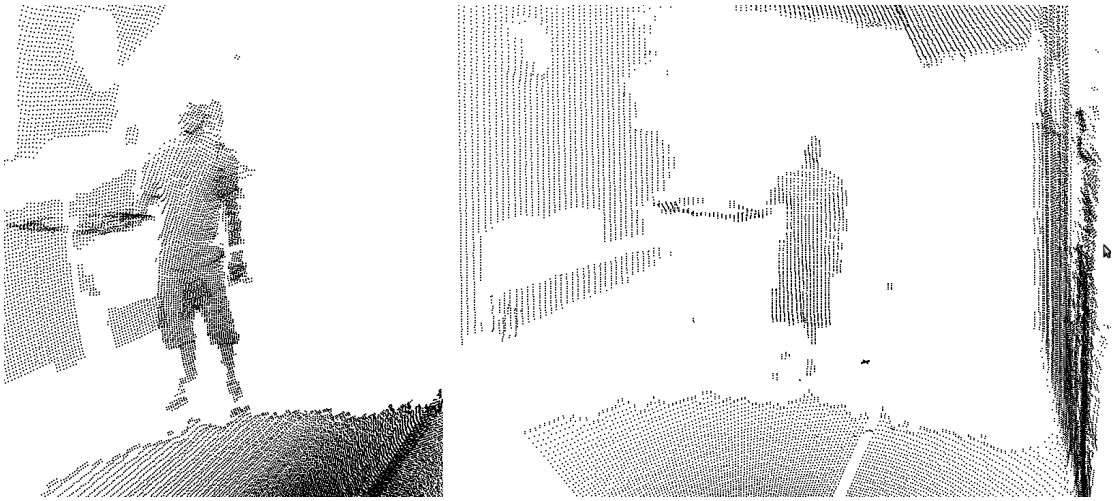


Abbildung 4.1: Ein 3-D-Scan eines Stepfields durch Nickbewegung des Tilt-Servo

Abbildung 4.2: 3-D-Scans einer Person: Links: Ein Scan durch Drehbewegung des Roll-Servo; Rechts: Ein Scan durch Kippen des Roll Servo auf $\gamma = \pm 90°$ und Schwenken des Tilt-Servos.



4.2 Servokalibrierung

Das Ergebnis einer durchgeführten Servokalibrierung ist in Abb. 4.3 auf der nächsten Seite zu sehen. Bei dem durchgeführten Experiment wurden an den Servo alle generierbaren PWM-Signale von ca. 3475 bis 4375 (entsprechend 1,88 - 2,73 ms High-Phase im PWM-Signal) angelegt. Die Zahlen der X-Achse in Abbildung 4.3 geben die Zeitdauer der High-Phase eines PWM-Signals an, welche von der Vorteilrate des Zählers und der Taktrate abhängen, mit welcher der Mikrokontroller betrieben wird, der das PWM-Signal generiert. Der Wert konkret gibt die Dauer der High-Phase des PWM-Signals in einzelnen Taktschritten an.

4.2.1 Umrechnung des PWM-Werts

Der verwendete Mikrokontroller zur Erzeugung des PWM-Signals ist ein *Atmega 8* der Firma [1]*Atmel*, welcher auf der Experimentierplatine *Crumb8-USB* der Firma [2]*chip45.com* verbaut ist. Dieser Mikrokontroller wird von einem Quartz mit 14,7456 MHz getrieben. Der interne Zähler des Mikrokontrollers, welcher das PWM-Signal erzeugt wird mit einem Vorteilung des Taktes von 8 betrieben, also

[1] `http://www.atmel.com/`
[2] `http://www.chip45.com/`

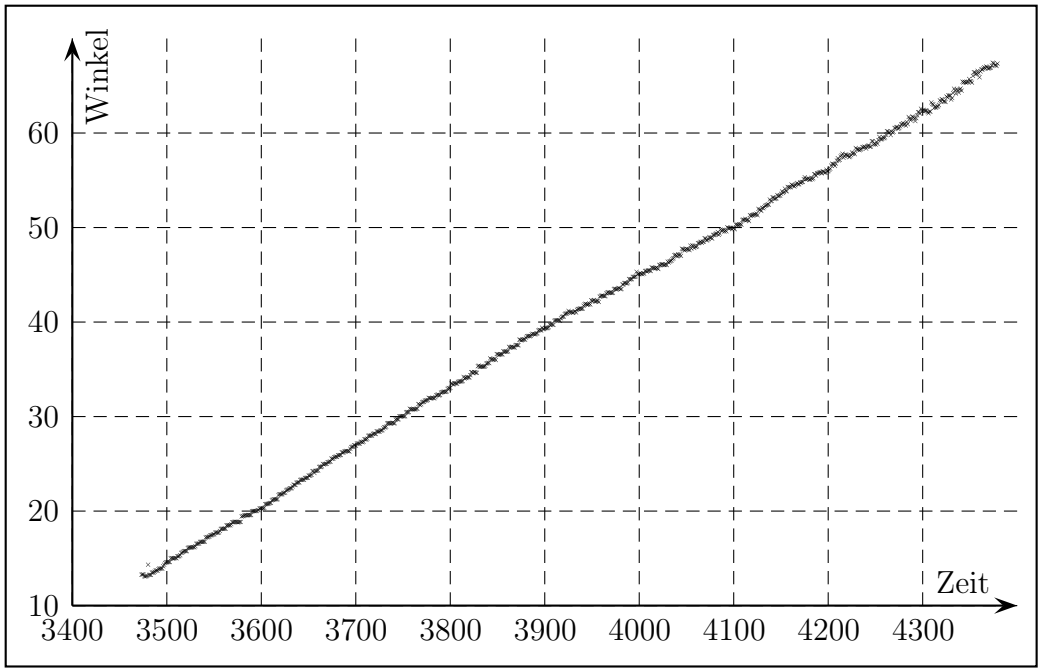


Abbildung 4.3: Vollständiges Ergebnis einer Kalibrierung:
Zuordnung der Dauer der High-Phase des PWM-Signals zu den dazugehörigen Winkeln

mit 1,8432 MHz. Die unnormierten PWM-Werte beziehen sich auf diese Taktrate. Ein PWM-Wert p lässt sich also sehr einfach in Sekunden umrechnen:

$$t(p) = \frac{p}{1843200} \tag{4.1}$$

Ein PWM-Wert von 1843 Takten entspricht also einer Dauer der High-Phase des PWM-Signals von ca. 1 ms.

4.2.2 Folgerungen

Aus den Daten des Plots in Abb. 4.3 lässt sich ein Wert von $17,3913$ Takten pro Grad Winkeländerung ableiten. Dies entspricht $0,009435$ ms pro Grad.

In der vergrößerten Abb. 4.4 auf der nächsten Seite lassen sich außerdem die Grenzen der Stellgenauigkeit des Servos erkennen. Es ist ein deutlicher Treppenstufeneffekt bedingt durch den toleranten Regler des Servos sichtbar. Der Servo bewegt sich nur, wenn das PWM-Signal mindestens um ca. 7-8 Takte (bezogen auf die Taktrate des Zählers in Unterabschnitt 4.2.1 auf der vorherigen Seite) verringert oder vergrößert wird. 8 Takte wiederum entsprechen $\frac{8}{17.3913} \approx 0,46°$. Veränderungen des Winkels des Servos unterhalb dieser Schwelle sind also nicht möglich.

Mit daraus resultierenden Fehlern bei der 3D-Punktberechnung muss man also rechnen.

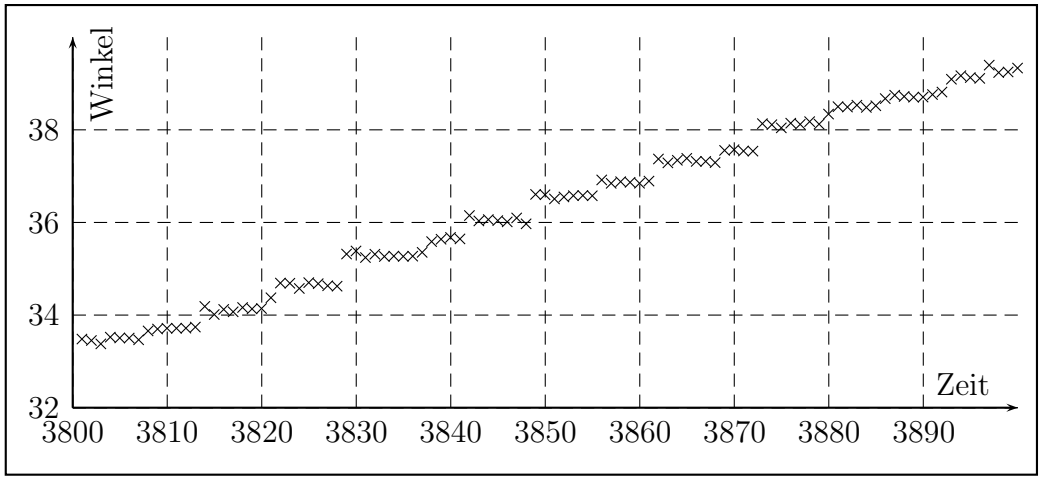


Abbildung 4.4: Treppenstufen-Effekt bei der Steuerung des Servos per PWM-Signal

4.2.3 Genauigkeitbetrachtung des Servos

Da die Qualität aller Klassifikationsverfahren direkt von der Qualität der 3-D-Scans abhängt, wurde eine Genauigkeitsbetrachtung des Servos durchgeführt. Hierfür wurde ein Winkel von 45° mehrmals aus verschiedenen vorhergehenden Positionen angefahren und jeweils der tatsächliche erreichte Winkel mit Hilfe des Laserscanners und des Modells, welches in Unterabschnitt 3.3.2 auf Seite 23 beschrieben ist, berechnet.

Dieses wurde sowohl mit festen als auch zufälligen Vorgängerwinkeln durchgeführt und jeweils 200 Messungen durchgeführt. Es wurde ebenfalls eine Messreihe mit nicht bewegtem Servo zum Vergleich angefertigt. Anschließend wurde für alle Messreihen die Standardabweichung zum arithmetischen Mittel bestimmt um die Streuung der angefahrenen Positionen festzustellen.

Beschreibung	Standardabweichung
Standardabweichung bei nicht bewegtem Servo	$0,0918°$
Mehrmaliges Anfahren eines Winkels von 45° von einem zufälligen vorhergehenden Winkel $\beta = 45° \pm 5°$.	$0,2497°$
Mehrmaliges Anfahren eines Winkels von 45° von einem Winkel $\beta = 35°$ aus.	$0,1470°$

Erkennbar ist, dass insbesondere bei zufälligem vorhergehendem Winkel die Genauigkeit deutlich abnimmt, was dafür spricht, dass ein eingestellter Winkel von seinem vorhergehendem Winkel abhängt.

4.3 Terrainklassifikation

Im folgenden Abschnitt werden die Ergebnisse der verschiedenen Klassifikationsmethoden vorgestellt und deren Visualisierung in der GUI. Weiterhin werden die Varianzen und Eigenwerte in den Gittern, die bei der Höhenvarianz- und der PCA-Klassifikation verwendet werden, grafisch dargestellt.

4.3.1 Varianz basiert

In Abb. 4.5 auf der nächsten Seite ist das Ergebnis einer Klassifikation mittels Höhenvarianz zu sehen. Rote Punkte sind als unbefahrbar markiert, grüne als befahrbar, blaue wurden nicht bewertet aufgrund einer zu geringen Punktanzahl in der Gitterzelle und schwarze wurden nicht berücksichtigt, da die Entfernung zu ihnen zu groß ist. In der Abb. ist außerdem das der Klassifikation zugrundeliegende Gitter dargestellt, worin die Varianzen als Grauwert kodiert worden sind.

4.3.2 Principle Component Analysis basiert

Wie im vorhergehenden Unterabschnitt über die Höhenvarianz basierte Klassifizierung sind im Folgenden die Visualisierung der PCA-Klassifikation in der GUI (Abb. 4.6 auf Seite 41) und eine bildliche Darstellung des Gitters zu sehen. In der Visualisierung des Gitters sind die Eigenwerte, ähnlich wie bei der Höhenvarianz-Klassifikation, als Grauwert kodiert worden.

In Abb. 4.7 auf Seite 41 sind die beiden Klassifikationen (Höhenvarianz- und PCA-basiert) noch einmal gegenüber gestellt. Die adaptive Zellengröße ist durch größere Flächen mit gleichbleibenden Grau- oder Eigenwert klar erkennbar im Bild. Es ist außerdem erkennbar, dass Hindernisse durch die hierarchische PCA mit adaptiver Zellengöße stärker herausgearbeitet werden, als durch die Höhenvarianz basierte Klassifikation. Der in diesem 3-D-Scan auf dem Boden festgeschraubte Balken ist in der PCA-basierten Klassifikation viel klarer erkennbar.

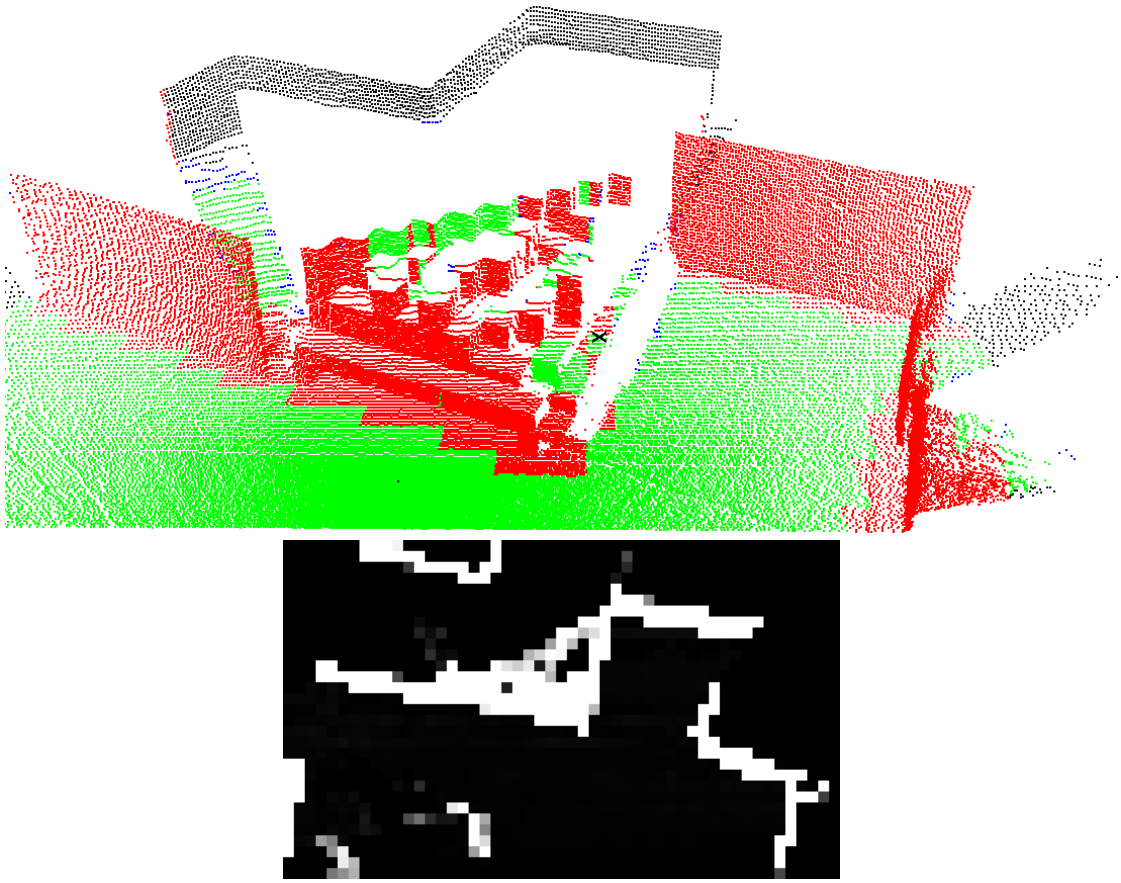


Abbildung 4.5: Oben: Visualisierung der Varianz-Terrainklassifikation in der GUI. Zu sehen ist ein Stepfield. Unten: Visualisierung der Varianzen im Gitter als Grauwertbild. Hohe Varianzen sind hell, niedrige dunkel dargestellt.

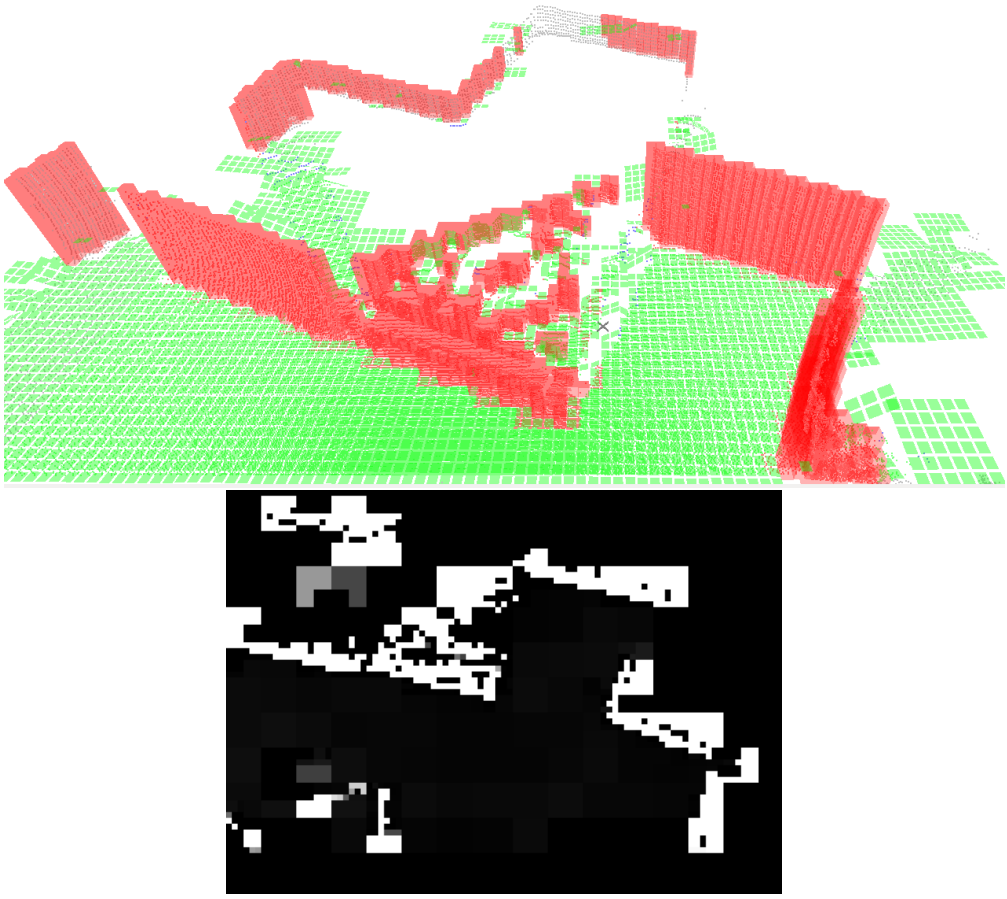


Abbildung 4.6: Oben Visualisierung der PCA-Terrainklassifikation in der GUI. Zu sehen ist ein Stepfield. Unten: Visualisierung der Eigenwerte im Gitter als Grauwertbild. Hohe Eigenwerte sind hell, niedrige dunkel dargestellt.

Abbildung 4.7: Gegenüberstellung zwischen PCA- (links) und Höhenvarianz - Klassifizierung (rechts). In der PCA basierten Klassifikation ist der Balken (rot umkreist) klarer erkennbar.

4.3.3 Kantendetektion im Tiefenbild

In Abb. 4.8 ist eine beispielhafte Verarbeitungskette des Tiefenbilds der in Abb. 4.9 zu sehenden Kante dargestellt. Die Kante wird hierbei klar erkannt. Die übrigen erkannten potentiell unbefahrbaren Punkte liegen alle an sich überschneidenden Wänden und Böden oder Löchern in der Wand und werden mittels der in Abschnitt 3.4.3 auf Seite 31 beschrieben Verfahrensweise auf den jeweils höchsten Punkt der Kante, also auf die Wände abgebildet. Es werden also keine befahrbaren Flächen fälschlicherweise gesperrt.

Das Ergebnis der Klassifikation sind sowohl im letzten Tiefenbild als auch in der resultierenden Karte in der GUI (Abb. 4.10 auf der nächsten Seite) zu sehen. Dass die Höhenvarianz- und PCA-basierte Klassifikation diese Kante nicht erkennen ist Abb. 4.11 auf Seite 44 zu entnehmen. Die Abgrund-Kante ist nicht heller eingefärbt als die befahrbaren Flächen. Die Klassifikation mittels Tiefenbild erkennt auch oft andere Hindernisse, wenn diese horizontale Kanten im Tiefenbild erzeugen. Dies ist z.B. auch bei Stepfields der Fall. Ein Beispiel ist in Abb. 4.12 auf Seite 44 zu sehen. Das Stepfield war bereits in Abb. 4.1 auf Seite 35 zu sehen. Die Kanten links und rechts sind Wände oder Personen und stellen keinen falsch erkannten Hindernisse dar.

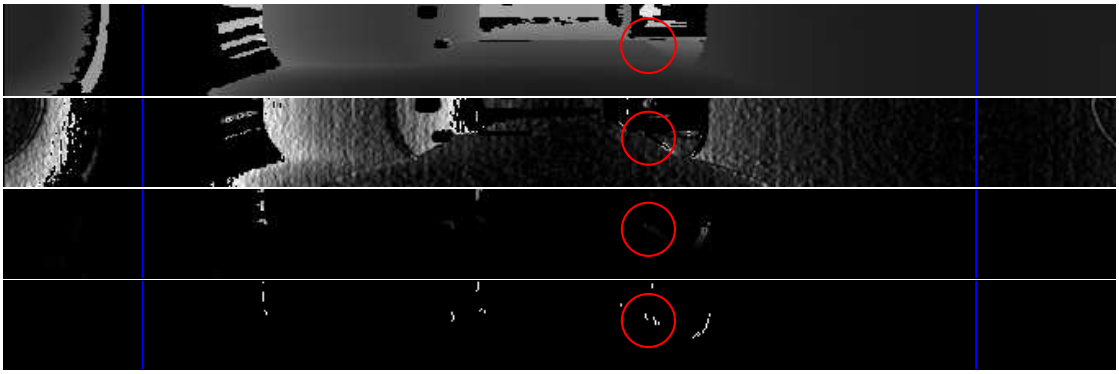


Abbildung 4.8: Die Verarbeitungskette des Tiefenbilds. Von oben nach unten: Tiefenbild, mit Sobel-Operator abgeleitetes Tiefenbild, erneute Ableitung mit symmetrischer Differenz, Endergebnis (die Kante zum Abgrund ist jeweils rot eingekreist) Die durch die blauen Linien abgetrennten Bereiche rechts und links sind Laserstrahlen, welche nach hinten gerichtet sind und somit nicht berücksichtigt werden sollten.

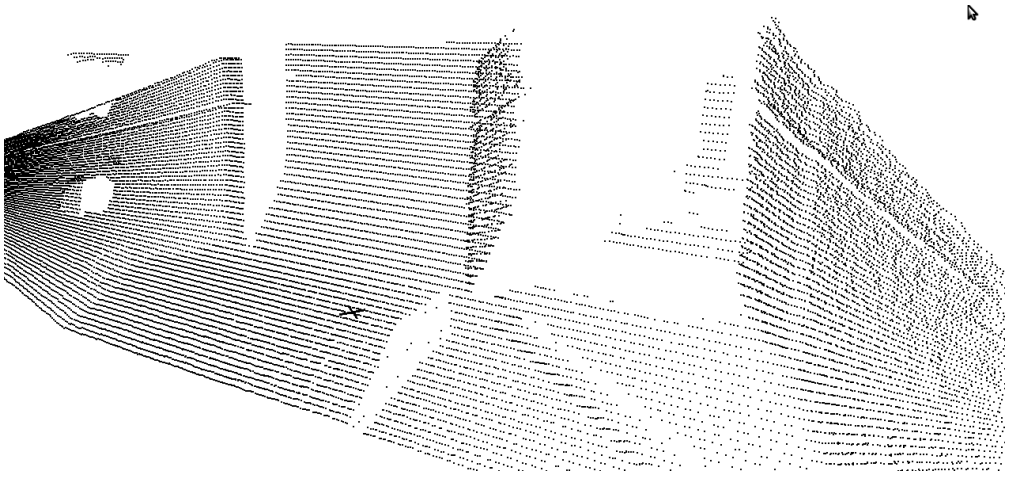


Abbildung 4.9: Eine Abgrund-Kante in der 3D-Ansicht

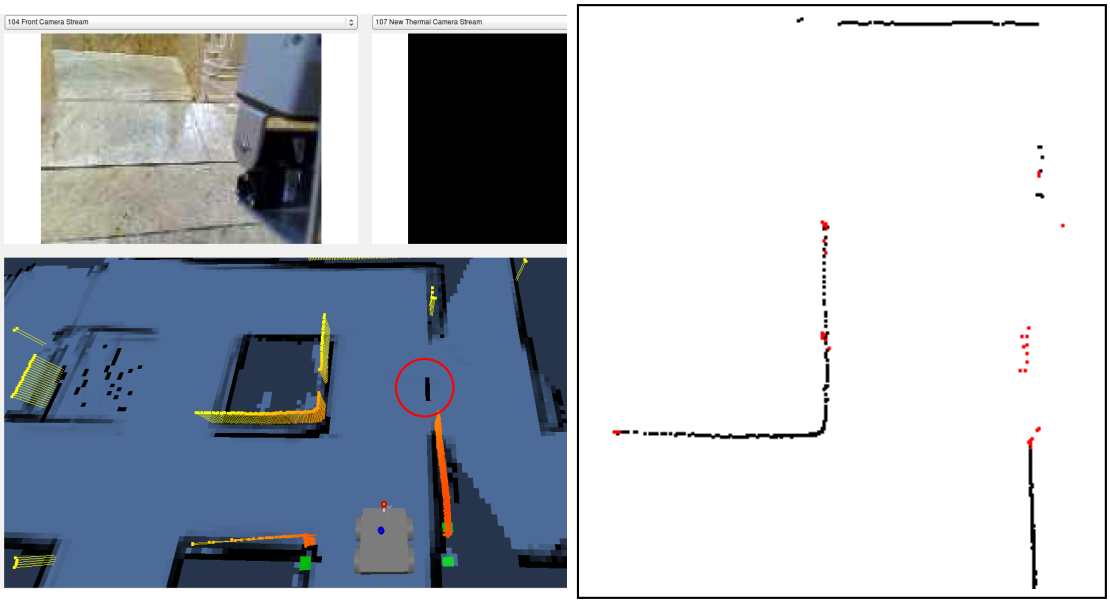


Abbildung 4.10: Links: Der eingezeichnete Abgrund in der GUI-Karte (rot eingekreist) Der aktuelle 2-D-Laserscan ist hier gelb/orange dargestellt; Rechts: Die von der Tiefenbildklassifikation erkannten potentiell unbefahrbaren Punkte (rot) im aktuellen 2-D-Laserscan (schwarz)

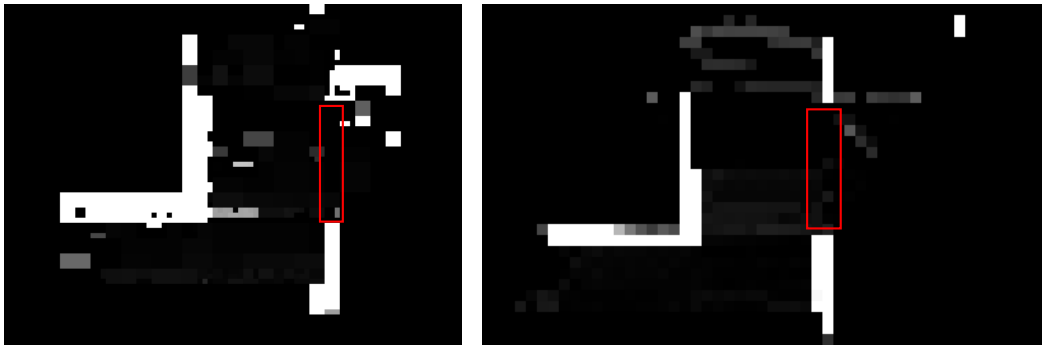


Abbildung 4.11: Varianz und PCA basierte Klassifikation an der Kante des Abgrunds. (Jeweils mit einem roten Rechteck gekennzeichnet)

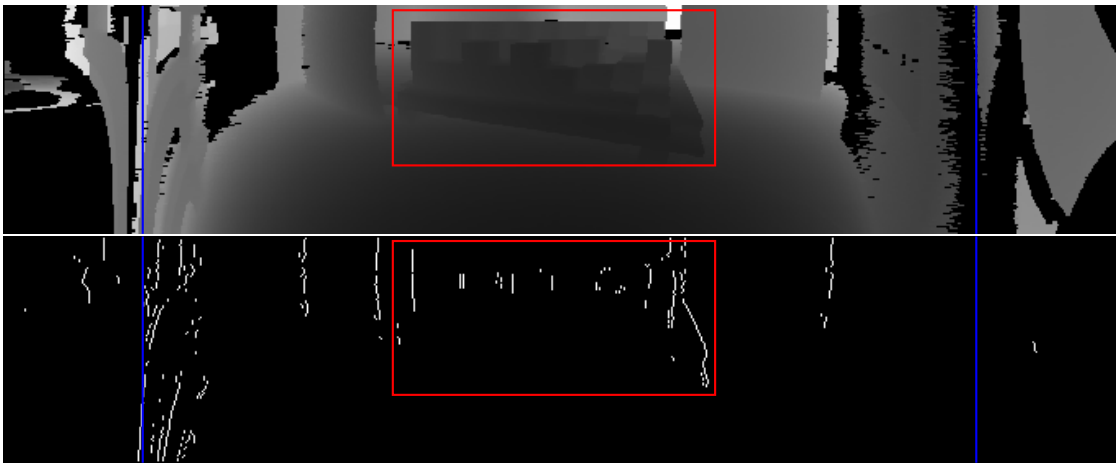


Abbildung 4.12: Die Kantenerkennung im Tiefenbild auf ein Stepfield angewandt. Die durch die blauen Linien abgetrennten Bereiche rechts und links sind Laserstrahlen, welche nach hinten gerichtet sind und somit nicht berücksichtigt werden sollten.

4.3.4 Vergleich und Diskussion

Im vorangehenden Abschnitt wurden drei verschiedene Verfahren vorgestellt, welche sich alle für die Klassifikation von Terrain mit einem schwenkbaren 2-D-Laserscanner eignen. Diese drei Vefahren haben verschiedene Vor- und Nachteile, welche im folgenden kurz diskutiert werden:

Die Terrainklassifikation mittels Varianzberechnung in Gitterzellen ist vergleichsweise einfach zu implementieren und bietet eine sehr robuste Klassifikation in relativ strukturierten Umgebungen. Sie eignet sich insbesondere für die Navigation mit einem Roboter in Gebäuden, in welchen es keine Abgründe (Löcher oder Treppen) gibt. Hindernisse, die vom Fußpunkt des Roboters aus gesehen, eine positiv Höhe haben werden zuverlässig erkannt. Schrägen können je nach Wahl der Schwellwerte fälschlicherweise als nicht befahrbar markiert werden.

Die Terrainklassifikation mittels PCA ist etwas komplexer zu implementieren, bietet jedoch eine feingranularere Erkennung von Hindernissen in der Nähe und keine Fehl-Klassifizierung von Schrägen. Sie liefert außerdem bessere Ergebnisse bei geringer und stark schwankender Punktdichte, da sie sich aufgrund des vorgestellten hierarchischen Verfahrens automatisch anpasst. Sie bietet ebenfalls keine Detektion von Abgründen.

Die Kantendetektion im Tiefenbild zielt hauptsächlich auf die Erkennung von Abgründen (z.B. Treppen) ab und erkennt diese sicher, wenn der Abgrund in einem bestimmten Winkel gescannt wird und die Kante deutlich im Tiefenbild erkennbar ist. Sie bietet oft auch eine Erkennung von anderen Hindernissen, wenn diese horizontale Kanten im Tiefenbild erzeugen. Diese werden jedoch nicht so sicher klassifiziert wie von dem Varianz-Klassifizierer und der PCA-Klassifikation.

4.4 Verifikation der Ergebnisse

Die in den vorhergehenden Abschnitten vorgestellten Ergebnisse sind exemplarisch aus einer Serie von Tests mit Logfiles entnommen, welche benutzt wurden um die Algorithmen und Verfahren zu testen. Die verwendete Softwareversion, sowie die Logfiles und eine Beschreibung für den Aufbau der Softwareumgebung befinden sich auf der dieser Studienarbeit beiligenden CD. Die Übersicht über den Inhalt der CD ist in Anhang B auf Seite 53 zu finden.

Kapitel 5

Zusammenfassung

In den vorhergehenden Kapiteln wurde verschiedene Verfahren für die Klassifikation von Terrain vorgestellt, welche es einem Roboter, der nur mit einem 2-D-Laserscanner ausgestattet ist, ermöglichen, Hindernisse zu erkennen, die sich nicht auf der Höhe des Laserscanners befinden.

Hierzu wurde im ersten Schritt der Lageausgleich basierend auf handelsüblichen Servos, welcher für die 2-D-Navigation in unebenem Gelände bereits vorhanden war, um die Möglichkeit erweitert mit ihm beliebige 3-D-Scans durchführen zu können und die Transformationsmatrizen ermittelt um die einzelnen 2-D-Scans in ein gemeinsames Koordinatensystem überführen zu können.

Im zweiten Schritt wurden, teilweise auf Basis wissenschaftlicher Literatur, mehrere Verfahren implementiert, welche die 3-D-Punktwolke auf befahrbare Flächen und Hindernisse untersuchen und diese Algorithmen in die bereits bestehende Software integriert. Eine Übersicht über die erstellten Bestandteile ist in Anhang A auf Seite 49 zu finden.

Diese Algorithmen wurden schließlich mit Hilfe von Logfiles auf deren Funktionsfähigkeit getestet und die anpassbaren Parameter der Algorithmen entsprechend eingestellt. Weiterhin mussten sich die Algorithmen auch im praktischen Einsatz in der Robocup Rescue League[1] beweisen.

Insgesamt konnten, von der Seite der Algorithmen betrachtet, keine Fehler oder Fehlklassifikationen beobachtet werden. Spalten im Boden in RoboCup Rescue, die jedoch normal überfahren werden könnten, haben sich oft als problematisch herausgestellt, kommen in der Realität außerhalb des Wettbewerbs aber eher selten vor. Für den Einsatz in Gebäuden oder im normalen Outdoor-Bereich reichen die Algorithmen im Normalfall jedoch ohne weiteres aus. Für den Einsatz in einem echten Erdbebenszenario muss jedoch noch weitere Forschung betrieben werden.

[1] `http://www.robocup.org/`

5.1 Ausblick

Aufgrund der Erkenntnisse über die Genauigkeit der Servos in Abschnitt 4.2 auf Seite 36 und daraus in der Praxis entstehender Fehlklassifikationen eignen sich diese nur bedingt für die Erstellung von 3-D-Scans. Das Problem wird vor allem in der Ferne deutlich. Durch die Toleranz von $0,46°$ des Servo internen Reglers ist eine Erhöhung der Punktdichte, obwohl theoretisch denkbar, in der Ferne unmöglich, da der Servo nicht genau genug angesteuert werden kann. Durch den Einsatz eines Schrittmotors oder einer professionellen Tilt-Einheit mit Hochpräzisionsgetriebe würde dieses Problem beseitigt.

In der Klassifikation mittels Höhenvarianz und PCA könnten noch andere Aufteilungsstrategien als das hierarchische Verfahren und die damit verbundenen Gitter erprobt und implementiert werden. Z. B. eine Klassifikation basierend auf Quad-Trees. Weiterhin sind maschinelle Lern-Vefahren im Zusammenspiel mit Kameras bei der Klassifikation von 3-D-Scans durchaus von Interesse. Hier könnte z.B. die Farbe von befahrbarem Untergrund, wie es in [TMA06b] beschrieben wird, automatisch gelernt und somit eine noch bessere Klassifikation erreicht werden.

Anhang A

Übersicht über die entwickelte Software

A.1 Integration in die Robbie Software

Der folgende Abschnitt gibt eine Übersicht über die Robbie-Architektur, in welche die beschriebenen Verfahren integriert wurden, sowie die implementierten Module, Messages und Worker-Klassen. Die entsprechende Quellen sind auf der dieser Studienarbeit beiliegenden CD zu finden. Nähere Informationen zur CD sind in Anhang B zu finden.

A.1.1 Architektur

Die Robbie-Architektur sieht eine Aufteilung in verschiedene Komponenten vor. Zur Verfügung gestellt werden sie in Form von entsprechenden (teils abstrakten) Basisklassen, die von Nutzern der Library verwendet werden können (aber nicht müssen). Die Aufgaben der Komponenten werden im Folgenden erläutert:

Dispatcher Der Dispatcher stellt den Kern (engl. System Core) der Software dar. Er ist dafür verantwortlich *Module* zu starten bzw. zu stoppen und die Kommunikation unter ihnen zu ermöglichen. Jegliche Kommunikation zwischen Modulen erfolgt über den Kern – es gibt keine direkte Kommunikation zwischen Modulen. Ebenfalls ist es nicht möglich eine Nachricht gezielt an ein bestimmtes Modul zu verschicken. Möchte ein Modul eine Nachricht versenden, wird diese an den Kern übergeben. Möchte ein Modul eine bestimmte Nachricht empfangen, muss es diese zuvor abonnieren (Publisher-Subscriber Paradigma). Der Kern leitet dann die empfangene Nachrichten an alle Subscriber weiter.

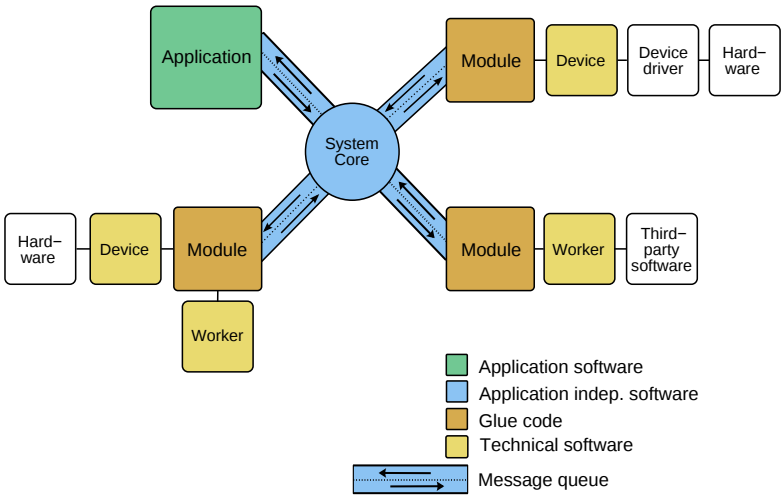


Abbildung A.1: Software-Architektur bestehend aus Systemkern, Modulen, Workern und Devices

Module Module implementieren die Schnittstelle zwischen den Messaging System und dem fachlichen Programmcode. Diese Implementierung befindet sich nicht direkt im Modul, sondern in den *Workern* (für die Datenverarbeitung) oder in *Devices* (für die Datenerfassung oder die Gerätesteuerung). Jedes Modul ist als separater Thread realisiert und besitzt eine Queue, in die der Dispatcher Nachrichten für das Modul einfügen kann. Dies ermöglicht das parallele Ausführen mehrerer Module, was besonders bei den heute immer häufiger anzutreffenden Multi-Core-Prozessorsystemen Vorteile mit sich bringt. Dadurch dass Module nicht direkt miteinander interagieren können bzw. müssen, muss bei der Entwicklung dieser auch nicht auf Thread-Sicherheit geachtet werden, wodurch eine große Fehlerquelle eliminiert wird. Die Thread-Sicherheit wird durch den Kern gewährleistet.

Devices/Worker Worker entsprechen im Grunde dem eigentlichen datenverarbeitenden Programmcode. Die strikte Trennung vom Message System nach dem Prinzip der Trennung der Belange reduziert die Kopplung der Klassen an die konkrete Architektur. Devices sind per Konvention spezielle Worker, deren Aufgabe es ist bestimmte Hardware (z. B. einen bestimmten Sensor) als eine Klasse zu abstrahieren.

Messages Messages (Nachrichten) werden von Modulen versandt und stellen sogenannte Transportobjekte dar. Wie bereits erwähnt müssen sich Module, welche diese Message empfangen möchten zunächst beim Kern registrieren und die Nachrichten abonnieren. Danach erfolgt die Zustellung automatisch.

A.1.2 Erstellte Module

ActiveLaserscannerModule

- Ansteuerung und Auslesen der Daten des Laserscanners
- Steuerung der Servos, mit welchen der Laserscanner justiert werden kann
- Auslesen und Filterung der Werte des Lagesensors, sowie Justieren des Laserscanners
- Senden von Standard-Lasernachrichten
- Empfangen und Ausführen von 3-D-Scan-Anfragen mittels der Nachrichten *GetLaserscanDirectedM* und *LaserscanDirectedM*.

ObstacleDetectionModule

- Anfordern von 3-D-Scans
- Analyse von 3-D-Scans mittels der entwickelten Worker
- Aktualisieren der Karte mit den gefundenen 3-D Hindernissen

A.1.3 Erstellte Messages

GetLaserscanDirectedM

Dient zum Anfordern eines 3-D-Scans. Als Parameter wird eine Liste von Winkelpaaren erwartet, auf welche die beiden Servos eingestellt werden sollen. Zu jedem Winkelpaar wird später ein einzelner Laserscan zurückgeliefert.

LaserscanDirectedM

Enthält das Ergebnis eines mit *GetLaserscanDirectedM* angeforderten 3-D-Scans. In der Message enthalten ist eine Zuordnung zwischen den angeforderten Winkelstellungen und den dazugehörigen Laserscans.

A.1.4 Erstellte Worker

Die im folgenden beschriebenen Klassen sind alle im Verzeichnis `Workers/LaserDetection` des Robbie-Programmcodes zu finden.

LaserCoordinateConverter

Konvertiert die Winkelpaare und Laserscans aus einer *LaserscanDirectedM* zu 3-D-Laserkoordinaten, wie in Abschnitt 3.1 auf Seite 19 beschrieben.

TerrainVarianceClassifier

Klassifiziert befahr- und unbefahrbare Flächen mittels Höhenvarianz, wie in Abschnitt 3.4.1 auf Seite 25 beschrieben. Erwartet eine 3-D-Punktwolke als Eingabe. Ausgabe sind die Mittelpunkte der unbefahrbaren Gitterzellen.

TerrainPCAClassifier

Klassifiziert befahr- und unbefahrbare Flächen mittels Höhenvarianz, wie in Abschnitt 3.4.2 auf Seite 28 beschrieben. Erwartet eine 3-D-Punktwolke als Eingabe. Ausgabe sind die Mittelpunkte der befahrbaren und unbefahrbaren Gitterzellen.

TerrainImageClassifier

Findet Anomalien und damit potentiell unbefahrbare Punkte im Tiefenbild, wie in Abschnitt 3.4.3 auf Seite 31 beschrieben. Erwartet eine 3-D-Punktwolke und ein Tiefenbild als Eingabe. Ausgabe sind unbefahrbare Punkte entsprechend den Kanten im Tiefenbild.

A.2 Mikrokontroller

Für den verwendeten Mikrokontroller zur Erzeugung des PWM-Signals (Ein *Atmega 8* der Firma [1]*Atmel*, welcher auf der Experimentierplatine *Crumb8-USB* der Firma [2]*chip45.com* verbaut ist.) wurde entsprechender Mikrokontroller-Code geschrieben, welcher sich auf der beiliegenden CD befindet.

Ebenfalls wurde die Device-Klasse `ServoController` in der Robbie Architektur erstellt um mit dem Mikrokontroller kommunizieren zu können.

[1] `http://www.atmel.com/`
[2] `http://www.chip45.com/`

Anhang B

Übersicht über den Inhalt der CD

Vezeichnis	Inhalt
Ausarbeitung/	Diese Ausarbeitung in elektronischer Form, mit LaTeX-Quellcodes und verwendeten Bildern.
Programmcode/	Robbie-Software inkl. implementierter Klassen, Mikrokontroller-Quellcodes zur Ansteuerung des Servomotors
Experimente/	Alle erstellten und verwendeten Messdaten zur Servokalibrierung, sowie Logfiles mit 3-D-Scans.
TechDoc/	Datenblätter der unterschiedlichen Hardwarekomponenten wie Kamera, Laserscanner, Servomotor, Mikrocontroller und Aufbau der Softwareentwicklungsumgebung für die Robbie Architektur.
Literatur/	In elektronischer Form verfügbare Literatur, die für die Studienarbeit verwendet wurde.

Tabelle B.1: Inhalt der beiligenden CD-ROM.

Literaturverzeichnis

[ATC+05] Anguelov, Dragomir ; Taskar, Ben ; Chatalbashev, Vassil ; Koller, Daphne ; Gupta, Dinkar ; Heitz, Geremy ; Ng, Andrew: Discriminative Learning of Markov Random Fields for Segmentation of 3D Scan Data. In: *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2*. Washington, DC, USA : IEEE Computer Society, 2005, 169–176

[Brö07] Bröhl, Stefan: *Bildbasiertes, aktives 3D-Laserscannen*, Koblenz-Landau, Diplomarbeit, 2007

[Del07] Delis, Christian: *Entwicklung einer Rotationsplattform für den Hokuyo URG-04LX Laserscanner*, Universität Koblenz-Landau, Studienarbeit, 2007

[Hot33] Hotelling, Harold: Analysis of a complex of statistical variables into principal components. In: *Journal of Educational Psychology* (1933), Nr. 24, S. 417–441

[Jäh05] Jähne, Bernd: *Digitale Bildverarbeitung*. 6., überarbeitete und erweiterte. Springer Verlag, 2005

[LVH07] Lalonde, Jean-Francois ; Vandapel, Nicolas ; Hebert, Martial: Data Structures for Efficient Dynamic Processing in 3-D. In: *The International Journal of Robotics Research* 26 (2007), 8, Nr. 8, S. 777–796

[LVHH06] Lalonde, Jean-Francois ; Vandapel, Nicolas ; Huber, Daniel ; Hebert, Martial: Natural terrain classification using three-dimensional ladar data for ground robot mobility. In: *Journal of field Robotics* 23 (2006), 11, Nr. 10, S. 839 – 861

[MBB+08] Montemerlo, Michael ; Becker, J. ; Bhat, S. ; Dahlkamp, H. ; Dolgov, D. ; Ettinger, S. ; Hähnel, Dirk ; Hilden, T. ; Hoff-

MANN, G. ; HUHNKE, B. ; JOHNSTON, D. ; KLUMPP, S. ; LANGER, D. ; LEVANDOWSKI, A. ; LEVINSON, J. ; MARCIL, J. ; ORENSTEIN, D. ; PAEFGEN, J. ; PENNY, I. ; PETROVSKAYA, A. ; PFLUEGER, M. ; STANEK, G. ; STAVENS, D. ; VOGT, A. ; THRUN, Sebastian: Junior: The Stanford Entry in the Urban Challenge. In: *Journal of field Robotics* (2008)

[MVH08] MUNOZ, Daniel ; VANDAPEL, Nicolas ; HEBERT, Martial: Directional Associative Markov Network for 3-D Point Cloud Classification. In: *Fourth International Symposium on 3D Data Processing, Visualization and Transmission*, 2008

[NDPP09] NEUHAUS, Frank ; DILLENBERGER, Denis ; PELLENZ, Johannes ; PAULUS, Dietrich: Terrain Drivability Analysis in 3D Laser Range Data for Autonomous Robot Navigation in Unstructured Environments. In: *ETFA 2009 - 14th IEEE International Conference on Emerging Techonologies and Factory Automation*, 2009

[Pea01] PEARSON, Karl: On lines and planes of closest fit to systems of points in space. In: *Philosophical Magazine* (1901), Nr. 2, S. 559–572

[TMA+06a] THRUN, Sebastian ; MONTEMERLO, Michael ; ARON, A. ; DIEBEL, J. ; FONG, P. ; GALE, J. ; HALPENNY, M. ; HOFFMANN, G. ; LAU, K. ; OAKLEY, C. ; PALATUCCI, M. ; PRATT, V. ; STANG, P. ; STROHBAND, S. ; DUPONT, C. ; JENDROSSEK, L.-E. ; KOELEN, C. ; MARKEY, C. ; RUMMEL, C. ; NIEKERK, J. van ; JENSEN, E. ; ALESSANDRINI, P. ; BRADSKI, G. ; DAVIES, B. ; ETTINGER, S. ; NEFIAN, A. ; MAHONEY, P. ; DAHLKAMP, Hendrik ; STAVENS, David ; KAEHLER, Adrian: Winning the DARPA Grand Challenge. In: *Journal of field Robotics* (2006)

[TMA06b] THRUN, Sebastian ; MONTEMERLO, Michael ; ARON, Andrei: Probabilistic Terrain Analysis For High-Speed Desert Driving. In: SUKHATME, Gaurav S. (Hrsg.) ; SCHAAL, Stefan (Hrsg.) ; BURGARD, Wolfram (Hrsg.) ; FOX, Dieter (Hrsg.): *Robotics: Science and Systems*, The MIT Press, 2006