Zürcher Hochschule
für Angewandte Wissenschaften

School of
Engineering

InIT Institut für angewandte
Informationstechnologie

Projektarbeit Systemtechnik

Bewegungsplanung für einen mobilen Roboter zum Glätten von Beachvolleyball-Feldern

Autoren	Lukas Ruckstuhl Stefan Welle
Hauptbetreuung	Mark Cieliebak
Nebenbetreuung	Joachim Wirth
Datum	20.12.2013

,

Zürcher Hochschule
für Angewandte Wissenschaften

Erklärung betreffend das selbständige Verfassen einer Projektarbeit an der School of Engineering

Mit der Abgabe dieser Projektarbeit versichert der/die Studierende, dass er/sie die Arbeit selbständig und ohne fremde Hilfe verfasst hat. (Bei Gruppenarbeiten gelten die Leistungen der übrigen Gruppenmitglieder nicht als fremde Hilfe.)

Der/die unterzeichnende Studierende erklärt, dass alle zitierten Quellen (auch Internetseiten) im Text oder Anhang korrekt nachgewiesen sind, d.h. dass die Projektarbeit keine Plagiate enthält, also keine Teile, die teilweise oder vollständig aus einem fremden Text oder einer fremden Arbeit unter Vorgabe der eigenen Urheberschaft bzw. ohne Quellenangabe übernommen worden sind.

Bei Verfehlungen aller Art treten die Paragraphen 39 und 40 (Unredlichkeit und Verfahren bei Unredlichkeit) der ZHAW Prüfungsordnung sowie die Bestimmungen der Disziplinarmassnahmen der Hochschulordnung in Kraft.

Ort, Datum:	Unterschriften:
………………………………………………	………………………………………………………………
	……………………………………………………………...
	………………………………………………………………

Zusammenfassung

Wird über eine Saison hinweg auf dem Beachvolleyballfeld gespielt, entstehen so über das gesamte Feld Höhenunterschiede von bis zu 25 cm. Bisher gibt es keine Möglichkeit ein Beachvolleyballfeld automatisch zu ebnen. Doch muss dieses am Ende einer Saison wieder geebnet werden. Dies kann bedeuten, dass eine Beachvolleyballmannschaft einen ganzen Tag lang Sand von einem Ort zum anderen schaufelt. Ziel ist es, diese Arbeit zu einem späteren Zeitpunkt einem Roboter zu überantworten.

Gegenstand der vorliegenden Arbeit ist es, einen Algorithmus zu entwickeln, der mit Hilfe des Roboters, der in einer parallellaufenden Projektarbeit realisiert wurde, Sand auf einem Beachvolleyballplatz zu ebnen. Dazu muss geprüft werden, wie es möglich ist, dass der Roboter sich autonom auf dem Sandfeld bewegt und den aufgetürmten Sand auf dem Feld verteilt.

Um dieses Ziel zu realisieren wurde Literatur zu ähnlichen Problemstellungen und Algorithmen hinzugezogen. Weiter wurden Experimente zur Aufnahme eines Sandfeldes unternommen um herauszufinden, wie die Oberfläche eines Beachvolleyballfeldes, zur weiteren Verwendung, aufgenommen werden kann. Diese Daten dienten zudem dazu um die Algorithmen zu testen. Um möglichst viele Parametereinstellungen im Algorithmus zu definieren, wurde ein Mechanismus konstruiert welcher die Grundfunktionen des verwendeten Roboters auf einem Sandfeld simulieren kann. Dies ermöglichte das Verhalten des Roboters im Vorfeld zu untersuchen und so den Algorithmus auf möglichst reale Bedingungen einzustellen.

Beim Programmieren des Algorithmus wurde auf verschiedene Weisen vorgegangen. Es wurden verschiedene Methoden zur Glättung eines Feldes zusammengetragen und in einfachen Simulationen getestet. Grosse Erkenntnis über die Art des Glättens lieferten auch die Eigenversuche mit dem Robotermodell, bei denen das Vorgehen beim Glätten des Felds beobachtet wurde.

Die realisierten Algorithmen wurden in einer Matlab Umgebung erstellt. Zusätzlich wurde eine Methode entwickelt, die es ermöglicht die Oberfläche einer Box, welche mit Sand befüllt ist, aufzunehmen. So wurde eine Bibliothek erstellt welche verschiedene Sandfelder beinhaltet die als Matrizen eingelesen werden, um die entwickelten Algorithmen unter verschiedenen Bedingungen zu testen. Die verschiedenen Algorithmen wurden verglichen und lösen Teilaspekte des gesamten Problems, jedoch wurde noch kein endgültiger Algorithmus entwickelt.

Vorwort

Algorithmen sind für viele Menschen integrierte, unsichtbare Helfer. Kaum jemand realisiert wie oft und welch komplexe Programme sich im Hintergrund von ganz alltäglichen Gegenständen abspielen.

Im Rahmen dieser Projektarbeit wurden Algorithmen für einen Roboter, der sich auf Sand fortbewegt, entwickelt und ausgetestet. Automatisierung und das Einsetzen von Robotern findet in immer neuen Bereichen Anwendung. Es war deshalb spannend sich mit dem Entwickeln von Algorithmen auseinander zusetzen und eine Herausforderung, immer neue Methoden zu entwickeln, um die Algorithmen zu verbessern und zu testen.

Die Verfasser dieser Projektarbeit, Lukas Ruckstuhl und Stefan Welle von der Zürcher Hochschule für angewandte Wissenschaften in Winterthur (ZHAW), bedanken sich bei den hochschulinternen Kollegen Christoph Gugg und Niculin Lutz für die gute Zusammenarbeit. Danken möchten wir an dieser Stelle auch dem Team der ZHAW internen Werkstatt für Ihre Hilfe, all die benötigten Teile zu besorgen und herzustellen. Ein spezieller Dank gilt unseren Betreuern Dr. Mark Cieliebak und Joachim Wirth für die vielen konstruktiven Treffen und die engagierte Unterstützung während der gesamten Projektdauer.

Inhaltsverzeichnis

1 Einleitung

Es gibt bereits viele Roboter, die monotone und anstrengende Arbeiten übernehmen. Die Meisten werden bis anhin in gewerblicher Umgebung eingesetzt. Jedoch werden immer häufiger auch Roboter, zur Unterstützung des Menschen, für private Einsatzgebiete entwickelt. In diesem Fall soll eine ganz spezifische Arbeit an einen Roboter übergeben werden können: Das Glätten eines Beachvolleyballfeldes. Dazu braucht der Roboter einen Algorithmus, der bestimmt, wie sich der Roboter über das Feld bewegt, wo er Sand abtragen und in welche Richtung er den abgetragenen Sand verteilen soll.

Dieses Kapitel beinhaltet zu Beginn einen Abschnitt 1.1 über den aktuellen Stand der Technik. Im zweiten Abschnitt 1.2 wird erläutert, wie das Anwendungsgebiet des Roboters und des integrierten Algorithmus aussehen könnte. Im letzten Abschnitt 1.3 ist die Zielsetzung dieser Projektarbeit aufgeführt.

1.1 Stand der Technik

Da zum Thema Sandglättung keine bestehende Arbeit gefunden wurde, wird an dieser Stelle erläutert, welche Informationen bei der Recherche zum Thema Sandglättung im allgemeinen sowie zum Thema automatisiertes Fahren und Verteilen durch Algorithmen gefunden wurden.

1.1.1 Sandglättungsverfahren

Bei der Informationssuche zur Umverteilung von Sand, zu Beginn dieses Projekts, wurden verschiedene Ansätze gefunden, welche bei ähnlichen Problemstellungen Verwendung fanden. Diese Ansätze werden an dieser Stelle vorgestellt um die Problematik eines automatisierten Glättungssystems hervorzuheben.

In den meisten Fällen werden die Anlagen, die geglättet werden müssen, noch von Hand und mit viel Schweiss wieder zu einer ebenen Fläche umgegraben. Das Verschieben von geringen Mengen an trockenem Sand bedeutet einen grossen Kraftaufwand und nimmt viel Zeit in Anspruch.

Abbildung 1.1: Sand eines Beachvolleyballfeldes wird von mehreren Arbeitern geglättet[7]

Weitere Lösungen finden vor allem im Reitsport Anwendung, welche grosse Flächen von Sand in kurzer Zeit wieder ebnen müssen. Dort werden spezielle Anhänger an landwirtschaftliche Fahrzeuge angehängt und über das Feld gezogen. Jedoch werden bei dieser Anwendung nur geringe Sandmassen bewegt. Das Ziel der grossen Konstruktionen ist es, die Unebenheit zu glätten, dazu müssen nur die neu entstandenen Löcher von der Grösse eines Pferdehufes beseitigt werden. Die Konstruktion besteht aus mehreren Stahlspitzen im vorderen Teil, die den Sand aufrauen sollen und einer Walze am hinteren Teil, die den Sand am Ende aufwühlt und auflockert. Zudem wird über Wasserschläuche ein Sprühnebel erzeugt, der dafür sorgt, dass beim Aufwirbeln des Sandes nicht zu viel Staub entsteht. Um die richtige Höhe des Hallenbodens zu gewährleisten, sind in den Hallenwänden Laser angebracht, damit der Aufbau, mit Hilfe von Sensoren, sich immer automatisch in seiner Höhe verstellt.

Abbildung 1.2: Sand einer Reithalle wird von einer Anhängerkonstruktion geglättet[8]

1.1.2 Algorithmen

Einen Algorithmus, der sich damit befasst, Sandmengen zu verschieben, konnte während der Recherche nicht gefunden werden. Es gibt aber bereits viele Algorithmen, die sich mit dem Verschieben von Objekten oder dem Abfahren von Strecken befasst haben. Das Ziel dieser Algorithmen variiert stark, manche wurden ausgelegt um Strecken so zu fahren, dass sie jeden Punkt auf der Strecke einmal durchlaufen, andere wiederum suchen den kürzesten Weg von einem Wegpunkt zum Anderen. Auch bei der Umverteilung von Objekten liessen sich verschiedene Ansätze finden. Einige wurden darauf ausgelegt die Objekte schnell umzuverteilen, andere versuchten das Volumen des Objekts möglichst genau zu ermitteln und so einen möglichst passenden Platz für das Objekt zu finden. Es wurde versucht diese Ansätze in dieser Arbeit zu verwenden. [1, 2, 3, 5]

1.2 Anwendungsgebiet

Der fertiggestellte Roboter mit dem integrierten Algorithmus soll primär dazu eingesetzt werden, zwischen den Spielen oder am Ende eines Spieltages das Beachvolleyballfeld so zu glätten, dass nur geringe Höhenunterschiede bestehen. Somit wird bei regelmässigem Gebrauch des Roboters nie die Situation entstehen, dass grosse Gräben wieder mit Sand aufgefüllt werden müssen. Der Roboter könnte auch auf anderen Sandfeldern zum Einsatz kommen, sofern die selben Möglichkeiten bestehen, die Informationen über die Sandoberfläche zu erheben wie beim Beachvolleyballfeld.

1.3 Zielsetzung

Das Ziel dieser Projektarbeit ist es, einen Algorithmus zur Glättung eines Beachvolleyballfeldes zu entwickeln. Dabei soll darauf geachtet werden, dass dieser kompatibel ist zur parallel entwickelten Projektarbeit, welche sich mit der Aufgabe beschäftigt einen Roboter zu konstruieren, welcher Sand umverteilen kann. Hierbei handelt es sich um ein Forschungsprojekt, in dem erste Erkenntnisse über die Durchführbarkeit eines solchen Roboters gewonnen werden soll.

Diese Projektarbeit besteht aus zwei Teilzielen. Als erstes sollen im Labor 3D-Modelle von Sandfeldern erstellt werden können. So soll eine Datenbank mit verschiedenen Sandfeldern realisiert werden. Diese Aufnahmen sollen dazu verwendet werden, die entwickelten Algorithmen zu testen und zu vergleichen. Das zweite Teilziel ist die Entwicklung eines Algorithmus zum Glätten dieser Felder. Hierbei sollen verschiedene Glättungsabläufe betrachtet werden.

2 Theoretische Grundlagen

Zur Durchführung dieser Arbeit mussten Kenntnisse in verschiedenen Fachgebieten und der Umgang mit speziellen technischen Geräten erarbeitet werden. An dieser Stelle finden sich Informationen, auf denen dieses Projekt aufbaut. Im ersten Kapitel 2.1 wird erklärt, was ein Algorithmus ist. Um die Übersichtlichkeit der Programme zur Simulation zu verbessern wurden Structures verwendet. Aus diesem Grund wird in Kapitel "2.2 Begriffserklärung Structure“ erklärt, wie Structure aufgebaut sind und wie sie eingesetzt werden. Im letzten Kapitel 2.3 werden die wichtigsten Fakten, der zur Aufnahme der Sandfelder verwendeten Roboter und Laser, erläutert.

2.1 Begriffserklärung Algorithmus

Ein Algorithmus ist eine eindeutige Handlungsvorschrift zur Lösung eines Problems oder einer Klasse von Problemen. Algorithmen bestehen aus endlich vielen, wohldefinierten Einzelschritten.[4]

Einfacher gesagt, handelt es sich bei einem Algorithmus um Arbeitsanweisungen zur Lösung einer Aufgabe ähnlich einem Kochrezept. Jedoch sollten diese Arbeitsanweisungen einige Eigenschaften erfüllen um die Bezeichnung Algorithmus zu rechtfertigen.

Zum einen muss ein Algorithmus allgemein gültig sein. Darunter wird verstanden, dass er nicht nur genau ein Problem löst, sondern alle darauf aufbauenden Probleme ebenfalls bewältigen kann. Es soll damit also nicht ein Problem gelöst werden: "wie sortiere ich 3 Zahlen", sondern ganz allgemein: "wie sortiert man Zahlen". Dabei sollte es keinen Unterschied machen, wie viele Zahlen es sind.[5]

Weiter sollte jeder Algorithmus so geschrieben sein, dass er ausführbar ist. Dies bedeutet, dass es möglich sein muss den Algorithmus bis zum Ende abzuarbeiten. Es darf also nicht unendlich viele Anweisungen geben. Zusätzlich muss jede dieser Anwendungen klar formuliert sein. Da die meisten Algorithmen beim Arbeiten mit einem Computer Anwendung finden, bedeutet dies, dass er in einer verständlichen Sprache für den Computer geschrieben werden muss.

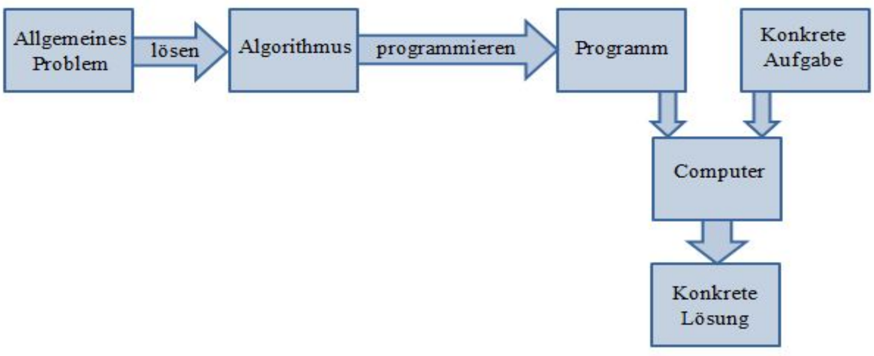


Abbildung 2.1: Der Weg vom Problem zur Lösung

2.2 Begriffserklärung Structure

Beim Programmieren der Algorithmen wurden Structures verwendet, daher wird im folgenden kurz erläutert, um was es sich bei Structures handelt. Eine Structure ist eine von fünf Datentypen, die es beim Programmieren gibt. Eine Structure wird gebraucht um etwas komplizierteres als eine einfache Zahl (int), Zeichen (char) oder einem Boolean (richtig oder falsch) zu bezeichnen, auch komplizierter als ein Array (eine Variable die mehrere Werte eines Typs enthält) von einem dieser Datentypen. Zum Beispiel, ein Student kann definiert werden durch seinen Namen, sein Alter, seine Noten oder seinen Wohnort. Jedes dieser Informationsstücke sollte gekennzeichnet werden mit einem einfachen und möglichst gut umschreibenden Namen. Gleichzeitig wäre es gut wenn alle diese Informationen zusammen gespeichert werden könnten. Structures ermöglichen die Speicherung von unterschiedlichen Datentypen unter einer einzigen Variablen sowie einen einfachen Zugriff auf die einzelnen Informationen. Dies ist einfacher und übersichtlicher als eine Gruppe von Arrays zu verwenden.[6]

```
%Beispiel eines Structures:

%Initialisierung von 4 Studenten mit Namen und Alter:

students(1).name  = 'jim';
students(1).age   = 21;

students(2).name  = 'Jane';
students(2).age   = 33;

students(3).name  = 'Joe';
students(3).age   = 25;

students(4).name  = 'Janet';
students(4).age   = 24;

%Nun kann das Gesamte Struct von Student 3 Aufgerufen werden:
students(3)

%Ausgabe:
name: 'joe'
age: 25
```

2.3 Technische Hilfsmittel

Für die Durchführung der Projektarbeit wurden mehrere technische Hilfsmittel verwendet. Diese werden im folgenden Kapitel kurz erläutert.

2.3.1 Roboter

Bei dem verwendeten Roboter handelt es sich um das Modell IRB 120 von ABB (Datenblatt siehe Anhang 7.4.3). Es ist, nach Angaben der Firma ABB, der zurzeit schnellste 6-achsige Roboter auf dem Markt. Er besitzt eine Handhabungskapazität von 3 kg und eine Reichweite von 580 mm. Der Roboter wurde verwendet um den Distanzmesslaser positionsgenau über die Sandkiste zu bewegen. Der Roboter wird über das Ethernet von einem Computer angesteuert. Zur Steuerung des Roboter wurden die Programme Matlab und RAPID verwendet.

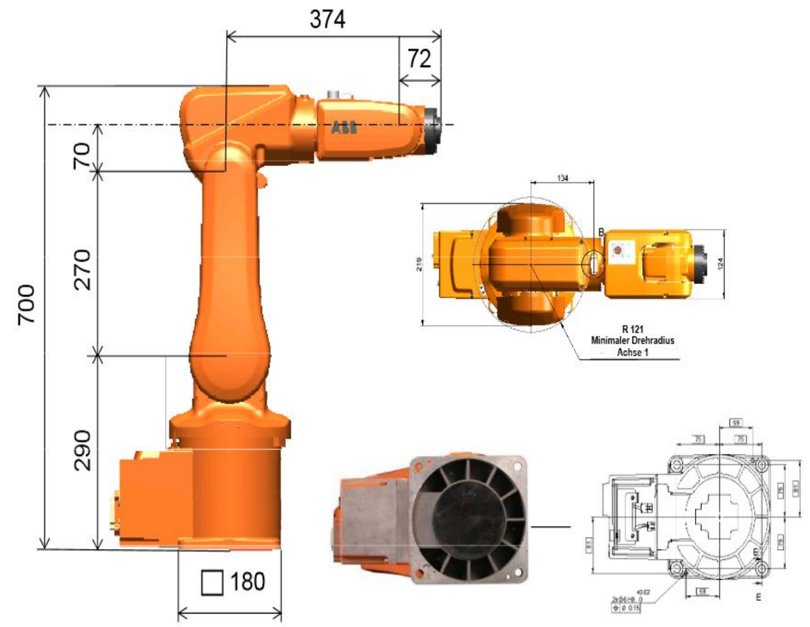


Abbildung 2.2: Roboter von ABB mit eingezeichneten Abmessungen

2.3.2 Laserscanner

Bei dem verwendeten Laserscanner handelt es sich um einen Lasermesssensor des Typs LMS511-10110 der Firma SICK. Der Laserscanner wurde zur Aufnahme verschiedener Szenarien von Sandfeldern, für das Testen des entworfenen Algorithmus, verwendet. Der Laserscanner sendet Lichstrahlen aus und misst die Zeit bis zur Rückkehr dieses Lichtes. Somit kann die Distanz zwischen dem Messobjekt und dem Laserscanner bestimmt werden. Am Laserscanner kann die Breite des Blickfeldes, mittels Anfangs- und Endwinkel, sowie die Unterteilung eingestellt werden. Die Unterteilung gibt die Anzahl Messpunkte pro Grad an und bestimmt somit die Anzahl Messpunkte. Jede Messung nimmt einen Streifen mit den Distanzen zwischen dem Scanner und dem zu messenden Gegenstand auf. Die Summe dieser Streifen ergibt eine Punktewolke des Messobjektes, die dann weiterverarbeitet werden kann. Neben dem Ethernetkabel wird auch ein Netzgerät benötigt, welches den Laser speist. Optimal wird der Laser mit 24 V betrieben. Der Laser verfügt an seiner Rückseite über vier M6 Gewinde und kann so problemlos an seinem vorgesehenen Einsatzort befestigt werden. Das zugehörige Datenblatt befindet sich im Anhang. 7.4.3

Abbildung 2.3: Laserscanner der Firma SICK

3 Vorgehen

Zur Erstellung eines zuverlässigen Algorithmus werden immer wieder verschiedene Szenarien benötigt um den Ablauf zu testen und zu verbessern. Da der zu erstellende Algorithmus die Route für einen Roboter, der Sand glättet, beinhaltet, wurden möglichst unterschiedliche Szenarien von Sandflächen benötigt.

Zudem mussten viele Parameter, betreffend dem Verhalten von Sand und Sandverschiebung erhoben werden. Aus diesem Grund entschloss man sich, eine Versuchsumgebung aufzubauen, bei der Sandfelder erstellt und studiert werden konnten. Die Versuchsumgebung konnte in der Werkstatt der ZHAW eingerichtet werden. Die ZHAW stellte für dieses Projekt ausserdem einen Roboterarm der Firma ABB so wie einen Laserscanner zur Distanzmessung zur Verfügung, deren Nutzung wird im Kapitel 3.1 erläutert.

In einem nächsten Schritt, war es erforderlich herauszufinden wie sich eine grosse Menge von Sand verhält, wenn Teile davon verschoben oder unplatziert werden. Dieses Wissen wurde durch entsprechende Experimente in unserer Versuchsumgebung in Erfahrung gebracht, welche im Kapitel 3.2 ersichtlich sind.

Mit dem notwendigen Wissen ausgerüstet konnten nun die Werkzeuge, wie Sandabtragen oder einen Weg abfahren, für die Algorithmen erstellt werden (Kapitel 3.3). Mit diesen Funktionen konnte der Roboter, der Vierteilmechanismus sowie das Sandfeld simuliert werden. Die Algorithmen konnten diese Programmteile verwenden um die Bearbeitung der virtuellen Sandfelder zu Visualisieren.

Ab diesem Punkt konnten erste Algorithmen erstellt und getestet werden. Im Kapitel 3.4 “Ausarbeitung eines Algorithmus“ wird erläutert mit welchen Methoden vorgegangen wurde, um herauszufinden wie sich der Algorithmus sinnvollerweise verhält.

3.1 Erhebung der Sandoberfläche

Als Grundlage der Versuchsumgebung wurde eine Sandbox aus Pressholzplatten zusammengebaut, die anschliessend mit Sand von einem Beachvolleyballfeld befüllt wurde. Die Sandbox wurde ebenfalls von der Projektgruppe genutzt, die sich mit der Ausarbeitung des Roboters beschäftigte, um Ihre Ideen zur Bewegung der Sandmassen zu testen. Im Verlauf des Projektes wurde sie zusätzlich mit Plexiglasscheiben ausgestattet um das Bürstenmodell zu testen, welche zur Evaluation von Parametern im Algorithmus verwendet wurde. Um eine Aufnahme des Feldes zu generieren wurde der SICK-Laser an den ABB Roboterarm, mit Hilfe einer eigens dafür angefertigten Adapterplatte, befestigt. Nun konnte die Oberfläche des Sandes in der Box mit dem Roboterarm abgefahren werden, während der Laser jeweils die Höhe einer Achse aufnahm. (Abbildung 3.1)

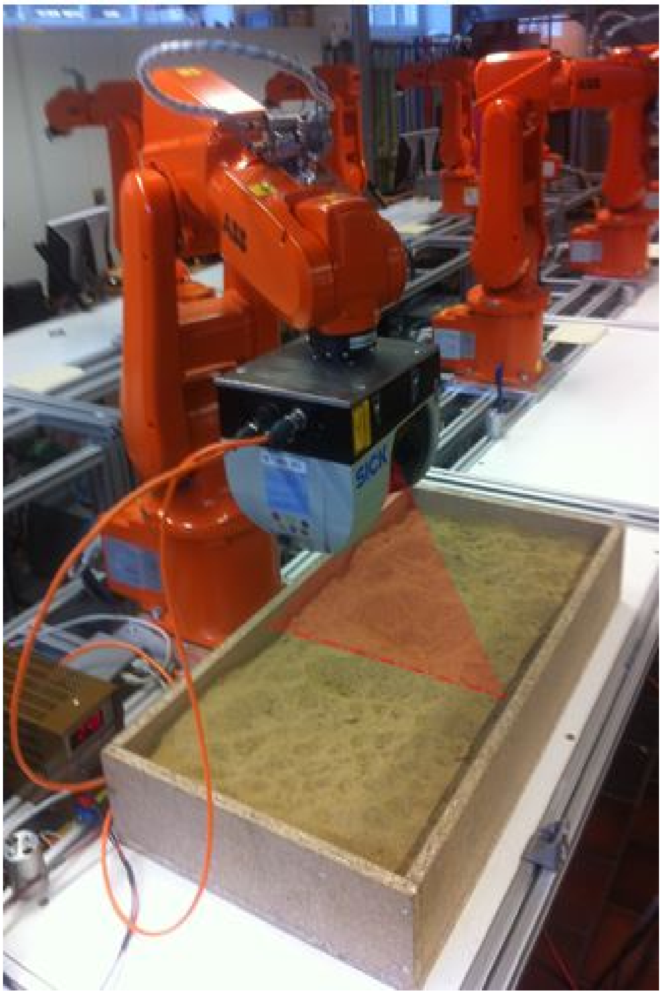


Abbildung 3.1: Nachbearbeitetes Bild: ABB Roboter mit montiertem SICK-Laser bei der Feldaufnahme

3.1.1 Digitalisierung der Aufnahmen

Die einzelnen Informationsstränge, die alle Höhen auf einer Linie beinhalten (in Abbildung 3.1 fürs bessere Verständnis mit Rot dargestellt), wurden laufend als Array auf einem über das Ethernet verbundenen Rechner in Textfiles abgespeichert. Mit Hilfe von Matlab wurden die Textfiles alle zwei oder fünf Millimeter geöffnet und die Arrays in eine Matrize abgefüllt. Die Matrizen wurden begrenzt und mit einem Mittelwertfilter aufbereitet um daraus ein Feld zu generieren. So konnten verschiede Szenarien für das Testen der Algorithmen geschaffen werden. Beispielsweise ein Feld (Abbildung 3.2), das sowohl genau eine Hoch- wie auch eine Tiefstelle beinhaltet. Die Algorithmen bearbeiten die Einträge in der Matrix, so kann jede Veränderung direkt dargestellt werden.

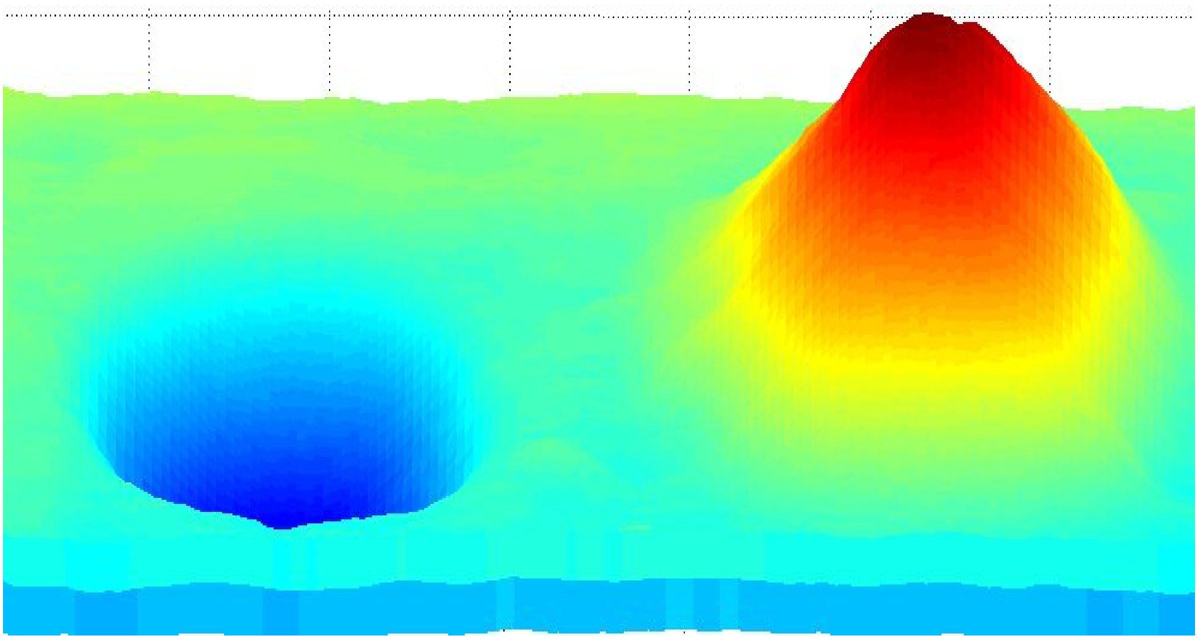


Abbildung 3.2: Ein in Matlab generiertes Bild zeigt die Oberfläche des Sandes in der Box

3.1.2 Validierung der Erhebung

Um sicherzustellen, dass die digitalisierten Aufnahmen mit den realen Werten übereinstimmen, wurde das Volumen der erstellten Aufnahme überprüft. Dazu wurden mehrere Versuche durchgeführt, deren Ergebnisse in der Tabelle 3.1 ersichtlich sind.

Zu Beginn wurde das Volumen einer Aufnahme, eines geglätteten Feldes, mit dem Volumen des Sandes in der Box, der mit Hilfe eines Messbechers genau ermittelt wurde, verglichen. In einem weiteren Schritt wurde überprüft, in welcher Grössenordnung sich die Abweichung zwischen dem Volumen, eines geglätteten Feldes und einer Aufnahme eines hügeligen Feldes, befindet. Zusätzlich wurde untersucht wie sich die Grösse der Schrittrate auf die Aufnahmegenauigkeit auswirkt. Es wurden Aufnahmen mit einer Schrittrate von zwei Millimeter und fünf Millimeter gemacht. Als Letztes wurde 4’131 cm^3 Sand aus der Box entfernt. Dieses Feld wurde ebenfalls digitalisiert, das Volumen berechnet und mit der entfernten Menge verglichen.

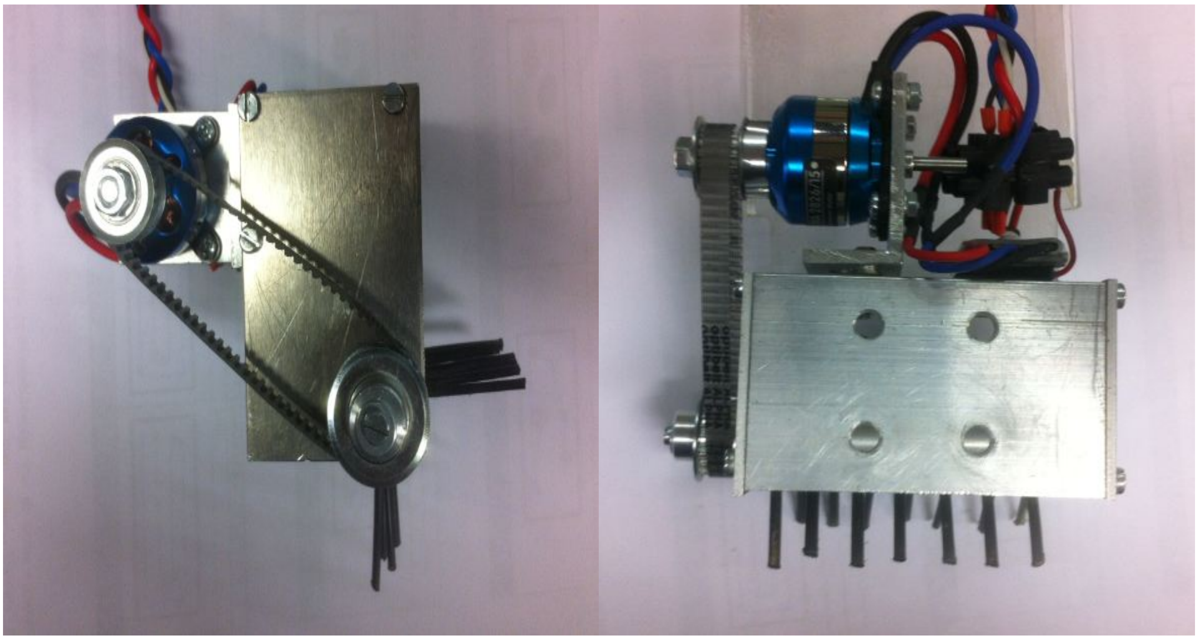
	Eben 5 mm	Uneben 5 mm	Uneben 2 mm	Sand Reduziert 2 mm
Volumen Sandbox	$18'375 cm^3$	$18'375 cm^3$	$18'375 cm^3$	$14'244 cm^3$
Volumen digitales Feld	$17'316 cm^3$	$17'679 cm^3$	$17'859 cm^3$	$14'177 cm^3$
Differenz	$1'059 cm^3$	$696 cm^3$	$516 cm^3$	$67 cm^3$

Tabelle 3.1: Validierung der digitalisierten Aufnahmen

Die Versuche zeigten, dass die verwendete Methode zur Erstellung der Oberflächen geeignet ist. Die Abweichungen, zwischen dem Inhalt der Sandbox und der Aufnahmen, scheint plausibel. Die Abweichungen sind durch das abschneiden der Felder an den Rändern zu erklären. Bei der Box mit der reduzierten Sandmenge ist der Unterschied geringer, da der Rand des Feldes genauer erkannt werden konnte. Weiter wurde ersichtlich, dass mit geringerem Abstand zwischen der Aufnahme der einzelnen Streifen die Genauigkeit ebenfalls verbesserte. Die Abweichungen von den Volumen war irrelevant für die Verwendung in den Simulationen.

3.2 Verifizierung der Sandverteilung

Die Partner-Projektgruppe entschied sich für die Methode Bürste zur Bewegung von Sand. Deshalb wurde, im Rahmen dieses Projektes, ein kleines Modell einer Bürste erstellt. Die Einzelteile für den Rahmen der Bürste wurden aus Aluminiumresten aus der Werkstatt der ZHAW erstellt. Verwendet wurden Winkelprofile und Stahlbleche. Das Herzstück des Models, die Bürste, wurde aus einer Aluminiumstange gefertigt und mit Plastikborsten durchsetzt. Als Antrieb wurde ein Brushless DC Motor verwendet, der über einen Zahnriemen, die mit Gleitlagern gelagerte Bürstenwelle, antreibt. Das Antreiben über Riemen wurde aus Platzgründen realisiert, damit in der Kiste genügend Platz für Fahrbewegungen bleiben. Die Liste der verwendeten Komponenten befindet sich im Anhang im Abschnitt 7.2. Versorgt wird das Modell von einem 7.4 V LiPo Akku, um auch Versuche im Freien zu gewährleisten. Die Umlaufsgeschwindigkeit der Bürste konnte mit Hilfe eines Motorentesters aus dem Modellbau gesteuert werden.

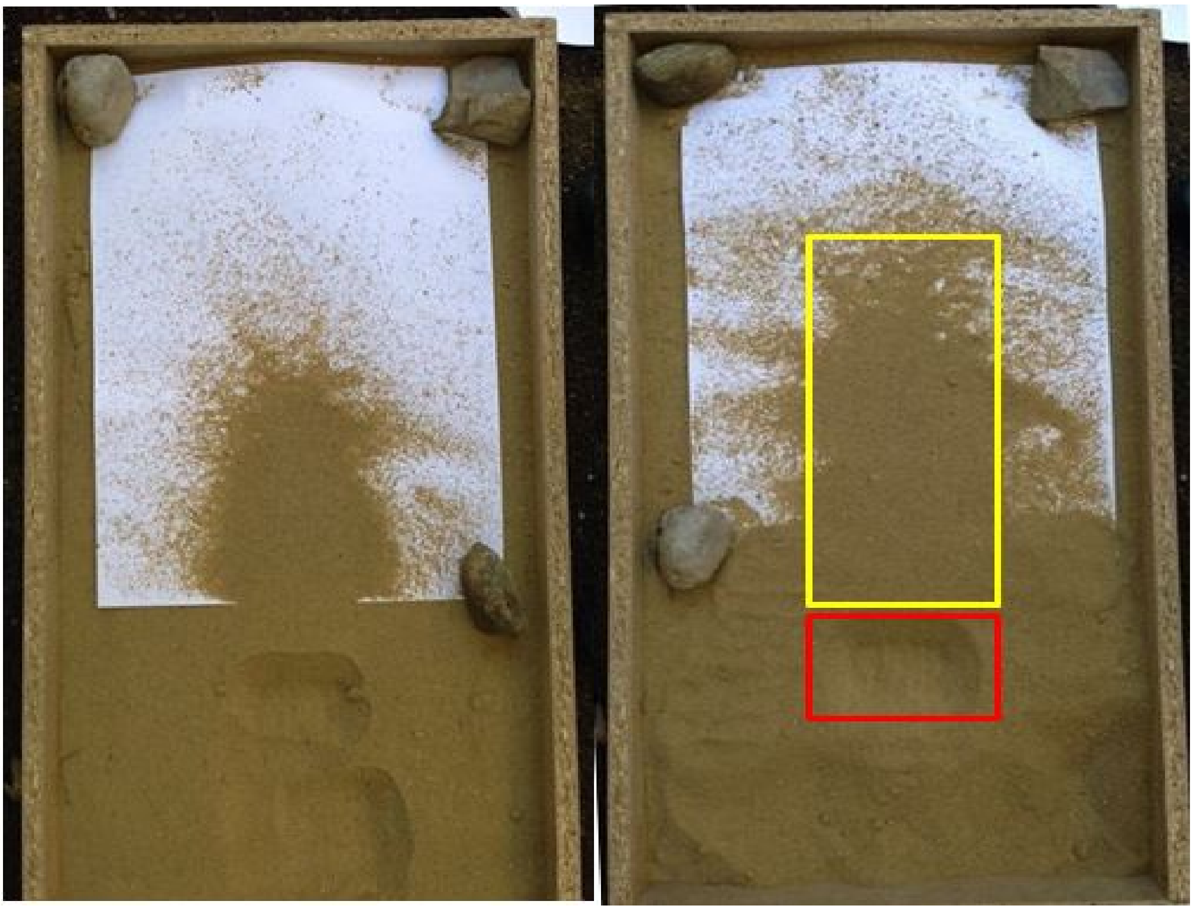


(a) Bürste von der Seite

(b) Bürste von Oben

Abbildung 3.3: Zusammengebautes Modell der Bürste

Die Bürste wurde primär hergestellt um Erfahrungen zu sammeln, wie sich der Sand bei Einwirkung der Bürste verhält. Daraus konnten wichtige Erkenntnisse für das Programmieren gewonnen werden. Beispielsweise wie weit und in welchem Winkel die Sandkörner fort geschleudert werden können. Dazu wurde die mit Sand befüllte Box nach draussen verlagert. Um deutlichere Resultate zu erhalten breitete man Blätter aus, auf denen der herumgeschleuderte Sand besser zu erkennen war. Bei jedem neuen Durchgang des Experiments wurde die Drehzahl der Bürste erhöht. Dies ergab ein klares Bild, wie sich der Sand verteilt.



(a) Versuch mit geringer Drehzahl

(b) Versuch mit höherer Drehzahl (Eingezeichneten Auf-/ Abgabe von Sand)

Abbildung 3.4: Aufnahme des Versuchs

Die wichtigste Erkenntnis für die in den Algorithmen verwendete Sandverteilungsmethode ist, das der weg geschleuderte Sand linear abnehmend vor der Bürste zu liegen kommt. Wie in Abbildung 3.4 zu sehen ist, wird mit zunehmender Drehzahl das Streufeld des Sandes länger. Im rechten Bild ist zusätzlich eingezeichnet, wie die Verteilung des Sandes simuliert wird. Im roten Rechteck wird Sand abgetragen und im gelben verteilt.

3.3 Implementation der Simulation

Um die Algorithmen für das Glätten des Sandfeldes zu überprüfen wurde eine Umgebung benötigt, in der das Abtragen und Verschieben von Sand simuliert werden konnte. Durch die Aufnahmen der Sandkiste standen mehrere verschiedene Felder zur Verfügung, die zur Simulation verwendet werden konnten. Als Programmiersprache wurde Matlab verwendet. Um die Übersichtlichkeit der Simulation zu verbessern, wurden für die Hauptteile Structures (Siehe Kapitel**??**) erstellt, auf welche die einzelnen Funktionen zugreifen können.

Structure	Parameter
Feld	Höhe, Länge und Breite des Feldes, Umrechnungsfaktoren zu x- und y-Koordinaten, Toleranz der Unebenheit des Feldes, Grundfläche und Volumen des Feldes, mittlere Höhe
Roboter	letzte und momentane x- und y-Koordinate, zurückgelegter Weg
Bürste	Breite, Länge und Höhe der Bürste, Drehwinkel

Tabelle 3.2: Verwendete Structures und enthaltene Parameter

Zu Beginn der Simulation werden diese drei Structures initialisiert, wobei alle Parameter eingegeben oder berechnet werden. Während der Simulation kann nicht mehr direkt auf die Structures zugegriffen werden, sondern nur über Funktionen. Im folgenden werden alle wichtigen Funktionen kurz erläutert. Die Funktionen wurden so erstellt, dass sie zur Funktionsweise des Roboters passen, der im parallel laufenden Projekt gebaut wurde. Dieser Roboter hat eine rotierende Bürste, die den Sand weg schleudert. Der Ort, an dem sich der Mittelpunkt der Bürste befindet, wird in die Structure Roboter unter dem Punkt Position abgespeichert. (Anhang 7.4)

3.3.1 Feld

Initialisierung

Bei der Initialisierung wird angegeben, welches Feld geladen werden soll, die Breite und Länge des Feldes in Zentimeter, sowie die Toleranz, die das bearbeitete Feld schlussendlich zur mittleren Höhe maximal haben darf. Anhand der Länge, Breite und der Anzahl Pixel in der Matrix des Feldes werden die Umrechnungsfaktoren zwischen Zentimeter und Indizes für die Matrix berechnet. Des Weiteren werden die Grundfläche, das gesamte Volumen und die mittlere Höhe des Sandes berechnet. Die mittlere Höhe wird verwendet um festzulegen, auf welcher Höhe sich der Sand nach dem glätten, befinden soll.

Feld einlesen

Um die mit dem SICK-Laser aufgenommenen Felder einzulesen wurde eine Funktion geschrieben, die aus der Punktewolke direkt eine Matrix mit den Werten der Höhe erstellt. Dafür mussten zuerst alle gescannten Z-Werte nacheinander eingelesen und in einer Matrix gespeichert werden. Da sich an den Rändern des Scanns falsche Werte befinden, verursacht durch die Seitenwände der Sandkiste, muss das Feld jeweils noch ausgeschnitten werden. Um ein stetiges Feld zu erhalten, ohne grosse Sprünge zwischen den einzelnen Punkten, wurden die Werte noch mit einem Mittelwertfilter bearbeitet. Da sich der entscheidende Teil des Feldes zwischen dem tiefsten und dem höchsten Punkt des Feldes befindet, wurde das Feld so verschoben, dass sich der tiefste Punkt in der Nullebene befindet. Zudem wurden die Werte, welche in Millimeter gemessen wurden, in Zentimeter umgerechnet.

Feld abtragen

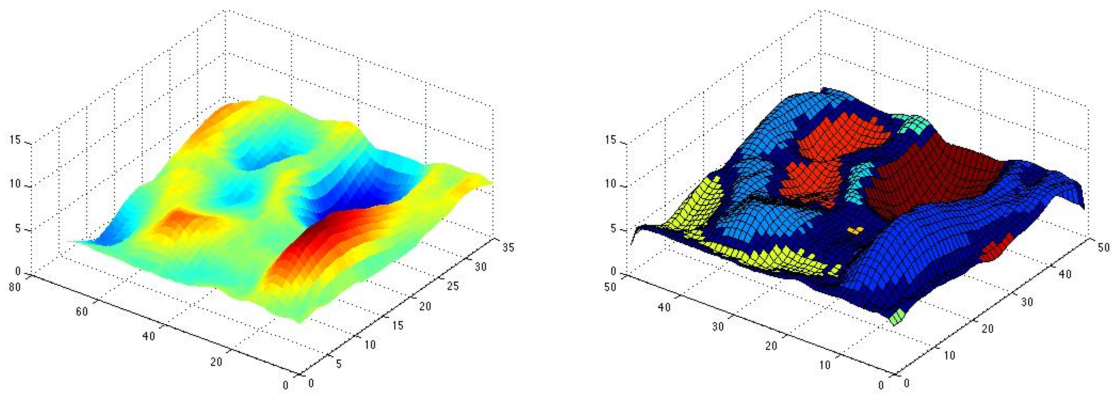
Um das Abtragen von Sand zu simulieren wurde eine Funktion erstellt, die, an der Stelle an welcher sich der Roboter befindet, die Höhe des Feldes auf die Höhe der Bürste absenkt. Die Funktion gibt das Volumen des abgetragenen Sandes zurück, gleichzeitig wird in der Structure Feld die Höhe des Feldes geändert.

Feld verteilen

Um den abgetragenen Sand zu verteilen wurden mehrere Funktionen erstellt. In einer ersten Phase wurde eine sehr einfache Verteilung, bei der der Sand an einer bestimmten Stelle auf der Fläche eines Rechteckes gleichmässig zur Höhe des Feldes dazu addiert wird. In einer zweiten Phase wurde eine Verteilung erstellt, welche den Sand zwischen dem Abtragungsort und einem bestimmbaren Ort verteilt. Der Sand wird dann gleichmässig auf der Fläche zwischen den beiden Orten verteilt. Dies simuliert das Verhalten von Sand besser, da eine Verteilung an genau einer Stelle nicht möglich ist.

Hügel und Täler finden

Um die Hügel zu finden wurde eine binäre Maske des Feldes erstellt, welche an allen Stellen, bei der das Feld grösser ist als die mittlere Höhe und die Toleranz zusammen gerechnet, eine Eins enthält. Um die einzelnen Hügel von einander trennen zu können, erhielt jeder ein eigenes Label. So kann mit einer passenden Maske, der einzelne Hügel extrahiert und so separat betrachtet werden. Ein weiterer Nutzen der Labels ist, dass die einzelnen Hügel auch separat eingefärbt werden können. Damit kann besser nachvollzogen werden, wohin der Sand eines Hügels verschoben wurde. Das selbe Verfahren wurde auch verwendet um die Täler zu finden. Dafür wurde die Maske so angepasst, dass sich überall dort eine Eins befindet, bei der das Feld einen kleineren Wert hat als die mittlere Höhe abzüglich der Toleranz.



(a) Originalfeld

(b) erkannte Hügel und Täler

Abbildung 3.5: Veranschaulichung der Methode: Hügel und Täler finden

3.3.2 Roboter

Initialisierung

Bei der Initialisierung der Structure des Roboters wird der Startort des Roboters und der gefahrene Weg auf den Anfangswert gesetzt.

Bürsten in X- und Y-Richtung

Um einen möglichst realen Bürstenvorgang zu erreichen, wurde eine kombinierte Funktion erstellt, welche von einem Startort, mit einer variablen Geschwindigkeit, in eine Richtung fährt und dabei den Sand abträgt. Dieser Sand wird dann mit einem variablen Abstand vor sich her geschoben. Um das gesamte Modell zu vereinfachen wurde nur die Bewegung in X- oder Y-Richtung zugelassen.

3.3.3 Bürste

Initialisierung

Bei der Initialisierung der Bürste wird die Breite, die Länge, die Höhe und der Drehwinkel angegeben.

3.3.4 weitere Funktionen

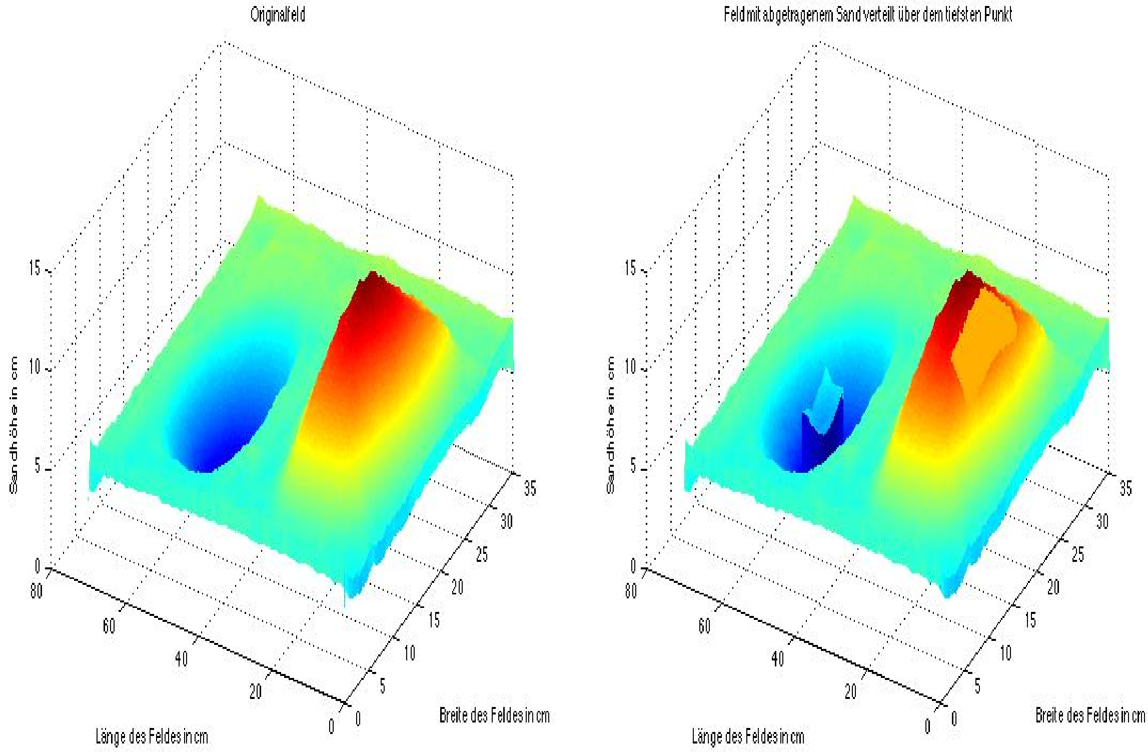
Zur Übersicht werden an dieser Stelle all jene Funktionen aufgelistet, die in den Algorithmen angewandt, jedoch an dieser Stelle nicht erläutert werden, da es sich meist um einfache Methoden handelt. Bei Interesse lassen sich die Funktionen in ihrer Gesamtheit auf der beigelegten CD finden.

- Längen in Koordinaten umrechnen und umgekehrt
- Feld ausgeben
- Höchst- und Tiefstpunkt finden
- Volumen des Feldes berechnen
- Werte von den Structure bekommen und überschreiben
- Umwandeln der Matrix in verschiedene Grössen, ändern der Anzahl Punkte
- Bewegen des Roboters auf dem Feld
- berechnen des gefahrenen Weges

3.3.5 Anwendungsbeispiel der Funktionen

```
% Setzen der Parameter des Feldes
s='/Users/lukasruckstuhl/ZHAW/Matlab Skripte/Projekte/Feldglaetter/test3';
                          % Pfad wo sich die Textfiles befinden
anz=381;                  % Anzahl Textfiles des Feldes
tol=0.5;                  % verwendete Toleranz in cm
L=70;                     % Laenge des Feldes in cm
B=35;                     % Breite des Feldes in cm
FeldInit(s,anz,B,L,tol) % Initialisierung des Feldes

% Setzen der Parameter der Buerste
lb=5;                     % Laenge der Buerste in cm
bb=10;                    % Breite der Buerste in cm
h=0;                      % Hoehe der Buerste in cm
phi=0;                    % Drehwinkel der Buerste in Grad
BuersteInit(bb,lb,h,phi)% Initialisierung der Buerste

% Setzen der Parameter des Roboters
xr=0;                 % x-Position des Roboters in cm
yr=0;                 % y-Position des Roboters in cm
weg=0;                % bisher gefahrener Weg des Roboters in cm
RobotInit(0,0,0)      % Initialisierung des Roboters

figure
FeldAusgeben()        % darstellen des originalen Feldes

% Bewegen des Roboters an die Stelle wo abgetragen werden soll
x=17;                 % x-Position des Abtrageortes
y=10.75;              % y-Position des Abtrageortes
[x,y]=cm2ko(x,y);     % Umrechnung der Position von cm in Matrixkoordinaten
RobotBewegen(x,y)     % Den Roboter an die Abtrageposition bewegen

% Bewegen der Buerste auf die Abtragehoehe
hb=10;                % Abtragehoehe in cm
setBuerste(hb,'h')    % Buerste auf die gewuenschte Abtragehoehe bewegen

vol=FeldAbtragen(); % Abtragen des Feldes
[m,x,y]=Feldmin();    % Bestimmung der tiefsten Stelle um den Sand zu verteilen
FeldVerteilenRechteck(vol,x,y,5,5)  % Verteilen des Sandes

figure
FeldAusgeben()        % darstellen des Feldes
```

In Abbildung 3.6 wird das Anwendungsbeispiel veranschaulicht. Dabei wird Sand am definierten Punkt von dem Berg abgetragen. Diese Menge wird am tiefsten Punkt des Feldes wieder platziert.



(a) Originalfeld

(b) abgetragenes Feld

Abbildung 3.6: Veranschaulichung des Anwendungsbeispiels

3.4 Ausarbeitung eines Algorithmus

Bevor mit dem Programmieren des Algorithmus begonnen werden konnte, musste überlegt werden, wie der Algorithmus dieses Problem lösen soll. Dazu wurden im Vorfeld verschiedene Szenarien überlegt, wie vorgegangen werden könnte. Dabei waren auch die eigenen Versuche mit dem Bürstenmodell und der Sandbox sehr hilfreich. Es zeichneten sich vor allem drei verschiedene Methoden ab.

Die am einfachsten zu realisierende Methode, wie sich der Roboter auf dem Sandfeld bewegen kann, ist ihn immer von einer Seite zur anderen zu schicken und bei einer Erhöhung die Bürste zu senken um Sand abzutragen. Diese Methode benötigt jedoch viel Zeit, da der Roboter auf diese Weise viel unnötigen Weg zurücklegt. Zudem müsste dieser Algorithmus oft angewendet werden, bis das Feld geglättet ist.

Eine Methode, die vor allem in den ersten erstellten Algorithmen Anwendung fand, ermittelt an welchem Ort sich die höchste und tiefste Stelle des Sandfeldes befindet. Dann wird von der höchsten Stelle Sand abgetragen und zur tiefsten verschoben. Nach diesem Vorgang werden die neuen Extremstellen ermittelt. In Abbildung 3.7 werden die ersten 10 Schritte dieser Methode aufgezeigt. Auch bei diesem Vorgehen werden noch zusätzliche und unnötige Wege abgefahren.

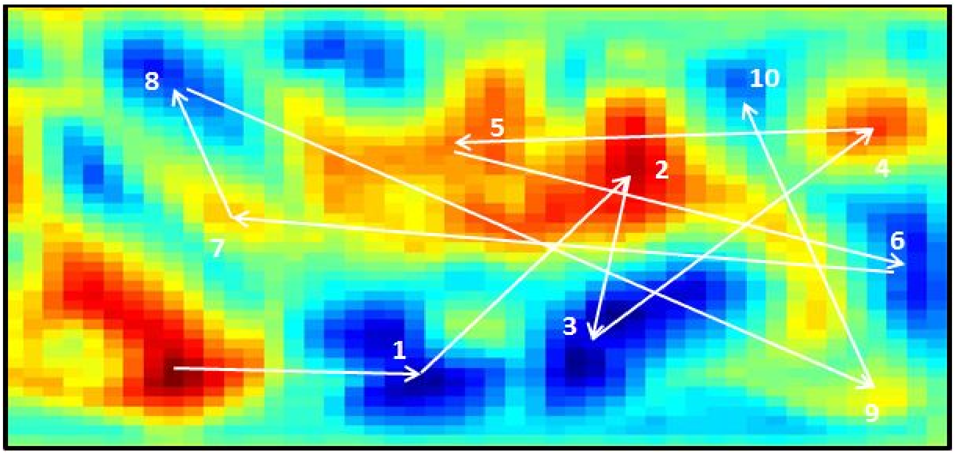


Abbildung 3.7: Methode: Von der höchsten Stelle zur tiefsten

Eine weitere Methode um Sandmengen zu verteilen, besteht darin, an einer beliebigen Stelle, an der ein Überschuss besteht, zu beginnen und den Sand zu dem nächstgelegenen Mangel zu verschieben. Danach wird wiederum der nächstgelegene Überschuss gesucht um erneut den Mangel, der sich am nächsten befindet, anzufahren. Dazwischen wird jeweils das Sandfeld neu eingescannt und die neuen Hoch- und Tiefstellen werden berechnet (Abbildung 3.8). Diese Methode wurde in den zuletzt erstellten Algorithmen angewandt. Der grosse Vorteil bei diesem Vorgehen ist, dass die abgefahrenen Strecken, bei denen kein Sand verschoben wird, insgesamt sehr klein sind. Jedoch kann es vorkommen, dass die bewegte Sandmenge viel grösser ist als das zu befüllende Loch, wodurch neue Hochstellen entstehen, die später erneut verschoben werden müssen.

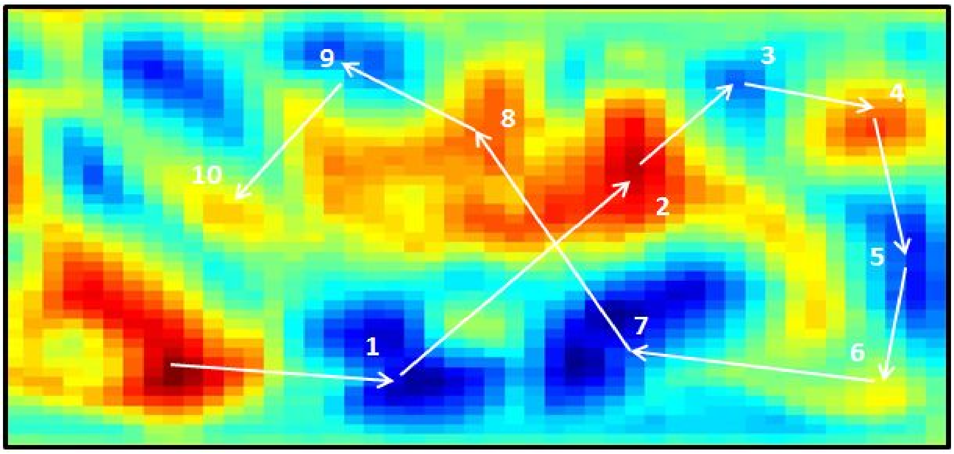


Abbildung 3.8: Methode: Next

4 Resultate

Während der Projektarbeit wurde laufend an den Algorithmen gearbeitet. Während der gesamten Dauer entstanden so viele verschiedene Programme welche Bestandteile von überarbeiteten und weiterführenden Versionen wurden. Aus diesem Grund werden an dieser Stelle nur die Algorithmen präsentiert an denen bis zum Ende gearbeitet wurde und das Sandfeld in einer nützlichen Frist glätten könnten. Um die einzelnen Algorithmen zu testen wurden sie auf die aufgenommenen Felder angewendet. Die Algorithmen wurden so lange ausgeführt, bis sich die Differenz zwischen dem höchsten und dem tiefsten Ort in einem vorbestimmten Toleranzbereich befindet. Dabei wurden die Toleranz und die abtragbare Höhe Δh variiert. Um die Ergebnisse zu bewerten wurden die Anzahl Schitte, das gesamte verschobene Volumen (GvV), das gesamthaft zu verschiebende Volumen (ZvV) und der gefahrene Weg berechnet.

4.1 Nicht kontinuierliche Glättungen

Als erstes wurden, um sich in das Thema "Glätten von Sandfeldern" einzudenken, damit begonnen nicht kontinuierlich zu glätten. Dies bedeutet, dass es möglich ist an jeder beliebigen Stelle Sand aufzutragen und zu verteilen. Auf diesem funktionierenden Algorithmus konnte dann aufgebaut werden. Dabei wurden nacheinander einschränkende Faktoren eingefügt. Diese sollten die realen Bedingungen, die man beim Glätten des Sandfeldes vorfindet, simulieren.

4.1.1 Algorithmus 1

Ziel

Der erste Algorithmus ermittelte den höchsten und den tiefsten Punkt des Feldes. Danach wurde an der Stelle des Hochpunktes Sand abgetragen. Dabei wurde nur in einem rechteckigen Bereich, der die Sandbürste darstellt, Sand entfernt. Des Weiteren wurde nur ein variabler Höhenunterschied von Δh abgetragen. Diese Einschränkung wurde gewählt, damit das verteilbare Volumen, welches beim Abtragen von Sand entstand, nicht so gross wurde, dass das Feld nicht zu glätten war. Denn wenn zu viel in einem Schritt abgetragen wird, werden die entstehenden Hügel jeweils so hoch, dass sie nie mehr in den Toleranzbereich kommen.

Auswertung

Dieser Algorithmus war in der Lage das Feld zu glätten und die Berechnungszeit war relativ kurz. Die beiden Testfelder unterscheiden sich in ihrem zu verschiebenden Volumen. Wie in Tabelle 4.3 ersichtlich war der gefahrene Weg des Roboters sehr gross, da er die ganze Zeit auf dem Feld hin und her fahren musste. Ausserdem wurde drei Mal mehr Sand verschoben als eigentlich notwendig gewesen wäre. Dieses Verhältnis verschlechtert sich wenn die Menge an Sand, welche mit einem Mal abgetragen werden kann, erhöht wird. Jedoch kann dadurch die Anzahl Schritte und der gefahrene Weg markant verkleinert werden. Das Verhältnis zwischen dem bewegten Sand verbessert sich bei kleiner werdender Toleranz geringfügig. Die Anzahl Schritte und der gefahrene Weg erhöhen sich dadurch stark.

Feld	Parameter		Ergebnis			
	Δh in [cm]	tol in [cm]	Anzahl Schritte	GvV in [cm^3]	ZvV in [cm^3]	gefahrener Weg in [m]
test 6	0.25	1	770	2’256	716.5	145
	0.5	1	421	2’619	716.5	81
	0.125	0.5	1’895	2’823	1’087	345
	0.25	0.5	1’009	3’123	1’087	195
test 7	0.25	1	242	610	47.6	53
	0.5	1	136	797	47.6	31
	0.125	0.5	762	1’000	202	158
	0.25	0.5	415	1’204	202	92

Tabelle 4.1: Ergebnisse des Algorithmus 1, uneingeschränkt

4.1.2 Algorithmus 2

Ziel

Als erste Verbesserung wurde der Verteilbereich des Algorithmus eingeschränkt. Der Verteilbereich wurde so gewählt, dass der Sand nur noch innerhalb des Feldes verteilt werden konnte. Somit sollte verhindert werden, dass der Sand das Feld verlässt.

Auswertung

Mit der Einschränkung des Algorithmus konnte beim ersten Testfeld der Aufwand merklich reduziert werden, da bei diesem Feld viel Sand verschoben werden muss. Jedoch musste festgestellt werden, dass die Ergebnisse weiterhin nicht zufriedenstellend ausfielen. Beim zweiten Testfeld waren die Ergebnisse in der selben Grössenordnung wie bei Algorithmus 1. Es wird davon ausgegangen, dass dies an der geringen Menge von Sand, die verschoben wird, liegt.

Feld	Parameter		Ergebnis			
	Δh in [cm]	tol in [cm]	Anzahl Schritte	GvV in [cm^3]	ZvV in [cm^3]	gefahrener Weg in [m]
test 6	0.25	1	656	2’157	716.5	120
	0.5	1	364	2’632	716.5	67
	0.125	0.5	1’580	2’741	1’087	286
	0.25	0.5	855	3’023	1’087	160
test 7	0.25	1	232	612	47.6	51
	0.5	1	138	839	47.6	31
	0.125	0.5	731	1’020	202	150
	0.25	0.5	406	1’234	202	88

Tabelle 4.2: Ergebnisse des Algorithmus 2, Verteilung nur im Feld

4.1.3 Algorithmus 3

Ziel

Um die gefahrene Strecke des Roboters zu verkleinern, sollte verhindert werden, dass der Roboter nach jedem abtragen seine Position stark ändern muss. Dazu wurde die Methode zum detektieren von Hügeln verwendet. Danach wurden alle gefundenen Hügel abgetragen und an der tiefsten Stellen verteilt. Danach wurden alle neuen Hügel detektiert und wieder abgetragen. Dies wird so lange wiederholt, bis sich die Feldhöhe im Toleranzbereich befindet.

Auswertung

Bei diesem Algorithmus wurde einerseits festgestellt, dass die gewählte Bestimmung der Toleranz, mittlere Höhe +/- Toleranz in Zentimeter, nicht optimal ist. Wenn die Toleranz falsch gewählt wurde, konnte das Feld nicht mehr geglättet werden. Da die obere Toleranzgrenze, Höhe bis zu der abgetragen werden soll, bereits erreicht war, bevor die Täler des Feldes gefüllt waren. Es wurde versucht dieses Problem zu lösen, jedoch konnte keine allgemeine Lösung gefunden werden. Aus diesem Grund wurde die Grenze, bis zu welcher die Bürste abtragen kann manuell eingestellt. So konnte erreicht werden das der Algorithmus das Feld glättet um vergleichbare Ergebnisse zu erhalten. So konnte das gewünschte Ziel erreicht werden. Der gefahrene Weg des Roboters halbierte sich.

Feld	Parameter		Ergebnis			
	Δh in [cm]	tol in [cm]	Anzahl Schritte	GvV in [cm^3]	ZvV in [cm^3]	gefahrener Weg in [m]
test 6	0.25	1	550	2’009	716.5	36
	0.5	1	281	1’985	716.5	23
	0.125	0.5	2008	2’593	1’087	170
	0.25	0.5	1029	2’662	1’087	91
test 7	0.25	1	139	627	47.6	8
	0.5	1	111	925	47.6	7.5
	0.125	0.5	841	1’380	202	64
	0.25	0.5	474	1’580	202	39

Tabelle 4.3: Ergebnisse des Algorithmus 3, Hügelweise abtragen

4.2 kontinuierliche Glättung

4.2.1 Algorithmus 4

Dieser Algorithmus entstand nach den Versuchen mit dem Bürstenmodell im Sandfeld. Dabei wurde erkannt, dass der verschobene Sand sich, linear mit dem Abstand zur Bürste, abnehmend vor der Bürste verteilt.

Ziel

Bei diesem Algorithmus sollte ein erstes Mal versucht werden das Abtragen und das Verteilen des Sandes zu kombinieren. Sodass der abgetragene Sand gerade anschliessend wieder verteilt wird. Der Algorithmus sucht dann die höchste und die tiefste Stelle im Feld. Die Bürste wird auf eine maximale Abtragehöhe eingestellt und hinter der höchsten Stelle positioniert. Die Bürste bewegt sich in Richtung der tiefsten Stelle und schiebt den Sand vor sich her. Dabei nimmt sie von allen Stellen Sand mit, die sich über der Abtragehöhe befinden. Danach wird das Feld erneut gescannt und es werden erneut die höchsten und tiefsten Stellen lokalisiert. Dies wird so lange durchgeführt bis der Toleranzbereich erreicht wird.

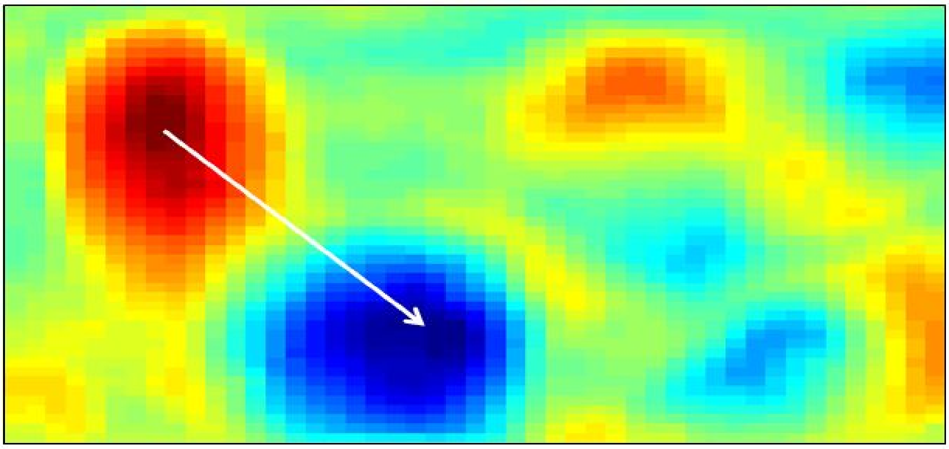


Abbildung 4.1: Vorgehen des Algorithmus 4

Auswertung

Um das Programm zu vereinfachen wurden nur Fahrten in X- und Y-Richtung zugelassen. Hierbei ist die X-Richtung, auf einem realen Feld, parallel zum Netz und die Y-Richtung senkrecht dazu. Der Algorithmus glättete das Feld wie gewünscht, jedoch ging viel Sand verloren. Weiter wurde festgestellt, dass die Bürste teilweise sehr grosse Mengen an Sand vor sich her schiebt. Dies ist nicht mit der Realität zu vergleichen. In einer ersten Verbesserung sollte verhindert werden, dass die Bürste den Sand ausserhalb des Feldes verteilt. Um dies zu verhindern wurde versucht den Aktionsbereich des Roboters einzuschränken, dabei wurde erkannt, dass es in diesem Fall nicht immer möglich ist alle Bereiche des Feldes zu erreichen. Das führte dazu, dass der Algorithmus das Feld nicht glätten kann.

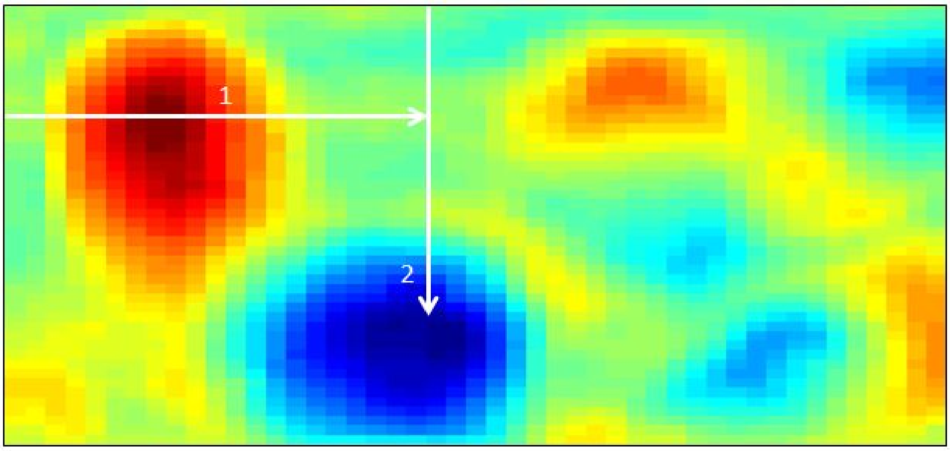


Abbildung 4.2: Programmiertes Vorgehen beim Algorithmus 4

5 Diskussion und Ausblick

5.1 Rückblick und Fazit

Dieses Projekt war für uns Studenten eine sehr spannende und herausfordernde Aufgabe. Es gab uns zum ersten mal die Gelegenheit uns an einem Algorithmus zu versuchen. Gleichzeitig wurde uns viel Spielraum gelassen, die an uns gestellten Aufgaben zu bewältigen, so dass wir unsere persönliche Herangehensweise in das Projekt einfließen lassen konnten.

Aufbau der Versuchsumgebung

Die Versuchsumgebung bestand aus einer zusammengeschraubten Holzbox, welche mit Sand befüllt wurde. Sie diente als Modell bei der Feldaufnahme durch den Laserscanner und als Testgelände für das Bürstenmodell. Beim Aufbau der Versuchsumgebung konnte unsere Neigung zur praktischen Arbeiten voll ausgelebt werden. Als Systemtechniker entschieden wir uns, einen Weg einzuschlagen der von Studenten anderer Studienrichtungen, bei dieser Aufgabenstellung, vielleicht weniger eingeschlagen worden wäre. Doch die erstellten Modelle halfen uns beim hineindenken in diese Arbeit.

Programmierung des Algorithmus

Es wurden im Verlauf der Projektarbeit viele kleinere Algorithmen erstellt die sich mit spezifischen Teilproblemen befassten. Auf diese kleineren Arbeiten wurde im Projekt nicht weiter eingegangen, da die Erkenntnisse oder sogar ganze Fragmente dieser Programme sich in den grösseren Algorithmen, die gegen Ende des Projekts ausgearbeitet wurden, wiederfinden. Diese Algorithmen verfolgten speziell definierte Ziele und wurden im Verlaufe des Projekts immer weiter entwickelt und abgeändert. Bei diesen verschiedenen Ansätzen ergaben sich auch immer wieder verschiedene Probleme und Erkenntnisse, dies war der Grund diese komplexeren Algorithmen nebeneinander weiter zu entwickeln und nicht wieder zu einem einzigen zu kombinieren.

Beim programmieren des Algorithmus, versuchten wir Anfangs zu viele Schritte auf einmal umzusetzen. Wir mussten uns daran gewöhnen unseren Ideenfluss Schritt für Schritt zu Implementieren, um uns nicht hoffnungslos in Fehlern zu verstricken. Doch war es spannend aus kleinen und einfachen Algorithmen heraus immer komplexere entstehen zu lassen. Gerne hätten wir noch mehr Zeit für diese Aufgabe verwendet.

Fazit

Wie aus den Resultaten ersichtlich, wurde in dieser ersten Arbeit noch kein abschliessender Algorithmus für dieses komplexe Problem realisiert. Jedoch konnten viele Lösungen für die beinhalteten Teilprobleme entwickelt und wichtige Fragen geklärt werden. Die erstellten Algorithmen sind durchaus in der Lage, das Ziel, ein Beachvolleyballfeld zu glätten, zu erfüllen. Jedoch würde ein Roboter viel Zeit benötigen um die Glättung durchzuführen. Des Weiteren ist der Unterschied zwischen Realität und Simulation noch relativ gross, somit ist eine direkte Umsetzung des Fahrplanes noch nicht möglich.

Es wurde eine Methode erstellt, wie in einer Laborumgebung die Oberfläche eines Sandfeldes eingescannt werden kann. Diese Methode könnte auch für ein reales Beachvolleyball verwendet werden, dabei gestaltet sich jedoch die Positionierung des Laserscanners schwierig. Mit Hilfe dieser Methode konnte eine Bibliothek mit Aufnahmen von mehreren Feldern angelegt werden, welche bei weiteren Arbeiten dienlich sind. Ausserdem kann sie für weitere Aufnahmen von Feldern in späteren Projekten verwendet werden. Mit dem Bürstenmodell konnten wichtige Informationen gesammelt werden bezüglich des Verhaltens von Sand. In einem weiteren Schritt könnte dieses Modell an einen der Werkstattroboter montiert werden, um so die ausgearbeiteten Algorithmen zu Prüfen.

Bei der Programmierung wurden die funktionierenden Unterfunktionen jeweils als eigenständige Methoden ausprogrammiert. Somit konnte die Übersichtlichkeit gesteigert werden und der Zugriff wurde vereinfacht. So wurden viele Werkzeuge geschaffen die bei einem wieder aufgreifen dieser Arbeit verwendet werden können.

5.2 Ausblick

In diesem Kapitel werden die weiteren Schritte welche zu unternehmen sind dargestellt.

Ausarbeiten des Algorithmus

Die präsentierten Algorithmen sind erste Ansätze wie es möglich ist ein Sandfeld zu glätten. Sie besitzen jedoch noch viel Optimierungspotenzial. Einerseits sollte in einem nächsten Schritt die Sandverteilung genauer untersucht werden. Dabei gilt es vor allem zu beachten, in welchem Bereich der vorhandene Roboter Sand bewegen kann und wie sich der Sand in der Realität verteilt. Bisher wurde die Höhe der Bürste immer relativ zu der durchschnittlichen Höhe oder der Nullebene angegeben. Um ein realistischeres Verhalten zu bekommen sollte ein Ansatz überprüft werden, bei dem die Bürstenhöhe relativ zur Standhöhe des Roboters ist.

Da während dieser Projektarbeit noch auf keinen fertigen Sandverteilungsroboter zugegriffen werden konnte, wurde auch noch an keiner Implementierung für den verwendeten Roboter gearbeitet. Dies würde sich in einem nächsten Schritt jedoch lohnen, da so viele weitere Erkenntnisse gewonnen werden könnten.

Weitere Versuche

Die Versuchsumgebung wurde darauf ausgelegt, zu einem späteren Zeitpunkt das Bürstenmodell an den Roboter zu montieren und so den Algorithmus in der Sandbox zu testen. Aus zeitlichen Gründen wurde in dieser Arbeit darauf verzichtet. Für Demonstrationszwecke könnte dies lohnend sein.

Anpassen der Programmierumgebung

Das Programmieren des Algorithmus wurde während dieser Arbeit auf einer Matlab Umgebung realisiert, dabei wurde festgestellt das bei den komplexeren Programmen, lange Rechenzeit benötigt wurden um die Sandverschiebungen zu visualisieren. In einem weiteren Schritt wäre es empfehlenswert auf eine Hochsprache wie C, C++ oder Java umzusteigen um das auswerten des erstellten Programmes zu beschleunigen.

Weitere Aufnahmemöglichkeiten

Die von uns verwendete Methode, um die Oberfläche des Sandfeldes aufzunehmen, hat für unsere Zwecke zufriedenstellende Ergebnisse geliefert, jedoch ist es nicht die einzige Möglichkeit, auswertbare Bilder der Oberfläche zu erzeugen. Bei einer weiterführenden Arbeit wäre es spannend noch weitere Aufnahmemöglichkeiten zu testen, besonders im Hinblick darauf, gewisse Methoden auch direkt an den Roboter anzubringen.

6 Verzeichnisse

Literaturverzeichnis

[1] Gisela Engeln-Müllges, Klaus Niederdrenk, and Reinhard Wodicka. Numerik-algorithmen. *Numerik-Algorithmen: Verfahren, Beispiele, Anwendungen, Xpert. press, ISBN 978-3-642-13472-2. Springer-Verlag Berlin Heidelberg, 2011*, 1, 2011.

[2] Doina Logofătu. *Algorithmen und Problemlösungen mit C++: von der Diskreten Mathematik zum fertigen Programm-Lern-und Arbeitsbuch für Informatiker und Mathematiker*. Springer DE, 2006.

[3] Thomas Ottmann and Peter Widmayer. *Algorithmen und Datenstrukturen*. Springer DE, 2012.

[4] Hartley Rogers Jr. *Theory of recursive functions and effective computability*. MIT press, 1987.

[5] Markus von Rimscha. *Algorithmen Kompakt Und Verstandlich: Lösungsstrategien Am Computer*. Springer DE, 2009.

[6] Stephen J Chapman. *MATLAB programming for engineers*. Cengage Learning, 2008.

[7] Der Kaiser von China (17.08.2007) [Online]. URL: http://china.plasmazone.de/archives/date/2007/08/17. [Stand:18.12.2013].

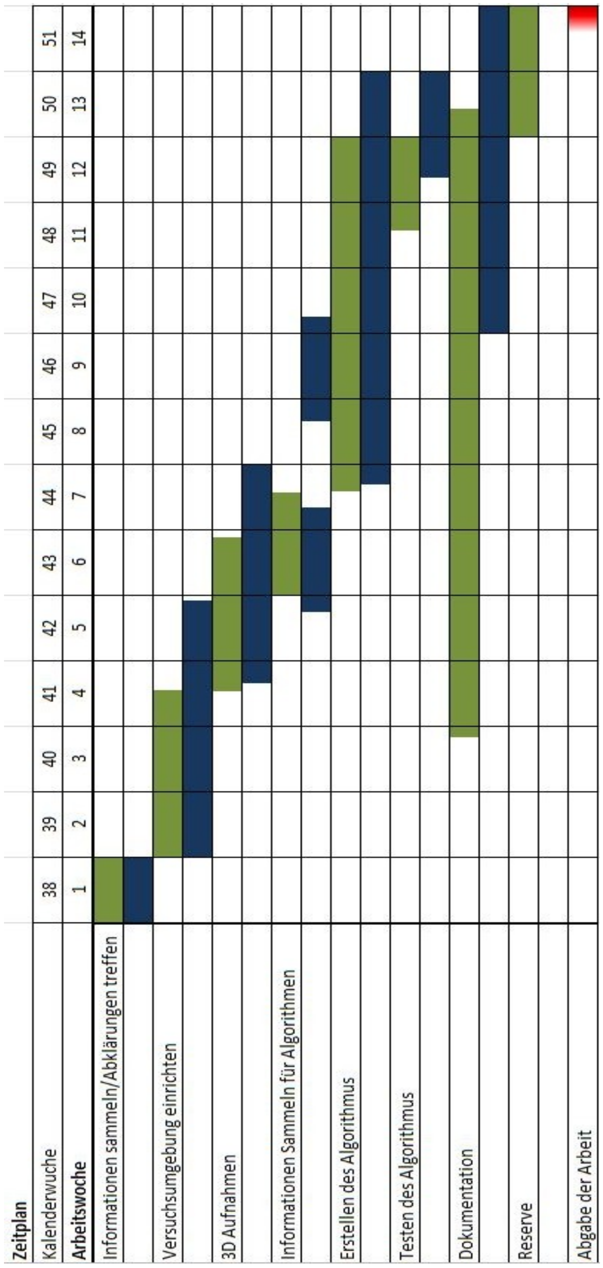
[8] Pferdebetrieb. Das Profi-Magazin(05.07.2010) [Online]. URL: http://www.pferde-betrieb.de/glatt-oder-gequirlt-reitbahnpflege-mit-den-neuen-bahnplanern. [Stand:18.12.2013].

Abbildungsverzeichnis

Tabellenverzeichnis

7 Anhang

7.1 Zeitplan



7.2 Materialliste

Stk	Sandbox
1	Bodenplatte aus Sperrholz 382 x 732 x 5mm
2	Seitenwände Länge aus Sperrholz 100 x 382 x 5 mm
2	Seitenwände Breite aus Sperrholz 100 x 700 x 5 mm

Stk	Bürste
1	Aluminium Winkel
2	Seitenwände aus Aluminium 100 x 382 x 5 mm
2	iGus Gleitlager d = 12
1	Aluminium Stange d = 12
1	Zahnriemenrad 18 Zähne
1	Zahnriemenrad 25 Zähne
1	Zahnriemen Breite 6 mm Wirklänge 200

7.3 Analysierte Algorithmen

7.3.1 Algo1

```
%% [i, volges]=algo1(L, B, tol, dh, mm)
% erster Algorithmus
% Traegt die hoechste Stelle ab und verteilt es an der tiefsten Stelle.
%
% 'L'        Laenge des Rechteckes auf welchem der Sand verteilt wird in cm
% 'B'        Breite des Rechteckes auf welchem der Sand verteilt wird in cm
% 'tol'      Toleranz um wieviel das Feld von der mittleren Hoehe abweichen darf
% 'dh'       Hoehendifferenz welche auf einmal abgetragen werden kann
% 'mm'       gibt das Intervall an in welchem das Feld ausgegeben wird
%
% Rueckgabewerte
%
% 'i'        Anzahl gemachte Schritte
% 'volges'   gesamtes verschobenes Volunmen
%

function [i, volges]=algo1(L, B, tol, dh, mm)

    i=0;
    volges=0;

    while Feldmax()-Feldmin()>tol
        i=i+1;
        [m,x,y]=Feldmax();
        %xx(i)=x;
        %yy(i)=y;
        RobotBewegen(x,y)
        setBuerste(m-dh,'h')
        vol=FeldAbtragen();
        volges=volges+vol;

        [m,x2,y2]=Feldmin();
        FeldVerteilenRechteck(vol,x2,y2,L,B);
        if mod(i,mm)==0
            FeldAusgeben()
        end
    end

end
```

7.3.2 Algo2

```
%% [i, volges]=algo2(L, B, tol, dh, mm)
% erster Algorithmus
% Traegt die hoechste Stelle ab und verteilt es an der tiefsten stelle. Es
% wird ausserdem darauf geachtet, dass kein Sand das Feld verlaesst.
%
% 'L'       Laenge des Rechteckes auf welchem der Sand verteilt wird in cm
% 'B'       Breite des Rechteckes auf welchem der Sand verteilt wird in cm
% 'tol'     Toleranz um wieviel das Feld von der mittleren Hoehe abweichen darf
% 'dh'      Hoehendifferenz welche auf einmal abgetragen werden kann
% 'mm'      gibt das Intervall an in welchem das Feld ausgegeben wird
%
% Rueckgabewerte
%
% 'i'       Anzahl gemachte Schritte
% 'volges'  gesamtes verschobenes Volunmen
%

function [i, volges]=algo2(L, B, tol, dh, mm)
    % L:        Laenge des Rechteckes auf welchem Sand verteilt wird
    % B:        Breite des Rechteckes auf welchem Sand verteilt wird
    % tol:      Toleranz der Unebenheit in cm
    % dh:       Abtraghoehe die die Buerste mit einem Mal abtragen kann
    % m:        gibt die Frequenz der Ausgaben an. jeder m-te Schritt wird
    %           ausgegebn
    %
    % i:        Anzahl gemachte Schritte
    % volges:   gesamtes verschobenes Volunmen

    i=0;
    volges=0;


    while Feldmax()-Feldmin()>tol
        i=i+1;
        [m,x,y]=Feldmax();
        [x, y]=imFeld(x,y);

        RobotBewegen(x,y)
        setBuerste(m-dh,'h')
        vol=FeldAbtragen();
        volges=volges+vol;
        [m,x2,y2]=Feldmin();
        [x2, y2]=imFeld(x2,y2);
        FeldVerteilenRechteck(vol,x2,y2,L,B);
        if mod(i,mm)==0
             FeldAusgeben()
        end
    end

end

function [x, y]=imFeld(x,y)
```

```
    feld=getFeld('feld');
    [m, n]=size(feld);
    bb=getBuerste('b');
    bl=getBuerste('l');
    xmin=bb;
    xmax=n-bb;
    ymin=bl;
    ymax=m-bl;

    if x<xmin
        x=xmin-2;
    end
    if x>xmax
        x=xmax+2;
    end
    if y<ymin
        y=ymin-2;
    end
    if y>ymax
        y=ymax+2;
    end
end
```

7.3.3 Algo3

```
%%  [p, volges]=algo3(L, B, dh, mm)
% Das Feld wird geglaettet, indem ein Huegel nach dem anderen abgetragen
% wird.
%
% 'L'       Laenge des Rechteckes wo Sand abgetragen wird
% 'B'       Breite des Rechteckes wo Sand abgetragen wird
% 'dh'      Schritte der Buerste, wieviel Sand auf einmal abgetragen wird
% 'mm'      jedes mm-te Feld wird ausgegeben
%
% Rueckgabewerte
%
%  'p'        Anzahl gemachte Schritte
%  'volges'   gesamtes verschobenes Volunmen

function [p, volges]=algo3(L, B, dh, mm)
    tol=getFeld('tol');
    hm=getFeld('hm');
    h=hm-0.05;%0.15;        % musste angepasst werden, da das Feld ansonsten nicht
                            % glaetten war.
    volges=0;
    p=0;

    while Feldmax()-Feldmin()>2*tol

        [hmaxh, bh, lh, anzxh, anzyh, xstarth, ystarth, Sxh, Syh]=HuegelExtrahieren
            ();
        [hmin, bt, lt, xstartt, ystartt, Sxt, Syt]=TalExtrahieren();

        %p=p+1;

        for l=1:length(hmaxh)
            hmax=hmaxh(l);
            anzx=anzxh(l);
            anzy=anzyh(l);
            xstart=xstarth(l);
            ystart=ystarth(l);
            anzh=round((hmax-h)/dh);

            for k=1:anzh
                if k==anzh
                    setBuerste(h-0.01,'h')
                else
                    setBuerste(hmax-dh*k,'h')
                end

                for i=1:anzx
                    x=xstart+round(B/2-0.5)+(i-1)*B;
                    for j=1:anzy
                        y=ystart+round(L/2-0.5)+(j-1)*(L-1);
                        RobotBewegen(x,y)
                        vol=FeldAbtragen();
                        volges=volges+vol;
```

```
                        [m,x2,y2]=Feldmin();
                        [x2, y2]=imFeld(x2, y2);
                        FeldVerteilenRechteck(vol,x2,y2,L,B);
                        p=p+1;
                        if mod(p,mm) == 0
                           FeldAusgeben()
                           drawnow;
                        end
                    end
                end
            end
        end


    end
end

function [hmin, bt, lt, xstart, ystart, Sx, Sy]=TalExtrahieren()
    [Ht, Lt, anz]=FeldFindTal();
    [bb, lb]=cm2ko(getBuerste('b'), getBuerste('l'));
    hm=getFeld('hm');
    tol=getFeld('tol');
    bf=getFeld('b');
    lf=getFeld('l');
    feld=getFeld('feld');
    feld=feld(2:end-1,2:end-1);
    ht=hm-tol;
    tal=zeros(size(Ht));
    hmin=ones(1,anz);
    Vt=ones(1,anz);
    xstart=ones(1,anz);
    xend=ones(1,anz);
    ystart=ones(1,anz);
    yend=ones(1,anz);
    bt=ones(1,anz);
    lt=ones(1,anz);
    anzx=ones(1,anz);
    anzy=ones(1,anz);
    Sx=ones(1,anz);
    Sy=ones(1,anz);

    for i=1:anz

        tal(:,:,i)=feld.*Ht(:,:,i);
        tal(logical(Ht(:,:,i)))=ht-tal(logical(Ht(:,:,i)));
        hmin(i)=ht-max(max(tal(:,:,i)));
        Vt(i)=volumen(tal(:,:,i),lf,bf);
        [y, x]=find(Ht(:,:,i)>0);
        xstart(i)=min(x);
        xend(i)=max(x);
        bt(i)=xend(i)-xstart(i);
        ystart(i)=min(y);
        yend(i)=max(y);
        lt(i)=yend(i)-ystart(i);
        anzx(i)=round(bt(i)/bb+0.5);
        anzy(i)=round(lt(i)/lb+0.5);
```

```
        [ys, xs]=find(tal(:,:,i)>0);

        Sx(i)=mean(xs);
        Sy(i)=mean(ys);
    end
end

function [hmax, bh, lh, anzx, anzy, xstart, ystart, Sx, Sy]=HuegelExtrahieren()
    % Extrahierung und Berechnung des Huegels
    [Hh, Lh, anz]=FeldFindHuegel();
    [bb, lb]=cm2ko(getBuerste('b'), getBuerste('l'));
    hm=getFeld('hm');
    tol=getFeld('tol');
    bf=getFeld('b');
    lf=getFeld('l');
    feld=getFeld('feld');
    feld=feld(2:end-1,2:end-1);
    hh=hm+tol;
    hugel=zeros(size(Hh));
    hmax=ones(1,anz);
    Vh=ones(1,anz);
    xstart=ones(1,anz);
    xend=ones(1,anz);
    ystart=ones(1,anz);
    yend=ones(1,anz);
    bh=ones(1,anz);
    lh=ones(1,anz);
    anzx=ones(1,anz);
    anzy=ones(1,anz);
    Sx=ones(1,anz);
    Sy=ones(1,anz);

    for i=1:anz

        hugel(:,:,i)=feld.*Hh(:,:,i);
        hmax(i)=max(max(hugel(:,:,i)));
        hugel(logical(Hh(:,:,i)))=hugel(logical(Hh(:,:,i)))-hh;
        Vh(i)=volumen(hugel(:,:,i),lf,bf);
        [y, x]=find(Hh(:,:,i)>0);
        xstart(i)=min(x);
        xend(i)=max(x);
        bh(i)=xend(i)-xstart(i);
        ystart(i)=min(y);
        yend(i)=max(y);
        lh(i)=yend(i)-ystart(i);
        anzx(i)=round(bh(i)/bb+0.5);
        anzy(i)=round(lh(i)/lb+0.5);
        [ys, xs]=find(hugel(:,:,i)>0);

        Sx(i)=mean(xs);
        Sy(i)=mean(ys);
    end

end

function [x, y]=imFeld(x,y)
```

```
    feld=getFeld('feld');
    [m, n]=size(feld);
    bb=getBuerste('b');
    bl=getBuerste('l');
    xmin=bb;
    xmax=n-bb;
    ymin=bl;
    ymax=m-bl;

    if x<xmin
        x=xmin-2;
    end
    if x>xmax
        x=xmax+2;
    end
    if y<ymin
        y=ymin-2;
    end
    if y>ymax
        y=ymax+2;
    end
end
```

7.3.4 Algo4

```
%% [i, volges]=algo4( tol, dh, mm, spritz)
% 'tol'     Toleranz der Unebenheit in cm.
% 'dh'      Abtraghoehe die die Buerste mit einem Mal abtragen kann.
% 'm'       jedes mm-te Feld wird ausgegeben
% 'sp'      Wie weit der Sand von der Buerste geschleudert wird.
%
% Rueckgabewerte
%
% 'i'       Anzahl gemachte Schitte
% 'volges'  gesamtes verschobenes Volumen

function [i, volges]=algo4(tol, dh, mm, sp)
    i=0;
    volges=0;
    while Feldmax()-Feldmin()>tol

    %while i < 10
        i = i+1;
        [max,x1,y1]=Feldmax();
        [min,x2,y2]=Feldmin();
         if (max <= getBuerste('h'))
             setBuerste (getBuerste('h')-dh,'h');
         else

            voly=moveroboY (x1,y1,y2,sp);
            volx=moveroboX (x1,x2,y2,sp);
            volges=volges+volx+voly;
            if mod(i,mm) == 0
                FeldAusgeben()
                axis equal
                drawnow
                pause(0.2);
            end
        end
    end

end


function volges=moveroboY (x1,y1,y2,spti)
    [xverschieb,yverschieb]=ko2cm(0,2.5);
    [xzus,yzus]=cm2ko(0,2.5+spti/2);
    yend = false;
    volges=0;
    if y1 < y2-(spti/5)
        y1 = 0;
        while y1 <= y2-(spti/2)
            y1 = y1 + yverschieb;
             if y1+yzus+spti/2 > 50
                y1 = 50 - yzus - spti/2;
                yend = true;
             end
```

```
            RobotBewegen(x1,y1);
            vol=FeldAbtragen();
            volges=volges+vol;
            FeldVerteilenRechteck(vol,x1,y1+yzus,spti,10);
             if yend == true
                y1 = y2-(spti/2)+1;
             end
        end

    else
        y1 = 50;
        while y1 >= y2+(spti/2)
            y1 = y1 - yverschieb;
            if y1-yzus-spti/2 < 0
               y1 = 0 + yzus + spti/2;
               yend = true;
            end
            RobotBewegen(x1,y1);
            vol=FeldAbtragen();
            volges=volges+vol;
            FeldVerteilenRechteck(vol,x1,y1-yzus,spti,10);
            if yend == true
               y1 = y2+(spti/2)-1;
            end
        end
    end
end

function volges=moveroboX (x1,x2,y2,spri)

    setBuerste(90,'winkel')
    [xzus,ynew]=cm2ko(2.5+spri/2,0);
    [xverschieb,ybuffer]=ko2cm(2.5,0);
    xend = false;
    volges=0;
    if x1 < x2
        x1 = 0;
        while x1 <= x2-(spri/2)
            x1 = x1 + xverschieb;
             if x1+xzus+spri/2 > 50
               x1 = 50 - xzus - spri/2;
               xend = true;
            end
            RobotBewegen(x1,y2);
            vol=FeldAbtragen();
            volges=volges+vol;
            FeldVerteilenRechteck(vol,x1+xzus,y2,10,spri);
            if xend == true
               x1 = x2-(spri/2)+1;
            end
        end
    else
        x1 = 50;
        while x1 >= x2+(spri/2)
            x1 = x1 - xverschieb;
             if x1-xzus-spri/2 < 0
```

```
                x1 = 0 + xzus + spri/2;
                xend = true;
            end
            RobotBewegen(x1,y2);
            vol=FeldAbtragen();
            volges=volges+vol;
            FeldVerteilenRechteck(vol,x1-xzus,y2,10,spri);
            if xend == true
                x1 = x2+(spri/2)-1;
            end
        end
    end
     setBuerste(90,'winkel')
end
```

7.4 Functions

7.4.1 Feld

Feld initialisieren

```
%% FeldInit(s,anzFiles,b,l,tol)
% das Feld initialisieren. Es wird angegeben welches Feld eingelesen werden
% soll, aus wie vielen txt-Dateien das Feld besteht, wie breit und wie lang
% das Feld in echt ist. alle Einheiten sind in cm.
%
% umrx und umry werden dazu verwendet um Laengen und Breiten von cm zu
% Koordinaten und umgekehrt umzurechnen.
%
% 's'           Pfad welcher zum abgespeicherten Feld fuehrt.
% 'anzFiles'    Anzahl Textfiles aus welchen das Feld besteht.
% 'b'           Breite des Feldes in cm.
% 'l'           Laenge des Feldes in cm.
% 'tol'         Toleranz, Abweichung von der durchschnittlichen Hoehe in cm
%

function FeldInit(s,anzFiles,b,l,tol)
    global feld
    feld.h=FeldEinlesen(s,anzFiles);    % in cm
    feld.l=l;                           % in cm
    feld.b=b;                           % in cm
    feld.umrx=size(feld.h,2)/feld.b;    % in 1/cm
    feld.umry=size(feld.h,1)/feld.l;    % in 1/cm
    feld.tol=tol;                       % in cm
    feld.A=l*b;                         % Grundflaeche des Feldes in cm^2
    feld.V=FeldVolumen();               % gesamtes Volumen des Feldes in cm^3
    feld.hm=feld.V/feld.A;              % mittlere Hoehe des Feldes
end
```

Feld: Einlesen

```
%% M=FeldEinlesen(sPfad,l)
% Funktion zum erstellen einer Matrix aus den
% aufgenommenen Punktewolken. Das Feld ist so skaliert, dass der
% tiefste Punkt bei z=0 ist. Die Einheit der Hoehe ist in cm
%
% sPfad: den Pfad ,als String, wo sich die Punktewolke
%        befindet, ohne / am Ende
% l:     Anzahl der Textfiles/Punktlinien
%
% Der Feldrahmen wird nur bei 153 oder 381 Textfiles abgeschnitten,
% da sonst die Raender nicht stimmen. Fuer neue Messungen werden diese
% Ausschnittparameter neu berechnet werden muessen.

function M=FeldEinlesen(sPfad,l)

    ymin=-400;
    dd=5;
    ymax=360;
    x0=340;
    z0=450;
    anzP=421;
    zarr=zeros(l,anzP);

    minx=0;
    maxx=0;
    miny=0;
    maxy=0;
    shit=false;         % benoetigt, falls der erste Vektor ein Null-Vektor ist

    for i=1:l
        s = sprintf ('/scan%03d.txt', i);
        s1 = [sPfad s];
        f = fopen (s1, 'r');
        a = fscanf (f, '%f', [3 inf]);
        fclose (f);

        % Fuellen der z-Matrix

        if size(a)==0

        else
            if length(a(1,:))==anzP
                zarr(i,:)=a(2,:)+356;
            else
                if i==1
                    shit=true;
                else
                    zarr(i,:)=zarr(i-1,:);
                end
            end
        end

```

```
        if shit
            zarr(1,:)=zarr(2,:);
        end

    end

    % extrahieren des Feldes, Rand des Sandkastens entfernen.

    if l==153
        zarr=zarr(3:144,21:410);
    end
    if l==381
        zarr=zarr(8:355,17:410);
    end

    H=fspecial('average',6);
    M=imfilter(zarr,H);

    mini=min(min(M));
    M=M-mini;
    M=M/10;

end
```

Feld: Abtragen

```
%% vol=FeldAbtragen()
% Traegt an der Stelle an welcher sich der Roboter befindet Sand ab, bis zu
% der Hoehe auf welcher sich die buerste befindet. Dabei wird das
% abgetragene Volumen berechnet und zurueckgegeben. Wenn sich die buerste
% zu einem Teil ausserhalb des Feldes befindet wird dort auch kein Sand
% abgetragen.

function vol=FeldAbtragen()
    % neu fuer structure

    [x0, y0]=cm2ko(getRobot('x'), getRobot('y'));
    zh=getBuerste('h');         % hoehe auf welche das feld heruntergesetzt werden
        soll
    Feld=getFeld('feld');

    [b, l]=cm2ko(getBuerste('b'), getBuerste('l'));

    if mod(l,2)==1
        l=l-1;
    end
    if mod(b,2)==1
        b=b-1;
    end

    ll=l;
    bb=b;

    [n m]=size(Feld);
    [y x]=meshgrid(0:1:m,0:1:m);

    % verifizieren ob sich die buerste komplett im Feld befindet
    i=0;
    j=0;
    if x0-b/2<0
        i=1;
        b=b-abs(x0-b/2);
    end
    if x0+b/2>size(Feld,2)
        i=2;
        b=b/2+size(Feld,2)-x0;
    end
    if y0-l/2<0
        j=3;
        l=l-abs(y0-l/2);
    end
    if y0+l/2>size(Feld,1)
        j=4;
        l=l/2+size(Feld,1)-y0;
    end

    % von verteilen uebernommen, wird evt nicht gebraucht

```

```
    dl=round(l/2);
    dll=round(ll/2);
    db=round(b/2);
    dbb=round(bb/2);

    D=ones(l,b);
    maske=zeros(size(Feld));

    if i==1
        if j==3
            maske(1:y0+dll,1:x0+dbb)=D;
        elseif j==4
            maske(y0-dll+1:end,1:x0+dbb)=D;
        else
            maske(y0-dll+1:y0+dll,1:x0+dbb)=D;
        end

    elseif i==2
        if j==3
            maske(1:y0+dll,x0-dbb+1:end)=D;
        elseif j==4
            maske(y0-dll+1:end,x0-dbb+1:end)=D;
        else
            maske(y0-dll+1:y0+dll,x0-dbb+1:end)=D;
        end

    else
        if j==3
            maske(1:y0+dll,x0-dbb+1:x0+dbb)=D;
        elseif j==4
            maske(y0-dll+1:end,x0-dbb+1:x0+dbb)=D;
        else
            maske(y0-dll+1:y0+dll,x0-dbb+1:x0+dbb)=D;
        end
    end

    maske2=Feld>=zh;      % maske um nur das abzutragen was zu viel ist
    maske3=logical(maske(1:n,1:m).*maske2); % vereinen der beiden Masken

    feld2=Feld.*maske3;     % feld mit den abzutragenden elementen
    feld3=Feld-feld2;       % feld welches unveraendert bleiben soll
    feld2(maske3)=zh;       % abtragen um h

    feld3=feld3+feld2;      % unveraendertes und abgetragenes feld wieder vereinen

    %felddiff=Feld-feld3;    % abgetragener sand

    fv=getFeld('feld');
    %vol=volumen(felddiff,getBuerste('b'), getBuerste('l')); % Berechnung des abgetragenen Volumens
    vol1=FeldVolumen();
    setFeld(feld3,'feld');
    vol2=FeldVolumen();
    vol=vol1-vol2;
end
```

Feld: Verteilen

```
%% FeldVerteilenRechteck(vol,x0,y0,L,B)
% rechteckige Verteilung eines uebergebenen Volumens von Sand gleichmaessig
% um die Stelle x,y. Wenn sich ein Teil ausserhalb des Feldes befindet wird
% dieser Sand entfernt wodurch Sandverlust entsteht.
%
% 'vol' in cm^3
% 'x'   als Koordinaten fuer die Matrix
% 'y'   als Koordinaten fuer die Matrix
% 'L'   Laenge des Rechteckes wo Sand verteilt wird in cm
% 'B'   Breite des Rechteckes wo Sand verteilt wird in cm
%


function FeldVerteilenRechteck(vol,x0,y0,L,B)
    %L=3;          % laenge des Bereiches wo Sand verteilt wird in cm
    %B=3;         % breite des Bereiches wo Sand verteilt wird in cm

    Feld=getFeld('feld');
    %[b, l]=cm2ko(getBuerste('b'), getBuerste('l'));
    [b, l]=cm2ko(B, L);

    if mod(l,2)==1
        l=l-1;
    end
    if mod(b,2)==1
        b=b-1;
    end

    ll=l;
    bb=b;

    % verifizieren ob sich die buerste komplett im Feld befindet
    i=0;
    j=0;
    if x0-b/2<0
        i=1;
        b=b-abs(x0-b/2);
    end
    if x0+b/2>size(Feld,2)
        i=2;
        b=b/2+size(Feld,2)-x0;
    end
    if y0-l/2<0
        j=3;
        l=l-abs(y0-l/2);
    end
    if y0+l/2>size(Feld,1)
        j=4;
        l=l/2+size(Feld,1)-y0;
    end
```

```
    Vteil=ones(l,b);
    A=L*B;
    p=vol/A;
    [m n]=size(Feld);
    M=zeros(m,n);


    dl=round(l/2);
    dll=round(ll/2);
    db=round(b/2);
    dbb=round(bb/2);

    % verteilen des Sandes, dabei wird beachtet wo sich der Roboter
    % befindet falls er sich am Rande des Feldes befindet geht der Sand der
    % ausserhalb des Feldes landet verloren. Es wird davon ausgegangen,
    % dass der Roboter sich nur innerhalb des Feldes bewegt. Ausserdem
    % sollte der Roboter nie in dem Bereich sein in welchem er das aeussere
    % des Feldes antreffen kann. wird nur gemacht um Fehler und Stoerungen
    % zu vermeiden.

    if i==1
        if j==3
            M(1:y0+dll,1:x0+dbb)=p*Vteil;
        elseif j==4
            M(y0-dll+1:end,1:x0+dbb)=p*Vteil;
        else
            M(y0-dll+1:y0+dll,1:x0+dbb)=p*Vteil;
        end

    elseif i==2
        if j==3
            M(1:y0+dll,x0-dbb+1:end)=p*Vteil;
        elseif j==4
            M(y0-dll+1:end,x0-dbb+1:end)=p*Vteil;
        else
            M(y0-dll+1:y0+dll,x0-dbb+1:end)=p*Vteil;
        end

    else
        if j==3
            M(1:y0+dll,x0-dbb+1:x0+dbb)=p*Vteil;
        elseif j==4
            M(y0-dll+1:end,x0-dbb+1:x0+dbb)=p*Vteil;
        else
            M(y0-dll+1:y0+dll,x0-dbb+1:x0+dbb)=p*Vteil;
        end
    end

    setFeld(Feld+M,'feld');

end
```

Feld: Hügel und Täler finden

```
%% [H, L, anz]=FeldFindHuegel()
% Findet alle von einander getrennten Huegel (Erhoehungen). Es werden die
% Huegel in Form von Masken in einer mehrdimensionalen Matrix 'H', die
% Huegel mit zugeordneten Labels in 'L' und die Anzahl gefundener
% Erhoehungen in 'anz' zurueckgegeben.
%

function [H, L, anz]=FeldFindHuegel()
    feld=getFeld('feld');
    feld=feld(2:end-1,2:end-1);
    tol=getFeld('tol');
    hm=getFeld('hm');

    maskeh=feld>=hm+tol;

    [L, num]=bwlabel(maskeh);

    % herausfinden ob die jeweilige Erhoehung so hoch ist, dass sie zu
    % beachten ist.

    anz=0;
    ind=[];
    for i=1:num
        [x,y]=find(L==i);
        if length(x)>1
            anz=anz+1;
            ind=[ind, i];
        else
            L(x,y)=0;
        end
    end

    [m, n]=size(feld);
    H=zeros(m,n,anz);

    for i=1:anz
        for x=1:size(L,1)
            for y=1:size(L,2)
                if L(x,y)==ind(i)
                    H(x,y,i)=1;
                end
            end
        end
    end

end
```

```
%% [H, L, anz]=FeldFindTal()
% Findet alle von einander getrennten Taeler (Vertiefungen). Es werden die
% Taeler in Form von Masken in einer mehrdimensionalen Matrix 'H', die
% Taeler mit zugeordneten Labels in 'L' und die Anzahl gefundener
% Erhoehungen in 'anz' zurueckgegeben.
%

function [H, L, anz]=FeldFindTal()
    feld=getFeld('feld');
    feld=feld(2:end-1,2:end-1);
    tol=getFeld('tol');
    hm=getFeld('hm');

    masket=feld<=hm-tol;

    [L, num]=bwlabel(masket);

    % herausfinden ob die jeweilige Vertiefung so tief ist, dass sie zu
    % beachten ist.

    anz=0;
    ind=[];
    for i=1:num
        [x,y]=find(L==i);
        if length(x)>6
            anz=anz+1;
            ind=[ind, i];
        else
            L(x,y)=0;
        end
    end

    [m, n]=size(feld);
    H=zeros(m,n,anz);

    for i=1:anz
        for x=1:size(L,1)
            for y=1:size(L,2)
                if L(x,y)==ind(i)
                    H(x,y,i)=1;
                end
            end
        end
    end

end
```

7.4.2 Roboter

Roboter: initialisieren

```
%% RobotInit(x,y,w)
% den Roboter initialisieren. Standort und gefahrener Weg
% Buerste werden angegeben
%
% 'x'   x—Koordinate des Startpunktes in cm
% 'y'   y—Koordinate des Startpunktes in cm
% 'weg' bis dahin gefahrener Weg des Roboters

function RobotInit(x,y,w)
    global robot
    robot.x=x;         % darf dich zwischen 0 und 35 befinden
    robot.y=y;         % darf sich zwischen 0 und 70 befinden
    robot.weg=w;
end
```

Roboter: Bürsten in X- und Y-Richtung

```
%% RobotMoveX(xstart,ystart,v,t,d)
% Der Roboter bewegt sich vom Ort (xstart, ystart) mit der Geschwindigkeit
% v in X-Richtung t mal und schleudert dabei den Sand um den Abstand d weg.
%
% 'xstart'  x-Koordinate des Startpunktes in Koordinaten
% 'ystart'  y-Koordinate des Startpunktes in Koordinaten
% 'v'       Geschwindigkeit in x-Richtung in Koordinaten pro Zeit
% 't'       Zeit in Anzahl Durchlaeufe minus 1
% 'd'       Schleuderdistanz des Sandes in Koordinaten, wurde entfernt
%

function RobotMoveX(xstart,ystart,v,t)

    bneu=getBuerste('l');
    lneu=getBuerste('b');
    setBuerste(bneu,'b');
    setBuerste(lneu,'l');
    if v>0
        d=bneu;
    else
        d=-bneu;
    end
    for i=0:t
        x=xstart+i*v;
        y=ystart;
        RobotBewegen(x,y)
        vol=FeldAbtragen();
        FeldVerteilenRechteck(vol,x+d,y,lneu,bneu)
    end

    bneu=getBuerste('l');
    lneu=getBuerste('b');
    setBuerste(bneu,'b');
    setBuerste(lneu,'l');

end
```

```
%% RobotMoveY(xstart,ystart,v,t,d)
% Der Roboter bewegt sich vom Ort (xstart, ystart) mit der Geschwindigkeit
% v in Y-Richtung t mal und schleudert dabei den Sand um den Abstand d weg.
%
% 'xstart'  x-Koordinate des Startpunktes in Koordinaten
% 'ystart'  y-Koordinate des Startpunktes in Koordinaten
% 'v'       Geschwindigkeit in y-Richtung in Koordinaten pro Zeit
% 't'       Zeit in Anzahl Durchlaeufe minus 1
% 'd'       Schleuderdistanz des Sandes in Koordinaten, wurde entfernt
%

function RobotMoveY(xstart,ystart,v,t)
    if v>0
        d=getBuerste('l');
    else
        d=-getBuerste('l');
    end

    for i=0:t
        x=xstart;
        y=ystart+i*v;
        RobotBewegen(x,y)
        vol=FeldAbtragen();
        FeldVerteilenRechteck(vol,x,y+d,getBuerste('l'),getBuerste('b'))
    end

end
```

7.4.3 Bürste

Bürste: initialisieren

```
%% BuersteInit(b,l,h,phi)
% die Buerste initialisieren. Es wird die Breite, die Laenge,
% die Hoehe und der Drehwinkel der Buerse uebergebn.
%
% 'b'   Breite der Buerste in cm.
% 'l'   Laenge der Buerste in cm.
% 'h'   Hoehe der Buerste in cm.
% 'phi' Drehwinkel der Buerste in Grad.
%

function BuersteInit(b,l,h,phi)
    global buerste
    buerste.b=10;               % in cm
    buerste.l=5;                % in cm
    buerste.winkel=phi;         % in grad (vielleicht auch rad)
    buerste.h=h;
end
```

IRB 120
Industrieroboter

Der IRB 120 ist das neueste Produkt der vierten Robotergeneration aus dem Hause ABB und gleichzeitig der kleinste Roboter einer umfangreichen Produktfamilie. Der kompakte, leichte und sehr agile Roboter eignet sich besonders für Materialhandhabungs- und Montageprozesse auf kleinstem Raum.

Kompakt und leicht
Sein geringes Gewicht von nur 25 kg sowie seine kompakte Bauweise ermöglichen eine flexible Montage, sei es in einer Zelle, auf einer Maschine oder in unmittelbarer Nähe zu anderen Robotern in der Produktion.

Vielseitig einsetzbar
Der IRB 120 eignet sich für den Einsatz in den verschiedensten Branchen, beispielsweise in der Nahrungsmittel- und Getränkeindustrie, im Maschinenbau, in der Solarindustrie, in der medizinisch-pharmazeutischen Industrie oder in der Forschung. Eine weiße Ausführung des IRB 120 in der ISO Reinraumklasse 5 steigert seine Vielseitigkeit, da dieser Roboter auch in Umgebungen mit hohen Reinheitsstandards eingesetzt werden kann. Der 6-achsige Roboter verfügt über eine Handhabungskapazität von bis zu 3 kg (bis zu 4 kg mit vertikalem Handgelenk) und einer Reichweite von 580 mm. Der IRB 120 ist der perfekte Baustein für kostengünstige Anwendungen, insbesondere in Bereichen mit geringem Platzangebot.

Leicht zu integrieren
Mit seinem geringen Gewicht lässt sich der IRB 120 leicht und einfach integrieren. Er kann in jedem beliebigen Winkel montiert werden. Die glatten Oberflächen sind problemlos zu reinigen. Alle Druckluftschläuche und Signalleitungen sind vollständig in den Roboter integriert.

Optimierter Arbeitsbereich
Zusätzlich zu seiner außerordentlichen Reichweite von 580 mm kann der Roboter bis zu 112 mm unterhalb seiner Basis arbeiten. Sein kompakter Drehradius ermöglicht eine unmittelbare Platzierung in der Nähe anderer Automatisierungskomponenten.

Schnell, exakt und beweglich
Leichte Aluminiumkomponenten und kompakte, leistungsstarke Motoren zuammmen mit der bewährten ABB-Bewegungssteuerung ermöglichen höchste Dynamik und Genauigkeit des Roboters in allen Anwendungsbereichen.

IRC5 Compact – optimiert für kleine Roboter
Die neue Steuerungsvariante IRC5 Compact bietet die bekannten Leistungsmerkmale der bewährten IRC5 in einem kompakten, platzsparendes Format. Die einphasige Stromversorgung, externe Steckverbindungen für alle Signale sowie ein integriertes, erweiterbares E/A-System mit 16 Ein- und Ausgängen ermöglichen eine besonders einfache Installation.

Geringer Platzbedarf
Der kompakte IRB 120 in Kombination mit der Steuerungsvariante IRC5 Compact benötigt in Anwendungsbereichen mit geringem Platzangebot eine deutlich reduzierte Grundfläche.

Power and productivity
for a better world™

IRB 120

Spezifikation

Roboterversion	Reichweite	Handhabungs-kapazität	Zusätzliche Armlast
IRB 120-3/0,6	580 mm	3 kg (4 kg)*	0,3 kg

Merkmale

Montageart:	beliebig
Schutzart:	IP30
Integrierte Anwenderleitungen:	10 Signalleitungen bis zum Handgelenk
Integrierte Druckluft:	4 Druckluftleitungen (5 bar) bis zum Handgelenk
Steuerungen:	IRC5 Compact / IRC5 Kompaktsteuerung

Leistung

Positionswiederholgenauigkeit:	0,01 mm

Bewegung	Arbeitsbereich	Max. Achsgeschwindigkeit IRB 120	IRB 120T
Achse 1	+165° bis −165°	250°/s	250°/s
Achse 2	+110° bis −110°	250°/s	250°/s
Achse 3	+ 70° bis −110°	250°/s	250°/s
Achse 4	+160° bis −160°	320°/s	420°/s
Achse 5	+120° bis −120°	320°/s	590°/s
Achse 6	+400° bis −400°	420°/s	600°/s

1 kg Pick- und Place-Zyklus	IRB 120	IRB 120T
25 × 300 × 25 mm:	0,58 s	0,52 s
25 × 300 × 25 mm mit 180° Drehung Achse 6:	0,92 s	0,69 s
Beschleunigungszeit 0–1 m/s:	0,07 s	0,07 s

Elektrische Anschlüsse

Netzspannung:	200 – 600 V, 50/60 Hz
Leistungsaufnahme:	0,25 kW

Grundfläche / Gewicht

Robotergrundfläche:	180 × 180 mm
Höhe:	700 mm
Gewicht:	25 kg

*mit vertikalem Handgelenk

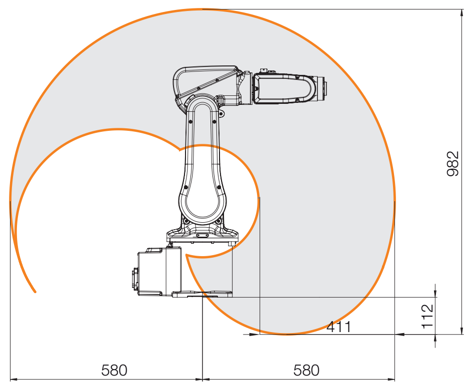
Betriebsbedingungen

Umgebungsbedingungen für die mechanische Einheit:

Umgebungstemperatur:	+5° C bis +45° C
Bei Transport und Lagerung:	−25° C bis +55° C
Kurzfristig (max. 24 Stunden):	bis zu +70° C
Relative Luftfeuchtigkeit:	max. 95 %
Optionen:	Reinraum, Klasse 5 (IPA-zertifiziert)** IPA
Geräuschpegel:	max. 70 dB (A)
Emission:	EMC / EMI-abgeschirmt

**ISO Reinraumklasse 4 kann unter bestimmten Bedingungen erreicht werden.

Arbeitsbereich



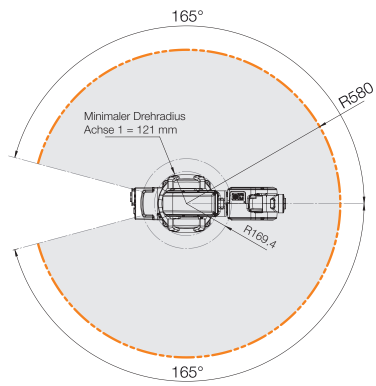


ABB Automation GmbH
Unternehmensbereich Robotics
Grüner Weg 6
D-61169 Friedberg
Phone: +49 60 31 85-0
Fax: +49 60 31 85-297
E-Mail: robotics@de.abb.com

www.abb.de/robotics

Hinweis:
Technische Änderungen der Produkte sowie Änderungen im Inhalt dieses Dokuments behalten wir uns jederzeit ohne Vorankündigung vor. Bei Bestellungen sind die jeweils vereinbarten Beschaffenheiten maßgebend. Die ABB Automation GmbH übernimmt keinerlei Verantwortung für eventuelle Fehler oder Unvollständigkeiten in diesem Dokument.

Wir behalten uns alle Rechte an diesem Dokument und den darin enthaltenen Gegenständen und Abbildungen vor. Vervielfältigung, Bekanntgabe an Dritte oder Verwertung seines Inhaltes – auch von Teilen – ist ohne vorherige schriftliche Zustimmung durch die ABB Automation GmbH verboten.

Copyright© 2012 ABB, alle Rechte vorbehalten

ROB0149DE_C August 2012

Power and productivity
for a better world™

ONLINE-DATENBLATT
www.mysick.com

Laserscanner
LMS5xx / LMS511 / Outdoor / Mid Range

LMS511-10100S01

SICK
Sensor Intelligence.

Laserscanner
LMS5xx / LMS511 / Outdoor / Mid Range

Typ > LMS511-10100S01
Artikelnr. > 1055659

Auf einen Blick

- Leistungsfähiger, effizienter Lasermesssensor für Messbereiche bis 80 m
- Herausragende Performance auch bei ungünstigen Witterungsbedingungen durch Multi-Echo-Technologie
- Kompaktes Gehäuse bis Schutzart IP 67 und eingebauter Heizung bei Outdoor-Geräten
- Geringe Leistungsaufnahme
- Schnelle Signalverarbeitung
- Mehrere Eingänge und Ausgänge
- Synchronisierung mehrerer Sensoren möglich

Ihr Nutzen

- Extrem leistungsfähig in zahlreichen Anwendungen
- Kleinster Lasermesssensor mit der höchsten Genauigkeit in dieser Sensorklasse
- Schnelle, zuverlässige Detektion von Objekten unter praktisch allen Umgebungsbedingungen
- Umfangreiche Produktfamilie mit verschiedenen Produktreihen und Typen für alle Anforderungen bezüglich Performance und Kosten
- Niedriger Stromverbrauch verringert Total Cost of Ownership
- Bestes Preis-Leistungs-Verhältnis in dieser Sensorklasse
- Schnelle und einfache Einstellung mit SOPAS Engineering Tool
- Self-Check-Funktionalität zur Erhöhung der Systemverfügbarkeit

IP67 III CE

Merkmale

Einsatzgebiet:	Outdoor
Version:	Mid Range
Variante:	PRO
Auflösungsvermögen:	Standard Resolution
Lichtquelle:	Infrarot (905 nm)
Laserklasse:	1 (IEC 60825-1 (2007-6)), augensicher
Öffnungswinkel:	190 °
Scanfrequenz:	25 Hz / 35 Hz / 50 Hz / 75 Hz / 100 Hz
Winkelauflösung:	0,167 ° 0,25 ° 0,333 ° 0,5 ° 0,667 ° 1 °

© SICK AG. Irrtümer und Änderungen vorbehalten.

Heizung:	Ja
Arbeitsbereich:	0 m ... 80 m
Reichweite bei 10 % Remission:	40 m
Spotgröße:	11,9 mrad
Anzahl der ausgewerteten Echos:	5
Nebelkorrektur:	Ja

Performance

Ansprechzeit:	≥ 10 ms
Detektierbare Objektform:	Nahezu beliebig
Systematischer Fehler [1]:	± 25 mm (1 m ... 10 m) ± 35 mm (10 m ... 20 m) ± 50 mm (20 m ... 30 m)
Statistischer Fehler [2]:	± 14 mm (20 m ... 30 m) ± 6 mm (1 m ... 10 m) ± 8 mm (10 m ... 20 m)
Integrierte Applikation:	Feldauswertung
Anzahl Feldsätze:	10 Felder
Simultane Auswertefälle:	10

[1] [2] Typischer Wert; realer Wert abhängig von Umgebungsbedingungen.

Schnittstellen

Seriell (RS-232, RS-422):	✓
Funktion (Seriell (RS-232, RS-422)):	Host
Datenübertragungsrate (Seriell (RS-232, RS-422)):	9,6 kBaud ... 500 kBaud
Ethernet:	✓
Funktion (Ethernet):	Host
Datenübertragungsrate (Ethernet):	10/100 Mbit/s
Protokoll (Ethernet):	TCP/IP, OPC
CAN-Bus:	✓
Funktion (CAN-Bus):	Erweiterung Outputs
PROFIBUS DP:	-
PROFINET:	-
DeviceNet:	-
USB:	✓
Funktion (USB):	AUX
Datenübertragungsrate (USB):	9,6 kBaud ... 500 kBaud
Bemerkung (USB):	Mini-USB
Schalteingänge:	4 (Encoder)
Schaltausgänge:	6
Optische Anzeigen:	5 LEDs (zusätzlich 7-Segment-Anzeige)

Mechanik/Elektrik

Elektrischer Anschluss:	4 4-pol. M12-Gerätebuchse (rückseitig)
Betriebsspannung:	24 V DC
Leistungsaufnahme:	22 W, + 55 W Heizung (typisch)
Gehäusefarbe:	Grau (RAL 7032)
Schutzart:	IP 67 (EN 60529, Abschnitt 14.2.7)
Schutzklasse:	III (EN 60529, Abschnitt 14.2.7)
Gewicht:	3,7 kg
Abmessungen:	160 mm x 155 mm x 185 mm

© SICK AG. Irrtümer und Änderungen vorbehalten.

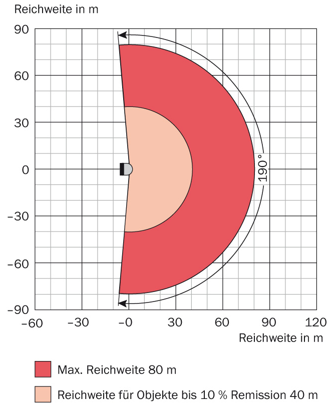
Umgebungsdaten

Objektremission:	2 % ... > 1.000 % (Reflektoren)
Elektromagnetische Verträglichkeit (EMV):	EN 61000-6-2:2005 ⁄ EN 61000-6-3 (2007-03)
Schwingfestigkeit:	EN 60068-2-6 (1995-04)
Schockfestigkeit:	EN 60068-2-27 (1993-03) ⁄ EN 60068-2-29 (1993-04)
Betriebsumgebungstemperatur:	-30 °C ... +50 °C
Lagertemperatur:	-30 °C ... +70 °C
Fremdlichtsicherheit:	70.000 lx

Allgemeine Hinweise

Hinweis zur Verwendung:	Nicht für den Personenschutz geeignet

Arbeitsbereichsdiagramm



USmod 4c int41

Australia
Phone +61 3 9457 0600
1800 334 802 – tollfree
E-Mail sales@sick.com.au

Belgium/Luxembourg
Phone +32 (0)2 466 55 66
E-Mail info@sick.be

Brasil
Phone +55 11 3215-4900
E-Mail sac@sick.com.br

Canada
Phone +1 905 771 14 44
E-Mail information@sick.com

Česká republika
Phone +420 2 57 91 18 50
E-Mail sick@sick.cz

China
Phone +86 4000 121 000
E-Mail info.china@sick.net.cn
Phone +852-2153 6300
E-Mail ghk@sick.com.hk

Danmark
Phone +45 45 82 64 00
E-Mail sick@sick.dk

Deutschland
Phone +49 211 5301-301
E-Mail info@sick.de

España
Phone +34 93 480 31 00
E-Mail info@sick.es

France
Phone +33 1 64 62 35 00
E-Mail info@sick.fr

Great Britain
Phone +44 (0)1727 831121
E-Mail info@sick.co.uk

India
Phone +91–22–4033 8333
E-Mail info@sick-india.com

Israel
Phone +972-4-6801000
E-Mail info@sick-sensors.com

Italia
Phone +39 02 27 43 41
E-Mail info@sick.it

Japan
Phone +81 (0)3 3358 1341
E-Mail support@sick.jp

Magyarország
Phone +36 1 371 2680
E-Mail office@sick.hu

Nederland
Phone +31 (0)30 229 25 44
E-Mail info@sick.nl

Norge
Phone +47 67 81 50 00
E-Mail austefjord@sick.no

Österreich
Phone +43 (0)22 36 62 28 8-0
E-Mail office@sick.at

Polska
Phone +48 22 837 40 50
E-Mail info@sick.pl

România
Phone +40 356 171 120
E-Mail office@sick.ro

Russia
Phone +7-495-775-05-30
E-Mail info@sick.ru

Schweiz
Phone +41 41 619 29 39
E-Mail contact@sick.ch

Singapore
Phone +65 6744 3732
E-Mail sales.gsg@sick.com

Slovenija
Phone +386 (0)1-47 69 990
E-Mail office@sick.si

South Africa
Phone +27 11 472 3733
E-Mail info@sickautomation.co.za

South Korea
Phone +82 2 786 6321/4
E-Mail info@sickkorea.net

Suomi
Phone +358-9-25 15 800
E-Mail sick@sick.fi

Sverige
Phone +46 10 110 10 00
E-Mail info@sick.se

Taiwan
Phone +886-2-2375-6288
E-Mail sales@sick.com.tw

Türkiye
Phone +90 (216) 528 50 00
E-Mail info@sick.com.tr

United Arab Emirates
Phone +971 (0) 4 8865 878
E-Mail info@sick.ae

USA/México
Phone +1(952) 941-6780
1 800-325-7425 – tollfree
E-Mail info@sickusa.com

More representatives and agencies
at **www.sick.com**

Sensor Intelligence.